Fast Online Segmentation of Activities from Partial Trajectories

Tariq Iqbal[1], Shen Li[1], Christopher Fourie[1], Bradley Hayes[2], and Julie A. Shah[1]

***Abstract*— Augmenting a robot with the capacity to understand the activities of the people it collaborates with in order to then label and segment those activities allows the robot to generate an efficient and safe plan for performing its own actions. In this work, we introduce an online activity segmentation algorithm that can detect activity segments by processing a partial trajectory. We model the transitions through activities as a hidden Markov model, which runs online by implementing an efficient particle-filtering approach to infer the maximum a posteriori estimate of the activity sequence. This process is complemented by an online search process to refine activity segments using task model information about the partial order of activities. We evaluated our algorithm by comparing its performance to two state-of-the-art activity segmentation algorithms on three human activity datasets. The proposed algorithm improved activity segmentation accuracy across all three datasets compared with the other two approaches, with a range from 11.3% to 65.5%, and could accurately recognize an activity through observation alone for 31.6% of the initial trajectory of that activity, on average. We also implemented the algorithm onto an industrial mobile robot during an automotive assembly task in which the robot tracked a human worker's progress and provided the worker with the correct materials at the appropriate time.**

I. Introduction

Robots currently have the capacity to help people in several fields, including health care, assisted living, and manufacturing and factory settings. In many of these scenarios, robots must share physical space and actively collaborate with humans [1–3]. The performance of many of these human-robot teams depends upon how fluently all team members can jointly perform their tasks [4, 5]. In order to successfully act within a group, people must be able to predict the intentions of other group members and use that knowledge to determine when, where, and how to act for the team's benefit [6]. In human-robot interaction scenarios, a robot similarly requires the ability to precisely identify and monitor other members' actions so that it can predict future actions and adapt its own plans accordingly [7, 8]. In particular, a robot requires the ability to segment others' activities online by detecting the start time of each activity and distinguishing it from the end time of the previous activity. This capacity is particularly crucial to efficient and safe human-robot interactions within factory environments, where humans and robots often work in close physical proximity to one another [9].

[1]The authors are with the Computer Science and Artificial Intelligence Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139, USA. {tiqbal,shenli,ckfourie}@mit.edu, julie_a_shah@csail.mit.edu

[2]The author is with the Department of Computer Science, University of Colorado Boulder, Boulder, CO 80309, USA. bradley.hayes@colorado.edu

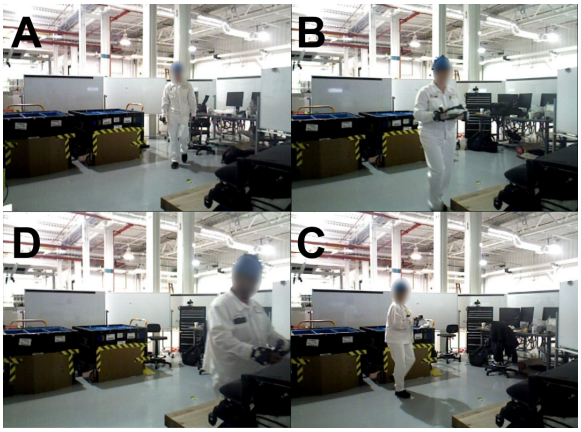

Fig. 1: Four associates perform four activities from the Auto-DA dataset: A) moving to the dashboard, B) collecting the speedometer unit, C) collecting the navigation unit, and D) placing the navigation unit onto the dashboard.

Researchers across many fields have attempted to address this concern, which is defined as the *online activity segmentation problem*. Prior techniques [10–13] are well-suited for data segmentation via post-processing but are not applicable to human-robot interaction scenarios, because either the algorithms become computationally expensive to run online, or they cannot function adequately with only a partial observance of data.

In this paper, we introduce FOSAPT (Fast Online Segmentation of Activities from Partial Trajectories), an online activity-segmentation algorithm able to identify and label activity segments by processing just a part of the full activity trajectory. We model the transitions through the activity classes as a hidden Markov model in a manner similar to Fearnhead and Liu's approach [14]. FOSAPT runs efficiently in an online setting by inferring the maximum a posteriori estimate of the sequence of activity classes, using particle filtering to make the inference tractable. However, FOSAPT's particle-filter approach is complemented by an online search process to refine activity segments via task model information about the partial order of activities, along with predictive models of the timings of future activities. Our algorithm is both computationally inexpensive and suitable for real-time implementation onto robots.

We evaluated FOSAPT by assessing its performance on three human activity datasets (UTKinect [15], Static-Reach [16], and Auto-DA) compared with two state-of-the-art activity segmentation algorithms from prior literature: the online change point detection algorithm [14] and the transition state clustering algorithm [10]. One of the datasets, Auto-DA, is derived from a real-world manufacturing task involved in automotive final assembly, in which 12 manufacturing associates performed variations of the task (see Fig. 1).

In our empirical evaluation, FOSAPT accurately segmented 72.2% of the activity segments, on average, across all four variants of the task orders in the Auto-DA dataset. We also observed that FOSAPT could accurately recognize and label an activity by observing only 31.6% of that activity's initial trajectory, on average. FOSAPT achieved an improvement to accuracy ranging from 11.3% to 65.5% compared with the other two approaches across all three datasets. Finally, we applied our algorithm to a demonstration in which a collaborative robot tracked a human worker's progress during a dashboard assembly task and provided the worker with the correct materials for assembly at the appropriate time.

II. Related Work

Researchers across many fields have attempted to address the problem of activity segmentation, with two primary approaches having been explored thus far: supervised and unsupervised learning of segmentation models. In a supervised segmentation approach, machine learning algorithms are trained with segmented and labeled activities prior to testing; in unsupervised methods, the algorithms do not require that the training trajectories be segmented and labeled.

Many supervised approaches involve two primary steps for activity segmentation: extracting spatiotemporal features from the data source, then training temporal models to identify segments. However, many temporal classifier models do not scale well for identifying long-range temporal patterns. To address this challenge, Lea et al. [17] developed temporal convolutional networks (TCNs), a class of time-series model, that uses a hierarchy of temporal convolutions to perform fine-grained action segmentation. They presented two types of TCNs, and using a combination of both, their method was able to segment video recordings of activities via temporal relations among the visual features. Similarly, Kuehne et al. [18] developed a generative approach to human activity segmentation, which used reduced Fisher vectors as features and combined these features with an HMM to segment activities temporally into action units. These methods work well when full trajectory information is available during the segmentation process; however, they are not suitable for online segmentation scenarios in which segmentation must be performed on partial data, which is the focus of our interest.

On the other hand, many unsupervised segmentation approaches have been proposed in the literature in which algorithms attempt to discover underlying structures within the data to cluster similar trajectories together. For example, Zhou et al. [19] developed hierarchical aligned cluster analysis (HACA), a hierarchical clustering algorithm designed to perform temporal segmentation of human motion. Their approach identifies a number of disjoint segments within multidimensional time series data and assigns similar segments into a cluster. Krishnan et al. [10] introduced the transition state clustering (TSC) method, an unsupervised segmentation approach that clusters transition states together from a set of demonstrations. TSC assumes that the low-level dynamics of a segment are noisy, but that high-level dynamics follow a consistent, spatially and temporally correlated partial order of events across demonstrations. The models used in many unsupervised approaches assume particular structures within the prior probabilities and trajectory data, and an unsupervised method designed for one dataset often fails to perform satisfactorily on another dataset with a different underlying data structure.

Another prominent segmentation approach involves statistical model-based changepoint detection algorithms. For example, Fox et al. [12] developed the Beta Process Autoregressive HMM (BP-AR-HMM), a Bayesian, non-parametric approach to jointly modeling multiple related trajectory time-series. Neikum et al. [20] utilized BP-AR-HMM to partition demonstrations into segments within their developed learning from demonstration framework and deployed it onto a PR2 robotic platform. In the same vein, Fearnhead and Liu [14, 21] developed a statistical approach to online changepoint detection problems by introducing a sampling method similar to particle filters to reduce computational cost. They assumed that if an observation sequence and a set of candidate models are given, then the observation sequence is generated from specific underlying models, and the points at which the underlying models change are detected as changepoints. Konidaris et al. [22] built upon this idea and implemented a model-based changepoint detection process for constructing skill trees to acquire skills from human demonstrations. However, these techniques were designed for offline processing of full trajectory data.

Although these methods work for activity segmentation in various scenarios, they either become computationally expensive to run online or do not perform adequately with only partial observance of data. In order to address these problems, we incorporate a supervised learning model with a statistical changepoint detection algorithm and a particle filtering-based sampling approach to performing online human activity segmentation onto a robot. This algorithm is able to function with only partial observance of trajectory data, as we describe in detail in Section III.

III. Methodology

In this section, we introduce the FOSAPT (Fast Online Segmentation of Activities from Partial Trajectories) algorithm. This segmentation algorithm takes a partial trajectory as input and can recognize and track activity segments as it progresses in real time. FOSAPT is also capable of further refining the start and the end times of an activity accurately as that activity ends. This approach is both computationally inexpensive and suitable for real-time implementation onto robots.

We present the algorithm in Algo 1 and formulate the problem of online activity segmentation in Section III-A. We also provide an overview of the algorithm in Section III-B and discuss it in greater detail in Sections III-C-III-K.

A. Problem Formulation

Activity segmentation involves identifying the start and end times of each activity segment within a trajectory and assigning the appropriate corresponding activity label to each segment. Given a set of trajectory spanning time $t = 1$ to

T with corresponding trajectory frames $F = (f_1, f_2, \ldots, f_T)$ and a set of activity labels $A = (a_1, a_2, \ldots, a_m)$, the activity segmentation problem is defined as follows: identifying the mapping of trajectory frames to activity labels. Activity classification represents a special case of activity segmentation wherein the activity's start and end times are given and the activity label must be assigned. In this work, we are interested in online activity segmentation with only partial observance of trajectory data.

We define the times at which an activity label changes in the trajectory frames as denoted by changepoints, τ. Consider that the trajectory data consists of n activities; we can then denote $(0 < \tau_1 < \tau_2 < \ldots < \tau_{n-1} < T)$ and $\tau_0 = 0$ and $\tau_n = T$ as the changepoints in trajectory frames.

The trajectory frames between two changepoints are denoted as a segment, s. We define a segment of frames as $s(i, j) = (f_i, \ldots, f_j)$, where $\forall f \in F$ and $i = \tau_p$ and $j = \tau_{p+1}$. Thus, given a set of activities A, activity segmentation produces a set S that contains an allocation of non-overlapping frame intervals in F with labels drawn from A. Thus, $s(i, j, a)$ $\forall s \in S$, $i, j \in T$ such that $i < j$, $a \in A$.

Online partial trajectory activity segmentation assumes a partial set of trajectories $s(i, k) = (f_i, \ldots, f_k)$ as input, where $i = \tau_p$, $k \leq j = \tau_{p+1}$. The output is $s(i, k, a)$ such that $s(i, k, a) \subseteq s(i, j, a)$, $(i, k, j) \in T$, $(i < k \leq j)$, and $a \in A$.

B. Approach in a Nutshell

FOSAPT relies upon a small portion of data as it arrives and checks the likelihood of the label of that segment via a set of trained activity classifiers. It then incrementally computes the likelihood value of a larger segment from smaller segment likelihoods (instead of calculating the likelihood of the larger segment again). In FOSAPT, transitions through the activity classes are modeled as a hidden Markov model. It runs efficiently in an online setting by inferring the maximum a posteriori estimate of the sequence of activity classes via a particle-filtering approach. This method is complemented by an online search process to refine activity segments using task model information about the partial order of activities, as well as predictive models of the timings of future activities.

C. Activity Likelihood Calculation from Partial Trajectories

This section details Algo 1, Line 3-5. FOSAPT utilizes an activity classification algorithm to measure the activity likelihood of a set of trajectory frames, described as the *likelihood* function (Line 5). In this implementation, we utilize RAPTOR [23], a real-time, state-of-the-art activity classifier; however, FOSAPT is not dependent upon any one particular activity classification algorithm.

There are multiple advantages to using RAPTOR in FOSAPT. First, RAPTOR is capable of training an ensemble of sub-classifiers for each activity class. Thus, given a set of trajectory frames, RAPTOR can provide a classification likelihood for each activity. Second, it is fast and computationally inexpensive, and thus implementable into real-time systems.

We first train a set of activity classifiers $C = (c_{a_1}, c_{a_2}, \ldots, c_{a_m})$ for each $a_i \in A$, where $m = |A|$, on a corpus of recorded trajectory frames, such that activity classifier c_{a_i} is trained for activity a_i. During the training phase, we train RAPTOR with an ensemble of sub-classifiers using a part of the full trajectory frames, and combine those temporally to generate a full activity classifier, similar to the idea proposed by Hayes and Shah [23]. We divide activity a_i into B sub-classes. We then train $(c^1_{a_i}, c^2_{a_i}, \ldots, c^B_{a_i})$ sub-classifiers separately and combine them temporally to generate classifier c_{a_i} for activity a_i. A training process such as this permits the algorithm to utilize sub-classifiers directly (e.g., check the likelihood of trajectory frames $(f_i, \ldots, f_k)$ with $c^b_{a_i}$, where $1 \leq b \leq B$ for activity a_i.

From the training data, FOSAPT models the marginal probability of each activity length with a probability mass function $g(.)$. Thus, we can define $P(\tau_k - \tau_{(k-1)} = d) = g_{a_i}(d)$, where $g(.)$ is a discrete distribution on the length of activity a_i. The corresponding cumulative distribution function is $G_{a_i}(l) = \sum_{i=1}^{d} g_{a_i}(i)$ [14]. Each activity classifier is a temporal ensemble of B sub-classifiers, and FOSAPT models the bin lengths as β from the training data, where $\beta(a)$ denotes the bin length of activity a. We explain how to dynamically adjust this value in Section III-J.

During the testing phase, FOSAPT utilizes these sub-classifiers to measure the likelihood of a partial set of trajectory frames being an activity. If ΔF denotes a partial set of trajectory frames from $(f_{j+1}, \ldots, f_t)$, then the algorithm first checks whether the length of the trajectory frames ($|\Delta F|$) is greater than or equal to the length of the bin size of that activity, before performing the likelihood computation (Line 3). FOSAPT then tests the likelihood of ΔF being an activity a with each sub-classifier C^b_a, where $1 \leq b \leq B$ (Lines 4 and 5).

D. Task Structure Modeling

For a set of trajectory frames (ΔF), were the algorithm to check all possible combinations of activity classes ($\forall c_a$ for $a \in A$), the process would be computationally expensive. To reduce the cost of this computation, FOSAPT leverages prior knowledge of the task's sequence of activities to reduce the number of classifiers on which the trajectory frames are

Algorithm 1 FOSAPT

Input: Partial trajectory $< \Delta F, j, t >$, Activity set $< A >$, Activity classifiers $< C >$, Number of sub-classes $< B >$, Bin lengths $< \beta >$, Particles $< P >$, Previously detected segments $< S >$, Future activity list $< FAL >$
Output: Activity segments $< S >$, Particles $< P >$, Bin lengths $< \beta >$

1: $pList \leftarrow \phi$ ▷ temporary particle list
2: **for** $a \in FAL$ **do**
3: **if** $(|\Delta F| \geq \beta(a))$ **then**
4: **for** $b \leftarrow 1 : B$ **do**
5: $l \leftarrow$ likelihood $(C^b_a, \Delta F)$ ▷ classifier fitness
6: $pr \leftarrow$ prior (a, FAL) ▷ task structure
7: $pmp \leftarrow$ prev_MAP (P, j, t, A) ▷ find MAP particle ends at j
8: $P_t(j, a) \leftarrow (1 - G_a(t - j - 1))$ * l * pr * $pmp.MAP$
9: $MAP \leftarrow P_t(j, a)$ * $g_a(t - j)$ / $(1 - G_a(t - j - 1))$
10: $np \leftarrow$ Particle $(a, b, j, t, l, P_t(j, a), MAP, pmp)$
11: $pList = pList + np$
12: **for** $p \in pList$ **do**
13: $P \leftarrow$ merge_or_insert (p, P) ▷ merge with other particles, or insert in P
14: $P \leftarrow$ resample (P) ▷ resample particles
15: $S \leftarrow$ find_all_segments (P, S) ▷ find all activity segments
16: $\beta \leftarrow$ dynamic_bin_length_adjustments (β, S) ▷ adjust bin lengths from evidence
17: refine_activity_segments (S) ▷ this step runs in parallel
18: return S, P, β

tested. Thus, a classifier is only used to evaluate a trajectory when a task is supposed to occur at that moment or at a time in the near future. As a result, this technique significantly reduces the number of classifiers required for testing.

The algorithm constructs a hierarchical task network similar to the clique/chain hierarchical task network (CC-HTN) proposed by Hayes et al. [24]. FOSAPT builds this network using the high-level task sequences of activities from the training demonstrations, and generates a *future_activity_list (FAL)* incorporating the possible activities that could happen at a given moment of time from this task network, including the current activity. Thus, FOSAPT only tests the trajectory frames for activities that are present in the *FAL* (Line 2), reducing the number of expensive *likelihood* computations.

E. Activity Prior Computation

The algorithm computes a *prior* probability ($pr(a)$) for an activity (a), described as $prior(a, FAL)$ (Line 6), utilizing the *FAL*. This value represents the probability of an activity occurring at that time. One can learn any prior probability distribution of an activity from the training data. In our case, the algorithm assumes that all the activities in the *FAL* are equally likely; thus, FOSAPT generates a uniform distribution over the number of activities in this list.

F. Activity Transition Modeling

Now, taking inspiration from the statistical changepoint detection method developed by Fearnhead and Liu [14, 21], as well as the follow-on work by Konidaris et al. [22], FOSAPT can utilize these values to model the activity transitions as a hidden Markov process. Here, the observed state at time t is the trajectory frame, $\{f_t\}$, and the hidden state is the activity, a_t. The probability of a set of trajectory frames that starts at time $(j+1)$ and ends at t ($\Delta F = f_{j+1}, \ldots, f_t$) being activity a_t can be modeled as the product of the likelihood of ΔF being activity a_t (l, measured in Line 5) and the probability of the segment lasting for $(t-j-1)$ time steps. Thus, this probability can be defined as $P(\Delta F|a_t) = l*(1-G_{a_t}(t-j-1))$. Similarly, the transition probability of the activities can be defined as $T(a_j, a_t) = g_{a_t}(t-j-1) * pr(a_t)$, where the transition from activity a_j at time j to activity a_t occurs at time t, and $pr(a_t)$ represents the prior probability of activity a_t (Line 6) [14, 22].

FOSAPT can now compute the maximum likelihood sequence for the activities (hidden states) given their transition probability and the trajectory frames (observation). This enables the algorithm to use an online Viterbi algorithm to calculate the MAP estimate of the activity changepoint positions and the orders. Thus, we compute the probability of an activity a starting at time $(j+1)$ and continuing at time t ($P_t(j,a)$) as follows:

$$P_t(j,a) = (1 - G_a(t-j-1)) * l * pr(a) * P_j^{MAP} \quad (1)$$

$$P_j^{MAP} = \max_{k,a} \frac{P_j(k,a) g_a(j-k)}{(1 - G_a(j-k-1))} \quad (2)$$

We can calculate this probability for each $k = 0, \ldots, t-1$ and for all $a \in A$ in Eq. 1. The values of k and a that maximize the P_j^{MAP} value in Eq. 2 represent the activity changepoint and the activity for the current segment at time j. Thus, we can reduce total computation time while calculating the value of P_t^{MAP} repeatedly for each time step t, changepoints $j < t$ and for each activity $a \in A$ by storing each value for the next computation.

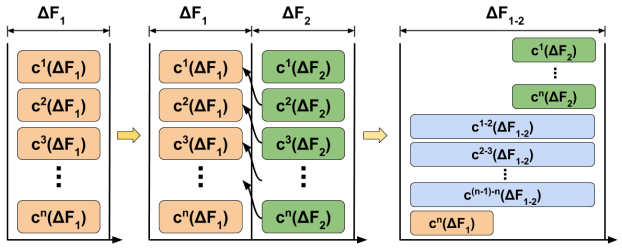

Fig. 2: A depiction of the particle generation and merging process. Left) Particles are generated by computing the sub-classifier likelihoods from ΔF_1 frames. Center) New particles are generated from ΔF_2 frames (green) and merged with the previously generated particles (orange). Right) Merged, new, and old particles are in the particle list (new particles from ΔF_2 in green, merged particles in blue, and old particles from ΔF_1 in orange).

A particle filter is employed to efficiently keep the computation tractable [14], where each particle represents a segment with activity a that starts at time $(j+1)$ and ends at time t. Each particle stores the activity (a), the start and the end bin numbers (b), the start time and the end of that activity segment (j and t), the activity likelihood value (l), $P_t(j,a)$, the MAP value, and the maximum likelihood sequence.

FOSAPT first computes the MAP particle that ends at time j according to Eq. 2 from previously generated particles (Line 7). The algorithm then computes $P_t(j,a)$ and the MAP value according to Eq. 1 and Eq.2 (Lines 8-9). It then generates a new particle and stores it in a temporary particle list (Lines 10-11).

It is computationally expensive to compute $P_t(j,a)$ for each time step in real time. In the following section, we describe an online algorithm that computes $P_t(j,a)$ incrementally to make it implementable onto a robot.

G. Incremental $P_t(j,a)$ Computation Through Merging

Consider a situation in which a_i is currently executing. The algorithm waits until it receives a number of trajectory frames equal to the bin size of that activity ($\beta(a_i)$). Next, the algorithm receives a small segment of trajectory frames, $\Delta F_1 \sim (f_{i_1}, \ldots, f_{j_1})$, which is the size of the bin size of activity a_i. Given segment ΔF_1, the algorithm does not have any information regarding how far the activity has already progressed, which would enable it to check a specific sub-classifier. Thus, it checks the likelihood of the segment ΔF_1 by testing against all the sub-classifiers ($c_{a_i}^b$) of that activity where $b = 1, \ldots, B$ (Line 4).

After measuring the likelihood values for all sub-classifiers, the algorithm generates B particles. Suppose these particles are called $(c_{a_i}^1(\Delta F_1), c_{a_i}^2(\Delta F_1), \ldots, c_{a_i}^B(\Delta F_1))$; the algorithm stores all particles in a particle list, P (Line 13, *insert* operation).

After generating the particles, the algorithm again waits for another segment of trajectory frames – for example, $\Delta F_2 \sim (f_{i_2}, \ldots, f_{j_2})$. Following a similar pro-

cess, the algorithm then generates another B particles, with each particle storing the likelihood of segment ΔF_2 tested against all sub-classifiers; these particles are called $(c^1_{a_i}(\Delta F_2), c^2_{a_i}(\Delta F_2), \ldots, c^B_{a_i}(\Delta F_2))$.

Next, the algorithm performs a *merge* operation. The idea behind this merge is that the algorithm computes the likelihood of a larger segment from already-computed likelihoods of smaller segments. This operation is computationally inexpensive, but enables the algorithm to efficiently compute the likelihood of larger segments while concurrently maintaining multiple possible hypotheses.

Particles must be temporally adjacent in order for the algorithm to merge them. Two particles are considered temporally adjacent when they contain the likelihood values computed from two adjacent sub-classifiers of the same activity. For example, the algorithm would merge particles $(c^1_{a_i}(\Delta F_1)$ and $c^2_{a_i}(\Delta F_2))$, as the likelihood value of particle $(c^1_{a_i}(\Delta F_1)$ is computed from sub-classifier 1 $(c^1_{a_i})$ and the likelihood value of particle $(c^2_{a_i}(\Delta F_1)$ is computed from sub-classifier 2 $(c^2_{a_i})$. As such, during this process, the algorithm merges $c^2_{a_i}(\Delta F_2)$ with $c^1_{a_i}(\Delta F_1)$, and the $c^1_{a_i}(\Delta F_1)$ particle becomes $c^{(1-2)}_{a_i}(\Delta F_{(1-2)})$, while $c^2_{a_i}(\Delta F_2))$ does not change.

To compute the $P_t(j,a)$ of this merged particle $(c^{(1-2)}_{a_i}(\Delta F_{(1-2)}))$, FOSAPT utilizes the computed likelihood values of the $c^1_{a_i}(\Delta F_1$ and $c^2_{a_i}(\Delta F_2)$ particles. As the trajectory segments are temporally adjacent but independent, the activity likelihood values of the merged trajectory segment $(\Delta F_1 + \Delta F_2)$ are computed by taking a product of the activity likelihood of both segments and then normalizing it over the number of segments (Line 5). Similar to RAPTOR, FOSAPT performs a max-polling operation on the likelihood values before performing this computation. The algorithm updates the values of other statistics by following the steps presented in Lines 6-9 for the merged segment. (We present this merging step in Fig. 2.) After merging the eligible particles, the algorithm updates the particle list, P.

H. Resampling

To keep the computation tractable, when the number of particles ($|P|$) reaches a predefined threshold (R_N), FOSAPT uses a resampling algorithm to filter the particle number to a fixed value (R_M) (Line 14). It applies the following three criteria to reduce the number of particles after the *merge* step.

First, the algorithm removes the particles with a starting sub-classifier number higher than a threshold, as these particles represent the final portion of an activity and are not likely to produce a larger segment to represent a major part of the activity in the future. Second, the algorithm removes any segments that are not updated for a set time period. In our implementation, FOSAPT measured the longest time each activity takes from the training data, summed this value with the square root of the standard deviation value of that activity time from the training trajectory, and set the result as the maximum allowable time for a particle of that activity to be alive. Finally, after applying both of the previous filtering methods, if the total number of particles is higher than R_M, FOSAPT prunes $m\%$ of the particles with the lowest likelihood.

I. Compute Activity Segments

After completing the particle generation process, the algorithm's goal is to identify the activity sequence from the observed trajectory up to that time step. To do so, FOSAPT first finds the particle within the particle list (P) with the highest $P_t(j,a)$ value at time t. We impose additional constraints that a particle must be longer than a predefined number of bin lengths (β) and the likelihood value of that particle must be greater than a likelihood threshold value (ϕ) before it can be selected as a candidate for an activity. After finding the particle with the highest $P_t(j,a)$ value at a given time point, the algorithm adds this particle to a *max_particle_path*, which continues to grow as each new *max_MAP_particle* is found. All the activity segments are computed by backtracking the *max_particle_path* (Line 15).

J. Activity Time Prediction and Dynamic β Adjustment

FOSAPT updates its predictions about activity timing to improve segmentation performance. If a test activity trajectory is longer or shorter than the training trajectories, the bin length (β) of that activity must be adjusted accordingly to compute accurate likelihood values from the sub-classifiers. To address this challenge, we designed an algorithm that generates predictions about the timing of future activities based on observed timings of previous activities and adjusts the bin length (β) according to these prediction values.

A Gaussian Mixture Model (GMM) is used to model temporal predictions of an activity time. The GMM can be formally defined as a normalized weighted sum of Gaussian modes, as follows:

$$p(\mathbf{x}|\lambda) = \sum_{i=1}^{M} w_i \cdot g(\mathbf{x}|\mu_i, \Sigma_i), \qquad \text{s.t.} \sum_{i=1}^{M} w_i \equiv 1 \quad (3)$$

$g(\mathbf{x}|\mu_i, \Sigma_i)$ is an n-dimensional Gaussian, defined as follows:

$$g(\mathbf{x}|\cdots) = \frac{1}{(2\pi)^{\frac{D}{2}} |\Sigma_i|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\mu_i)^T \Sigma_i^{-1}(\mathbf{x}-\mu_i)\right\}$$

To facilitate prediction based on learned data, we use the conditional decomposition of the model. This allows the algorithm to incorporate accumulated evidence as it is obtained from the activity segmentation. The conditional of the GMM can be calculated as follows:

$$p(\mathbf{x}_s|\mathbf{x}_d) = \sum_k w_k \frac{p_k(\mathbf{x}_d)}{p(\mathbf{x}_d)} \cdot p_k(\mathbf{x}_s|\mathbf{x}_d) \quad (4)$$

$\mathbf{x}_s$ represents the variables over which inference is to be performed, and $\mathbf{x}_d$ represents the variables for which evidence exists. This equation illustrates that the conditional of a sum is the weighted sum of the individual conditionals of the mode, with the weight defined by a ratio of the marginals. (Note that this outcome is independent of the modal distribution, and while Gaussian functions are employed here, additional or different distributions could be used instead.)

As more information is obtained over the course of the testing phase, the model can be updated to form a new distribution

with different means and variances. The conventional method for forming a prediction would be to use the maximum of the conditional distribution. However, to hasten inference, our approach incorporates the mean of the highest weighted mode in lieu of the true maximum, which is then interpreted as the prediction. This assumption may result in the incorrect maximum being selected if modes closely overlap.

After computing the timing of the current activity and predicting the timing of future activities, our algorithm adjusts the bin length (β) of each activity, while the number of sub-classifiers (B) remains the same. This process (Line 16) makes the algorithm robust to any deviations from the trained models with regard to activity timing.

K. Refine Activity Segments

As activity segmentation progresses, FOSAPT performs another concurrent search within previously detected activity segments to refine its assessment of the likely start and end time points. This search reduces the chance of erroneous detection near the activity transition points, and helps to determine the accurate transition points to nearby activities.

Suppose that FOSAPT had previously detected segment s_1 for activity a_1. To search for an accurate bound, the algorithm would move the beginning and ending of segment s_1 forward and backward by a portion of the frames of that segment length to identify the part with the highest likelihood value. When the algorithm identifies the part with the highest likelihood value within that segment, it reports that as the accurate segmentation of that activity.

As this segmentation search is only employed on completed activity segments, it does not interfere with the current activity segmentation approach; this step runs in parallel with the current activity segmentation steps and provides more-accurate segmentation results. To reduce computation, this step is only executed on an activity segment when that segment does not change in the *activity segment list, S,* for a predefined amount of time. Were the activity sequence to change in the future, the refinement process would be executed again on *S* (Line 17).

The overall time complexity of FOSAPT is $O(|FAL| * B * |\Delta F| + |P|)$.

IV. Experimental validation

We evaluated FOSAPT's performance using three activity datasets: a motion-capture dataset of the automotive dashboard assembly process of real industry associates, a publicly available single-person activity dataset collected using a Microsoft Kinect sensor (UTKinect [15]), and reaching behavior during a stationary manufacturing task recorded with a motion-capture system (Static-Reach [16]).

We also evaluated FOSAPT's performance by comparing it against two activity segmentation methods from prior literature: the Change Point Detection algorithm developed by Fearnhead and Liu [14], which uses a similar statistical approach to FOSAPT for tracking activities; and the Transition State Clustering (TSC) algorithm by Krishnan et al. [10], a state-of-the-art activity-clustering algorithm. Although TSC is unsupervised, it is reasonable to compare FOSAPT's performance with this method, as results from prior work indicate that TSC demonstrated state-of-the-art accuracy when detecting activity segments both in simulation and using data from robot demonstrations [10].

In collaboration with an industry partner, we designed a test scenario for the assembly of a car dashboard; we call this scenario the "automotive dashboard assembly" dataset, or "Auto-DA." This dataset included the following activity classes: move to the dashboard (*mv_dash*), move to receive a speedometer object (*mv_meter*), collect the speedometer (*col_meter*), place the speedometer on the dashboard (*pl_meter*), move to receive a navigation unit (*mv_nav*), collect the navigation unit (*col_nav*), place the navigation unit to the dashboard (*pl_nav*), and exit from the space (*exit*).

To collect data, we set up a testbed data collection facility within our industry partner's automotive assembly test factory. A total of 12 associates factory participated in the data collection process, including ten males and two females. Each task sequence was 54.4 seconds long, on average.

Each associate participated in a total of four variations of the assembly task. An associate performed all eight activities sequentially during each of the task orders. The positions and orientations of a total of seven objects (left hand, right hand, head, dashboard, speedometer, navigation unit, and a scanner gun) were tracked using a VICON motion capture system, with the data recorded at a frame rate of 30Hz.

The UTKinect dataset consists of a total of 10 activity classes performed by 10 people [15]. We combined similar activities into the following five activity classes performed in sequence: walk, sit (involving sitting down and standing up), pick up and transport object, shake object (involving throwing, pushing, pulling), and gesture with hands (involving waving and clapping of hands). The dataset contains 20 skeleton joint positions of a person tracked with a Kinect sensor. As the activities were not continuous (in that undefined activities took place between labeled activities), we removed the undefined activities from each trial, merged all the activities sequentially for evaluation purposes, and sampled at 15Hz. One trial was not included in the data, as it did not contain all the activities on the list.

The Static-Reach dataset was recorded via a PhaseSpace motion capture system during a collaborative task performed by a human-robot team [16]. Each trial consists of 16 sequential human activities, during which the participant would reach toward and retract from eight different locations on a table. Similar to the UTKinect dataset, we merged the sequential activities for evaluation purposes and sampled at 120Hz.

V. Results

A. Evaluation metrics

We first measured the intersection-over-union (*IoU*) scores of the algorithm, following a process similar to that used in prior work ([11] and [10]). For example, if the algorithm segments an activity from a trajectory as *s* (representing the time duration from a starting time point to an ending point)

TABLE I: Activity segmentation *accuracy* of the FOSAPT algorithm applied to the Auto-DA dataset for all task orders

	Segmentation accuracy (%)
Task order 1	68.8
Task order 2	65.6
Task order 3	80.2
Task order 4	74.1
Average	72.2

and the ground-truth activity segment is *GT*, then the *IoU* is measured as $IoU = (s \cap GT)/(s \cup GT)$.

For our Auto-DA dataset, we had two people label each activity segment; thus, the ground-truth segments often varied slightly between raters. During the GT calculation process, we took the average of each time point (the start and end times of a ground-truth segment) labeled by the annotators as the ground-truth value.

We then computed segment accuracies by following an approach similar to those taken by Wu et al. [11] and Krishnan et al. [10]. We considered a segment to be detected accurately if the IoU value of that segment was higher than a threshold (δ). As we used real-world datasets, in keeping with Wu et al. [11], we set $\delta = 0.4$.

To evaluate the accuracy of FOSAPT and the CPD algorithm, we performed leave-one-out cross-validation across the trials for each dataset. As TSC is an unsupervised segmentation algorithm, we provided all trajectories of each dataset as input, and report the mean accuracy of five runs.

B. Accuracy of FOSAPT

We measured FOSAPT's activity-segmentation accuracy across all four task orders within the Auto-DA dataset, and present the results in Table I. We also depict the rate at which each activity was accurately segmented in Table II, Col 2. For Auto-DA dataset, we utilized a sequential task structure and set $|FAL| = 3, B = 15, R_N = 120, R_M = 100, \phi = -5.0$.

The results in Table I indicate that FOSAPT detected segments with an accuracy of 72.2% for the Auto-DA dataset, with variation from 65.6% to 80.2% depending upon the task orders. The results in Table II, Col 2 show that FOSAPT's activity segmentation accuracy varied from 50.0% to 90.0% depending upon task type.

We implemented a version of the changepoint detection algorithm developed by Fearnhead and Liu [14] for online use. To determine the likelihood value of a segment, we used RAPTOR, which helped us to make a fair comparison between CPDs and FOSAPTs performances. Because CPD does not include a method for incrementally computing likelihood values from an activity classifier, we instead implemented a variant in which we gradually generated particles of varied lengths in a constant frame interval and checked the likelihood of the given segment, considering that as a full activity trajectory from a classifier.

We did not restrict the number of activities while generating particles via the CPD method; however, as CPD is unable to incrementally update particles likelihood, it must generate particles of various lengths. Thus, when CPD had to generate particles for all activity types, the segmentation process became very slow, and it was unable to finish within 10 minutes (each task was 54.4 sec long, on average). Therefore, we incorporated the *FAL* (in Section III-D) and ϕ (in Section III-I) to keep computation tractable. We set $|FAL| = 3$ and $\phi = -5.0$, same as the setting for FOSAPT.

TABLE II: Activity segmentation accuracy (%) of the FOSAPT algorithm and the percentage of initial frames of the whole trajectory it observed to accurately detect an activity applied to the Auto-DA dataset

	Segmentation accuracy (%)	Initial frames (%) to detect
mv_dash	78.0	14.3
mv_meter	76.0	23.4
col_meter	90.0	31.4
pl_meter	66.0	37.4
mv_nav	68.0	47.8
col_nav	82.0	32.5
pl_nav	68.0	50.1
exit	50.0	14.8
Average	72.2	31.6

We set the following parameters of TSC for all datasets, in keeping with the parameters used in Krishnan et al. [10]: $window\_size = 2$, $normalization = False$, and $pruning = 0.8$, and followed a similar approach for activity label generation for the segments taken in [10].

We report segmentation accuracy across all three datasets for FOSAPT, the changepoint detection (CPD) algorithm, and the transition state clustering algorithm (TSC) in Table III. The results suggest that FOSAPT was more accurate than either CPD or TSC across all three datasets, and that FOSAPT achieved an improvement in accuracy ranging from 11.3% to 65.5% vs. CPD and TSC. It also demonstrated accuracy as high as 88.8% for the Static-Reach dataset.

We also present the number of missing activity segments reported by the algorithms in Table IV. The number of missing segments was calculated by counting the number of segments from the ground-truth data that the algorithms did not reported. Our findings suggest that FOSAPT failed to detect segments in fewer cases than either CPD or TSC in both the Auto-DA and UTKinect datasets (5.8% and 0.0%, respectively). CPD yielded fewer missing segments (3.8%) than FOSAPT (4.4%) for the Static-Reach dataset; however, CPD also demonstrated a lower accuracy within that dataset (20.0% accuracy compared to 88.8% with FOSAPT).

C. Performance with partial observance of trajectory data

FOSAPT is capable of detecting activities by observing just a partial trajectory of the full demonstration. We first measured how long our algorithm took to report the ground-truth activity label after the start of the activity. We then subtracted the ground-truth start time of each activity from the time when FOSAPT first reported that activity, and normalized it using the total duration of that ground-truth activity segment. Here, we only considered activity segments that were accurately segmented by FOSAPT. The results, presented in Table II, Col 3, indicate that FOSAPT can segment out activities just by observing 31.6% of the full trajectory of an activity (on average).

VI. Discussion

FOSAPT outperformed all other evaluated baselines across all datasets and metrics tested. It performed best on the

TABLE III: Accuracy (%) of FOSAPT, CPD and TSC across three datasets

	Auto-DA	UTKinect	Static-Reach
FOSAPT	72.2	70.5	88.8
CPD	42.9	43.4	20.0
TSC	6.7	59.2	69.6

Static-Reach dataset (88.8% accuracy), as most of the demonstrations of this dataset followed similar paths during activity executions and exhibited relatively less jerky motion. Thus, the computed likelihood values were less ambiguous between consecutive activities, aiding in the accurate identification of activity segments. On the other hand, in the case of the UTKinect dataset (70.5% accuracy), various people performed similar actions in different ways, which could have contributed to ambiguous likelihood values for the changepoint positions and resulted in less-accurate performance by FOSAPT.

TSC demonstrated reasonably high accuracy for the UTKinect and Static-Reach datasets (59.2% and 69.6%, respectively); however, it only achieved 6.7% accuracy on Auto-DA. In the UTKinect and Static-Reach datasets, non-activity segments were removed and only labeled activity segments were temporally combined; thus, in many instances, there was a substantial change to the trajectories near the activity changepoints. As TSC clusters similar data patterns, this change might contribute to better activity segmentation. However, this was not the case for the Auto-DA dataset, as it contains continuous trajectory frames for each demonstration, and TSC failed to find appropriate changepoints across the demonstrations, as each person might perform the same activity differently in space and time.

On the other hand, CPD demonstrated segmentation accuracy of 42.9% for the Auto-DA dataset and 43.4% for the UTKinect dataset, but of only 20.0% on the Static-Reach dataset. As CPD is an online algorithm, the frame rate contributed to its performance: the Static-Reach dataset had a higher frame rate (120 Hz) than the other datasets (30 Hz for Auto-DA and 15 Hz for UTKinect); thus, CPD had to generate more particles with a small degree of variation on the data, which could result in many inaccurate segment detections.

VII. Robot Demonstration

We applied FOSAPT in an industrial setting via a robot demonstration. In this scenario, a collaborative industrial robot tracked a human worker's progress through a dashboard assembly task using FOSAPT, providing the person with the right materials for assembly at the appropriate time. A video of the demonstration is available here: http://tiny.cc/FOSAPT.

VIII. Conclusion

In this work, we presented FOSAPT, an online activity segmentation algorithm capable of accurately identifying and labeling activity segments. We modeled the transitions between activity classes as a hidden Markov model in FOSAPT, which inferred the maximum a posteriori estimate of the sequence of activity classes via a particle-filtering approach and predicted the timing of future activities, complemented by an online search process to refine activity segments via task model information about the partial order of activities. FOSAPT demonstrated improved segmentation accuracy compared with two state-of-the-art segmentation algorithms, and was able to segment activities only by processing a part of the full activity trajectory (31.6% of the initial trajectory) while running online. We implemented the algorithm in a collaborative industrial robot, in a scenario in which the robot tracked a human worker's progress through a dashboard assembly task and provided the person with the right materials for assembly at the right time.

TABLE IV: Missing segments (%) by each algorithm across three datasets

	Auto-DA	UTKinect	Static-Reach
FOSAPT	5.8	0.0	4.4
CPD	35.0	2.1	3.8
TSC	72.0	17.5	16.0

References

[1] L. Riek, "Healthcare robotics," *Communications of the ACM*, 2017.
[2] M. J. Matarić, "Socially assistive robotics: Human augmentation versus automation," *Science Robotics*, 2017.
[3] V. V. Unhelkar, H. C. Siu, and J. A. Shah, "Comparative performance of human and mobile robotic assistants in collaborative fetch-and-deliver tasks," in *HRI*, 2014.
[4] S. Lemaignan, M. Warnier, E. A. Sisbot, A. Clodic, and R. Alami, "Artificial cognition for social human–robot interaction: An implementation," *Artificial Intelligence*, 2017.
[5] T. Iqbal and L. D. Riek, "A Method for Automatic Detection of Psychomotor Entrainment," *IEEE TAC*, 2016.
[6] N. Sebanz, H. Bekkering, and G. Knoblich, "Joint action: bodies and minds moving together," *Trends Cogn Sci*, 2006.
[7] S. Nikolaidis, D. Hsu, and S. Srinivasa, "Human-robot mutual adaptation in collaborative tasks: Models and experiments," *IJRR*, 2017.
[8] T. Iqbal, S. Rack, and L. D. Riek, "Movement coordination in human-robot teams: A dynamical systems approach," *IEEE TRO*, 2016.
[9] P. A. Lasota, T. Fong, and J. A. Shah, "A survey of methods for safe human-robot interaction," *Foundations and Trends in Robotics*, 2017.
[10] S. Krishnan, A. Garg, S. Patil, C. Lea, G. Hager, P. Abbeel, and K. Goldberg, "Transition state clustering: Unsupervised surgical trajectory segmentation for robot learning," *IJRR*, 2017.
[11] C. Wu et al., "Watch-n-patch: unsupervised learning of actions and relations," *PAMI*, 2018.
[12] E. B. Fox, M. C. Hughes, E. B. Sudderth, M. I. Jordan, *et al.*, "Joint modeling of multiple time series via the beta process with application to motion capture segmentation," *Annals Applied Statistics*, 2014.
[13] P. Bojanowski, R. Lajugie, F. Bach, I. Laptev, J. Ponce, C. Schmid, and J. Sivic, "Weakly supervised action labeling in videos under ordering constraints," in *ECCV*, 2014.
[14] P. Fearnhead and Z. Liu, "Efficient bayesian analysis of multiple changepoint models with dependence across segments," *Statistics and Computing*, 2011.
[15] L. Xia, C.-C. Chen, and J. K. Aggarwal, "View invariant human action recognition using histograms of 3d joints," in *CVPRW*, 2012.
[16] P. A. Lasota and J. A. Shah, "Analyzing the effects of human-aware motion planning on close-proximity human–robot collaboration," *Human factors*, 2015.
[17] C. Lea, M. D. Flynn, R. Vidal, A. Reiter, and G. D. Hager, "Temporal convolutional networks for action segmentation and detection," in *International Conference on Computer Vision (ICCV)*, 2017.
[18] H. Kuehne, J. Gall, and T. Serre, "An end-to-end generative framework for video segmentation and recognition," in *WACV*, 2016.
[19] F. Zhou, F. De la Torre, and J. K. Hodgins, "Hierarchical aligned cluster analysis for temporal clustering of human motion," *PAMI*, 2013.
[20] S. Niekum, S. Osentoski, *et al.*, "Learning and generalization of complex tasks from unstructured demonstrations," in *IROS*, 2012.
[21] P. Fearnhead and Z. Liu, "On-line inference for multiple changepoint problems," *Journal of the Royal Statistical Society: Series B*, 2007.
[22] G. Konidaris, S. Kuindersma, R. Grupen, and A. Barto, "Robot learning from demonstration by constructing skill trees," *IJRR*, 2012.
[23] B. Hayes and J. A. Shah, "Interpretable models for fast activity recognition and anomaly explanation during collaborative robotics tasks," in *ICRA*, 2017.
[24] B. Hayes and B. Scassellati, "Autonomously constructing hierarchical

task networks for planning and human-robot collaboration," in *ICRA*, 2016.