Bayesian Estimator for Partial Trajectory Alignment

Przemyslaw A. Lasota and Julie A. Shah
Computer Science and Artificial Intelligence Laboratory (CSAIL)
Massachusetts Institute of Technology, Cambridge, MA, USA

***Abstract*—The problem of temporal alignment of time series is common across many fields of study. Within the domain of robotics, human motion trajectories are one type of time series that is often utilized for recognition and prediction of human intent. In these applications, online temporal alignment of partial trajectories to a full representative trajectory is of particular interest, as it is desirable to make accurate intent prediction decisions early in a motion in order to enable proactive robot behavior. This is a particularly difficult problem, however, due to the potential for overlapping trajectory regions and temporary stops, both of which can degrade the performance of existing alignment techniques. Furthermore, it is desirable to not only provide the most likely alignment but also characterize the uncertainty around it, which current methods are unable to accomplish. To address these difficulties and drawbacks, we present BEST-PTA, a framework that combines optimization, supervised learning, and unsupervised learning components in order to build a Bayesian model that outputs distributions over likely correspondence points based on observed partial trajectory data. Through an evaluation incorporating multiple datasets, we show that BEST-PTA outperforms previous alignment techniques; furthermore, we demonstrate that this improvement can significantly boost human motion prediction performance and discuss the implications of these results on improving the quality of human-robot interaction.**

I. Introduction

Temporal alignment of time series data is a problem under investigation across a wide variety of domains and tasks, including phenome alignment in speech processing [7], subsequence matching in medical data [21], and audio alignment in musical performance [3]. Within robotics-related applications — specifically, the field of human-robot interaction — human motion trajectories represent a type of time series that is of particular interest. Temporal alignment of such trajectories is a common problem in activity recognition [5, 20], action and motion prediction [11, 14, 4], and gesture recognition [1, 18]. These techniques allow robots to understand and predict human intentions, which can be leveraged to improve the safety and efficiency of human-robot interaction.

Within the aforementioned fields and applications, online partial trajectory alignment represents a particularly challenging problem. Distinct from temporally aligning two complete time series, online partial trajectory alignment involves observing streaming trajectory data and, as each new position is collected, identifying a suitable alignment between the latest point of the partial trajectory and a point along a reference trajectory for that motion.

Author contact: {plasota, julie_a_shah}@csail.mit.edu. This work was supported in part by a NASA Space Technology Research Fellowship.

To describe the problem more formally, let us define trajectories as matrices $X \in \mathbb{R}^{N \times T}$, where N is the number of tracked degrees of freedom (e.g., the spatial coordinates of a person's hand) and T is the number of time steps. Each column of X, denoted as $\vec{x}_t$, defines a vector of coordinates at a particular time step $t \in 1, \ldots, T$, and the trajectory is sampled at some time interval τ. A subset of a trajectory matrix up to some time t is denoted as $X_{[1,t]}$. A partial trajectory for some action, denoted as X^P, is appended with a newly observed set of coordinates, $\vec{x}^P_t$, at each consecutive time step t. Given a complete representative trajectory for the same action, X^R, the objective of online partial trajectory alignment is to identify a *correspondence point* in the representative trajectory, $\vec{x}^R_{t^*}$, that accurately maps to the currently observed position in the partial trajectory, $\vec{x}^P_t$. In other words, the goal is to find a t^* such that X^P maps to the subset $X^R_{[1,t^*]}$.

Accurate online alignment of partial trajectories is particularly useful for early prediction of motions, actions, or gestures, as only part of a trajectory is available in these situations, precluding the use of techniques designed for full trajectory alignment. As we show in this work, previous methods of online partial time series alignment can perform poorly when applied to trajectory alignment, especially with trajectories containing overlapping regions and temporary stops. The presence of these qualities in many real-world trajectories motivates the development of an online partial time series alignment method specifically geared toward such trajectories. Furthermore, while prior approaches only output a single alignment estimate, it is desirable to provide a distribution over correspondence points such that a given method using the alignment can potentially reason on the uncertainty over correspondence.

To address the drawbacks highlighted above, we introduce BEST-PTA (Bayesian ESTimator for Partial Trajectory Alignment), a Bayesian estimation framework composed of a mixture of optimization, supervised learning, and unsupervised learning components that are trained and synthesized based on a given set of example trajectories. We specifically address the problems of trajectory overlap and temporary stops via segmentation and stop segment detection, and compute and leverage correspondence point priors and distance-based likelihoods to produce the correspondence distribution. We evaluate our framework against three partial time series alignment baselines, and show that our approach outperforms these baselines across two human motion datasets. Furthermore, we demonstrate the benefit of this improved alignment in the context of human motion prediction.

II. Related Work

The problem of partial trajectory alignment is closely related to the general concept of time series alignment, wherein two complete time series are given and the objective is to find a time index mapping between the points in these time series. One common method, originally developed for speech recognition, is dynamic time warping (DTW) [15], which uses dynamic programming to identify an optimal time alignment that minimizes the distance between the aligned signals. Several variants of DTW have emerged since its introduction, including FastDTW [16], which provides an approximate solution to DTW with linear time and space complexity, and derivative dynamic time warping (DDTW) [8], which aligns signals with respect to their first derivative in order to achieve more-intuitive alignments between signals in which a feature (e.g., a peak) is more pronounced in one signal over the other. More recently, Zhou and De la Torre [22] developed generalized time warping (GTW), a technique specifically geared toward the alignment of human motion obtained from multi-modal sources (such as video, motion capture, and accelerometer data).

While these methods provide various ways of aligning full trajectories, the problem of online partial trajectory alignment is unique, and the above approaches are generally not well suited for this task. Consequently, researchers have developed other techniques to address temporal misalignment with partial trajectory observations. The approach used by Hayes and Shah [5], for example, does not search for an explicit temporal alignment, but instead divides trajectories into overlapping temporal segments over which to train independent Gaussian mixture models for activity recognition, and then uses a modified form of max pooling over a time window. Dong and Williams [4], on the other hand, identified the best alignment between the current state and trained probabilistic flow tubes (PFTs) based on the distance between the current state and the PFT, as well as the elapsed time in the partial execution. Lasota and Shah [11] also used a metric of alignment based on distance and time, but their method involved searching for the nearest point in a temporal moving window.

Besides the above techniques built into prediction and recognition frameworks, researchers have also developed techniques specifically for partial time series alignment. In the work by Dixon [3] (which Pérez-D'Arpino and Shah used for partial trajectory alignment in later research [14]), the authors developed an online variant of DTW (O-DTW) in which the entries of the time warp matrix are computed iteratively as new data is received, and the direction of the search is guided by the location of the current best alignment value. The number of computed values at each iteration is bounded, resulting in constant time complexity. The path cost calculations incorporate the standard DTW formulation but are restricted to the previously computed values, allowing for online computation of an optimal alignment. Building upon this work, Macrae and Dixon [13] created windowed time warping, which, by dividing the overall alignment problem into sub-alignments on smaller, overlapping regions, achieves accuracy comparable to O-DTW while improving run time.

Latecki et al. [12] presented another approach, optimal sequence bijection (OSB), which focused on creating a one-to-one mapping (bijection) between sequences and effectively handling trajectories containing outliers. The resilience to outliers is implemented by allowing for the skipping of points in both the partial and full trajectories during alignment, and by incorporating a *jump cost* that penalizes such skips proportionally to the number of points skipped in a row. To achieve linear time complexity, this approach includes a warping window size parameter and limits the maximum number of points the algorithm can skip in a row.

While the above methods achieved success in partial time series alignment, human motion trajectories pose unique challenges that can adversely affect these methods' performance. Specifically, trajectory overlap and temporary stops can be problematic, as the above methods' distance metrics are generally described in terms of the Euclidean norm. When combined with variability in execution speed, it becomes difficult to set the above methods' parameters such that they can account for all of these qualities and still produce satisfactory results. Furthermore, as mentioned in Section I, it is desirable to output a distribution over possible correspondence points, but the above techniques do not provide such a distribution.

III. Method

To address the drawbacks of previous partial trajectory alignment methods, we introduce BEST-PTA, a Bayesian estimation framework that combines optimization, supervised learning, and unsupervised learning approaches. Given a set of training trajectories, the first component of the system computes a spatially accurate representative trajectory, X^R, through an iterative process based on DTW. The next component then analyzes the velocity profile of this trajectory to discover stop segments. These segments and X^R are then fed into an optimization-based segmentation process that divides the representative trajectory such that overlapping trajectory regions are assigned to separate segments. Next, the framework learns a Random Forest classifier (RF) [6] over the computed segments, builds a library of segment-specific prior distributions over correspondence points, and learns a distance-based likelihood. During execution of a new trajectory, the framework combines the current segment probabilities and durations from a state machine, the learned prior distributions, and the likelihood function to generate the distribution over correspondence points. We describe the above components in detail in the following sections.

A. Representative Trajectory Computation

The first step of the framework is to define a representative trajectory, X^R, based on the training set, $\mathcal{D}^T$. In order to avoid smoothing out trajectory features by simply averaging over the trajectories in the set, we developed an iterative algorithm based on DTW. In this process, the algorithm first removes a trajectory at random from $\mathcal{D}^T$ and assigns it as the current

estimate of the representative trajectory, denoted as $\hat{X}^R$. Then, at each successive iteration, the algorithm randomly selects another trajectory from $\mathcal{D}^T$ without replacement, computes an alignment to $\hat{X}^R$ with DTW, takes a weighted average of the aligned points (and their time stamps) with weights assigned in proportion to the number of trajectories that $\hat{X}^R$ is already composed of, and resamples the trajectory back to the sampling rate, τ. The resulting trajectory becomes the new $\hat{X}^R$, and the process repeats until all trajectories in $\mathcal{D}^T$ are incorporated. Figure 1 depicts an example of how a representative trajectory computed using our method maintains features that are lost when simply taking a mean over the trajectories in the training set.

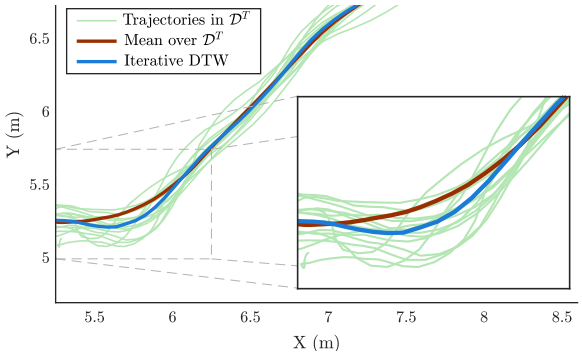

Fig. 1: A comparison between computing X^R by taking the mean over $\mathcal{D}^T$ and by using our iterative DTW method. Note how the former "smoothes out" the corner and results in a representative trajectory that is spatially inconsistent.

B. Stop Segment Discovery

One aspect of motion trajectories that can cause alignment complications is the presence of stops. This can be observed, for example, in the trajectories formed by people reaching for objects or walking to a location in order to perform a brief task before continuing their motion. Within a trajectory, a stop will result in a cluster of points, all within a very small distance of each other. As Euclidean distance is often used as an alignment metric, this can lead to errors. To address this problem, the second component of our framework incorporates a speed estimate in order to identify the dominant movement modes and finds potential "stop segments," which allows other components of our method to include special consideration for these unique regions.

The first step of stop segment discovery is to compute the speed at each point in the trajectory, denoted as $\vec{Z}$. This is performed by finding a cubic spline fit to the trajectory, taking a numerical derivative to determine velocities, and computing the norm at each time step. In order to find the dominant speed modes within X^R, the system utilizes a Dirichlet process Gaussian mixture model (DPGMM) fit to the derived speeds. This model is particularly suitable because human motion characteristics tend to be Gaussian-distributed, and no assumptions must be made about the number of speed modes. Importantly, however, while movement modes will tend to be Gaussian-distributed, the stop mode will be a half-Gaussian with a peak near zero. In order to create a full-Gaussian stop mode, the system utilizes an extended vector, $\vec{Z}^E = [-\vec{Z}, \vec{Z}]$. Figure 2 depicts a sample histogram of $\vec{Z}^E$ of a trajectory and the corresponding DPGMM fit with a stop mode and two movement modes.

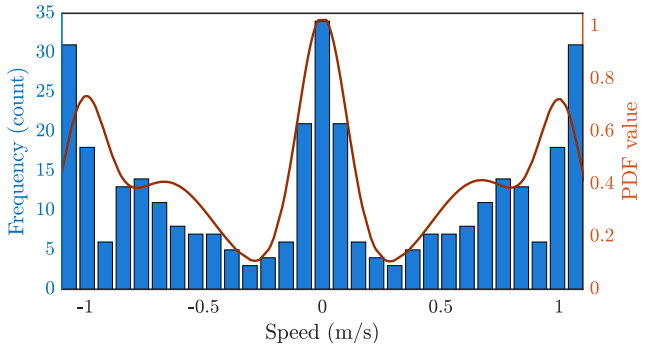

Fig. 2: A histogram of $\vec{Z}^E$ (blue) and the DPGMM fit (red). The DPGMM shows a dominant speed mode at 0 (a "stop mode") and two movement modes at ± 0.7 m/s and ± 1.0 m/s.

For every mixture component $i \in 1, \ldots, n_c$ of the DPGMM, the system identifies it as a stop mode if it is centered around zero ($|\mu_i| < 1 \frac{cm}{s}$) and its weight is at least 5% of what the value would be if the modes were uniformly distributed ($\pi_i > \frac{0.05}{n_c}$). Once a mixture component i is determined to be a stop mode, it can be used to define a speed threshold, $z^* = \alpha \sigma_i$; the stop segments are then defined as the portions of the trajectory below this threshold. The parameter α is chosen based on the desired confidence of speeds below z^* being generated by the half-Gaussian mode, and was set to 1.1 in our implementation. To account for noise in the speed estimates, the algorithm adjusts stop and motion segments shorter than a set percentage of the trajectory length by merging them with neighboring segments (this percentage was set to 7% in our implementation). Furthermore, as the representative trajectory is derived from many individual trajectories, its speed profile can be significantly smoother than a typical trajectory for that action. To address this, the algorithm adjusts the speed threshold by using alignments to each of the trajectories in $\mathcal{D}^T$, taking the speeds observed within the stop segment regions, and fitting a new half-Gaussian to these speeds to derive a new value of z^*. The final stop segment bounds are denoted as a set of start and end indices $\mathcal{W} = \{[w_1^s, w_1^e], \ldots, [w_{|\mathcal{W}|}^s, w_{|\mathcal{W}|}^e]\}$.

C. Ground Truth Alignment

As mentioned in Section II, many partial trajectory alignment techniques rely upon the Euclidean norm as a distance metric between trajectories, which is problematic for trajectories that contain stop segments. This issue also arises in full trajectory alignment, which is necessary to compute ground truth alignments for the purpose of training and evaluating our approach. As a result, we developed a novel algorithm for full trajectory alignment that utilizes the stop segment detection

component in order to generate more-accurate ground truth alignments between trajectories.

Given a representative trajectory X^R and a target trajectory X^T, the first step is to find the boundaries of the stop segments within X^T using the speed threshold z^* and matching each discovered stop segment to those in X^R. The alignment within these stop segments is defined as a linear temporal spacing rounded to the nearest time step. The motion segments in X^R and X^T are then matched and aligned with DTW. As depicted in Figure 3, our method produces ground truth alignments that more accurately represent the relative progress through the trajectory at each point in time.

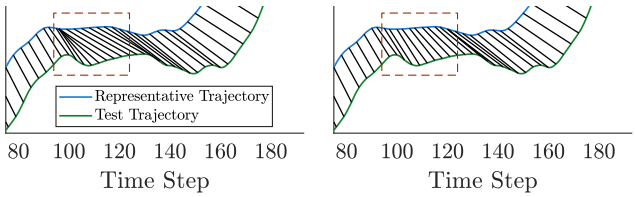

Fig. 3: Part of a ground truth alignment between X^R and a test trajectory X^T found with DTW (left) and our method (right). Note the poor alignment within the stop region (dashed rectangle) when using DTW.

D. Trajectory Segmentation

Trajectories often contain spatially overlapping regions, which can also complicate partial trajectory alignment: similarly to the problem of temporary stops, overlapping regions can lead to small Euclidean distances between large portions of the trajectory, resulting in alignment errors. The objective of the third component of our framework is to alleviate this problem by segmenting the representative trajectory X^R into m parts that include as little within-segment spatial overlap as possible. This is achieved by finding a vector of time indices, $\vec{S}$, that marks the boundaries of the segments (i.e., $\vec{S} = [1, s_2, \ldots, s_m, T]$).

We frame this segmentation as an optimization problem. A pair of points on a trajectory segment can be considered "overlapping" if the Euclidean distance between them is small but the difference between their respective time stamps is large. Therefore, we define a cost function in the range [0,1] for pairs of points at time steps i and j in a trajectory segment X as follows:

$$C(i,j;X) = \left[1 - \tanh\left(\frac{3\|\vec{x}_i - \vec{x}_j\|^2}{\eta_D}\right)\right] \tanh\left(\frac{3|i-j|}{\eta_T}\right) \quad (1)$$

In this equation, η_D and η_T are normalizing factors for the distance and time differences, respectively. The former is set to the mean plus one standard deviation of all pairwise distances in trajectory X, while the latter is set to $\frac{T}{2}$. The constant multiplications are included to shift the saturation of the hyperbolic tangent function to the point where the squared distance and time difference are equal to the respective normalizing factors. We can then define the mean cost of a segment with start and end indices t_s and t_e, respectively, as follows:

$$J(t_s,t_e;X) = \frac{1}{\binom{t_e-t_s}{2}} \sum_{i=t_s}^{t_e-1} \sum_{j=i+1}^{t_e} C(i,j;X) \quad (2)$$

In order to bias the optimization away from creating very short segments, we also introduce a regularization term:

$$G(t_s,t_e;X,m) = 1 - \tanh\left(3m\frac{t_e-t_s}{T}\right) \quad (3)$$

Similarly to Eq. 1, multiplication by a constant is performed so that the hyperbolic tangent function saturates, and thus brings the value of G near 0 when the length of this segment as a percentage of the total trajectory length equals $\frac{1}{m}$. Combining Eqs. 2 and 3, we arrive at the overall cost function for the segmentation:

$$R(\vec{S};X) = \sum_{k=1}^{|\vec{S}|-1} J(s_k,s_{k+1};X) + \lambda G(s_k,s_{k+1};X,m) \quad (4)$$

In the above, λ controls the influence of the regularization term, which we set to 0.75 in our implementation. To identify the optimal segmentation, $\vec{S}^*$, it is necessary to identify the segmentation time index vector, $\vec{S}$, that minimizes Eq. 4 — subject to the constraints that the first and last elements of $\vec{S}$ are 1 and T, respectively, that the elements of $\vec{S}$ are integers, and that they are monotonically increasing, as follows:

$$\begin{gathered} \vec{S}^* = \underset{\vec{S}}{\operatorname{argmin}}\, R(\vec{S};X) \\ s_1 = 1; \quad s_{|\vec{S}|} = T; \quad s_i \in \mathbb{Z}; \quad s_i > s_{i-1} \end{gathered} \quad (5)$$

This formulation makes the segmentation problem an integer nonlinear program (INLP). As the overall objective function $R(\vec{S};X)$ is trajectory-dependent and potentially non-convex, a global optimization method is necessary to calculate $\vec{S}^*$. In our implementation, we elected to use a genetic algorithm (GA) solver with a population size of 50, a crossover fraction of 0.8, and a constraint tolerance of 0.001. In the current implementation, the number of segments, m, needs to be provided by the user.

If stop segments are identified as described in Section III-B, the system first enforces that their bounds, $\mathcal{W}$, are part of $\vec{S}$ and not allowed to change. Next, to prevent any of the free values of $\vec{S}$ from being placed between the start and end indices of any $\mathcal{W}_i$, we incorporated a nonlinear constraint that penalizes such placements proportional to the square of the encroachment:

$$\sum_{i=1}^{|\mathcal{W}|} \sum_{j=1}^{|\vec{S}|} [w_j^s <= s_j <= w_i^e]\{min(s_j - w_i^s, w_i^e - s_j)\}^2 \leq 0 \quad (6)$$

Figure 4 shows examples of segmentations derived by using our optimization-based technique.

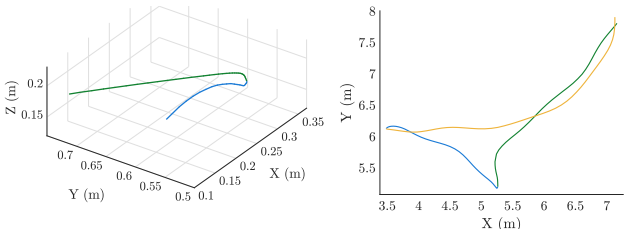

Fig. 4: Segmentation results for a 3D reaching motion (left) and a 2D walking motion (right). Each segment is depicted using a different color.

E. Segment Classification

Once the segmentation of the representative trajectory is complete, a classifier is required that would indicate which segment the head of the partial trajectory is currently in for each new partial trajectory observation $\vec{x}_t^P$. One common classification technique suitable for this task is the Random Forest (RF) [2], as it supports multi-class problems and its output is a discrete distribution over classes. In our case, the feature vector for a particular point $\vec{x}_t$ consists of a concatenation of trajectory coordinates, estimated velocities, and speed: $\vec{f}_t = [\vec{x}_t, \vec{v}_t, \|\vec{v}_t\|] \in \mathbb{R}^{2N+1}$.

To compute velocity estimates for the feature vector, the system removes high-frequency noise using the Savitzky-Golay differentiation filter [17]. The polynomial order and frame size for the Savitzky-Golay filter are hyperparameters that the system learns through the use of a validation set, $\mathcal{D}^V$. After training, given a feature vector corresponding to $\vec{x}_t^P$, the RF classifier outputs a discrete distribution over segments, denoted as $\theta \in \mathbb{R}^m$. This distribution is then used by a segment state machine, which we describe in the following section.

F. Segment State Machine

In order to monitor which segment the head of the current partial trajectory occupies and how long ago each segment transition occurred, we implement a state machine. Let us denote the indices of the segments as $k = 1, \ldots, m$ and the vector of durations since transition to each segment as $\vec{D} = [d_1, \ldots, d_m]$. For each new position of the current partial trajectory, $\vec{x}_t^P$, the state machine takes the current distribution over segments θ from the RF classifier and computes a moving average of these probabilities, denoted as $\hat{\theta} \in \mathbb{R}^m$, over a window of size γ_w.

The state machine also learns temporal duration priors by fitting Gaussian Kernel distributions to the durations for each segment observed in $\mathcal{D}^T$. Let us denote the cumulative distribution function (CDF) of this distribution for segment k as $F(d_k)$. In order to prevent potential overfitting, a uniform prior, computed by taking the product of the maximum value of the probability density function (PDF) and a multiplier γ_u, is added to the kernel distribution over the range of durations where $\epsilon < F(d_k) < 1 - \epsilon$ with $\epsilon = 0.001$.

Let us denote the CDF of the kernel distribution with the added uniform prior as $F^*(d_k)$ and the indices of the current, next, and previous segments as c, n, and p, respectively. During execution, the system determines whether a state transition should occur via a Bayesian formulation. Namely, the state machine transitions forward if $F^*(d_c) \cdot \frac{\hat{\theta}_n}{\hat{\theta}_n + \hat{\theta}_c} > \gamma_n$ and backwards if $[1 - F^*(d_c)] \cdot \frac{\hat{\theta}_p}{\hat{\theta}_p + \hat{\theta}_c} > \gamma_p$. The hyperparameters $\vec{\Gamma} = [\gamma_w, \gamma_n, \gamma_p, \gamma_u]$ are learned with the use of the validation set, $\mathcal{D}^V$.

G. Temporal Correspondence Prior

The next component of the framework constructs the temporal prior of the Bayesian formulation. To build our prior distributions, the system iterates through all trajectories $X^i \in \mathcal{D}^T$ and computes ground truth alignments to X^R for each of them with the approach described in Section III-C. As the system iterates through the training set, it counts, for each segment k and each time step in the range $1, \ldots, (s_{k+1} - s_k)$ within that segment, the number of times each of these time steps corresponded to each time step of the corresponding segment of X^R. This process results in a total of T vectors of such counts. The system then fits Gaussian kernel distributions to each of these vectors, generating smooth probability distributions without any assumptions about the underlying distribution structure. Each of these distributions, denoted as $P_{GK}(k, d_k)$, describes the prior probability of correspondence to the time steps of segment k for a given duration d_k since the transition to this segment occurred. In order to avoid poor Gaussian kernel fits to overly sparse data, these distributions are only defined up to durations d_k for which the count vector had at least 10 values. Let us denote this cutoff duration for segment k as d_k^C.

Next, we use the distributions $P_{GK}(k, d_k)$ to construct discrete priors over the time steps of X^R. First, we recognize that for any given segment k, all probabilities before time step s_k^* are zero. For the time steps of the segment, $s_k^* \leq t^* < s_{k+1}^*$, the system integrates each of the kernel distributions of segment k over intervals of length one centered around these time steps. Keeping in mind that $d_k = t^* - s_k^* + 1$, the discretized kernel distributions over possible values of t^* of X^R given a duration of d_k since transition to segment k and segmentation points $\vec{S}^*$ are then defined as follows:

$$P_{DGK}(t^*; k, d_k, \vec{S}^*) \propto \begin{cases} \int_{(t^* - s_k^* + 1) - 0.5}^{(t^* - s_k^* + 1) + 0.5} P_{GK}(k, t) dt, & s_k^* \leq t^* < s_{k+1}^* \\ 0, & \text{otherwise} \end{cases} \quad (7)$$

In order to account for durations longer than d_k^C, the distribution at d_k^C is taken and shifted forward in time by the difference between d_k^C and d_k. The final definition of the discrete, temporal prior probability over the time steps of X^R given a duration of d_k since transition to segment k and segmentation points $\vec{S}^*$ is then given as follows:

$$P_T(t^*; k, d_k, \vec{S}^*) \propto \begin{cases} P_{DGK}(t^*; k, d_k, \vec{S}^*) & d_k \leq d_k^C \\ P_{DGK}(t^* - (d_k - d_k^C); k, d_k^C, \vec{S}^*) & d_k > d_k^C \end{cases} \quad (8)$$

Similarly to the method used in the temporal priors of the state machine, in order to prevent potential overfitting caused by data sparsity, the system adds a uniform prior to the non-zero values of P_T. The magnitude of this prior is different for stop and motion segments, and is learned with the use of the validation set $\mathcal{D}^V$.

During execution of a trajectory, the system utilizes the learned collection of P_T distributions to formulate our overall prior. The continually incremented values of $\vec{D}$ of the segment state machine represent competing hypotheses of the true value of k and d_k (i.e., whether $\vec{x}_t^P$ is in segment k with duration d_k, or in some prior segment $k-j$ with duration d_{k-j}). In order to provide robust priors, especially near segment transitions, the system computes a weighted average over all P_T up to the current segment proportional to $\hat{\theta}$:

$$\widetilde{P}_T(t^*; \vec{D}, \hat{\theta}, \vec{S}^*) \propto \sum_{k=1}^{c} \hat{\theta}_k P_T(t^*; k, d_k, \vec{S}^*) \tag{9}$$

Finally, in order to bias the prior toward high-probability segments, we weigh the probability at each value of t^* proportionally to the probability of the segment that t^* belongs to. If we define the segment index of time step t in X^R as $k(t)$, then the final form of our temporal prior is:

$$P(t^*; \vec{D}, \hat{\theta}, \vec{S}^*) \propto \hat{\theta}_{k(t^*)} \widetilde{P}_T(t^*; \vec{D}, \hat{\theta}, \vec{S}^*) \tag{10}$$

This final formulation, given the candidate segment durations and probability estimates from the state machine, selects the appropriate learned priors and synthesizes them together in order to provide a discrete prior distribution over all possible values of t^* in X^R.

H. Distance-Based Likelihood

The second key probability distribution of our Bayesian formulation that the system learns during the training process is the likelihood of observing a specific position given the alignment time t^*. The system builds the likelihood model by characterizing the typical distance between aligned points within each of the m segments of X^R. Namely, the system finds the ground truth alignment to each trajectory in $\mathcal{D}^T$ and computes an exponential distribution fit for each segment (i.e., learns the rate parameter λ_k for $k = 1, \ldots, m$) based on vectors of observed distances between pairs of aligned points within each segment. The likelihood of the position $\vec{x}$ given a proposed alignment point t^* is then simply given by:

$$P(\vec{x} \mid t^*; X^R, \vec{S}^*) \propto f(\|\vec{x} - \vec{x}_{t^*}^R\|; \lambda_{k(t^*)}) \tag{11}$$

In this equation, $f(x; \lambda)$ is the PDF of an exponential distribution evaluated at x with rate parameter λ, and $k(t)$ is the segment index of time step t in X^R.

I. Bayesian Correspondence Distribution

Next, we describe the process of using the temporal prior and distance-based likelihood for computing the correspondence distribution over the time steps t^* of X^R given a partial trajectory X^P. As each successive point $\vec{x}_t^P$ is received, the system updates the segment state machine, computing new values for $\hat{\theta}$ and $\vec{D}$. Combining these values and Eqs. 10 and 11, we define the overall correspondence distribution using Bayes' rule:

$$P(t^* \mid \vec{x}_t^P; X^R, \vec{S}^*) \propto P(\vec{x}_t^P \mid t^*; X^R, \vec{S}^*) P(t^*; \vec{D}, \hat{\theta}, \vec{S}^*) \tag{12}$$

This distribution allows one not only to select a high-probability correspondence point t^*, but also to characterize the confidence in this alignment based on the relative probabilities of other candidate alignment points. Based on this formulation, the computational complexity of evaluating the state machine update, likelihood, and prior for each new point $\vec{x}_t^P$ all scale linearly with the length of X^R. Based on this fact, the overall complexity of BEST-PTA is $\mathcal{O}(n)$, which facilitates its use in online applications such as activity recognition or human motion prediction.

IV. Evaluation

In order to evaluate our technique, we compared its alignment performance against three baseline methods on two unique datasets. We also evaluated the benefit of our alignment method on the accuracy of human motion prediction.

A. Datasets

We utilized two distinct datasets to evaluate our method: reaching motions during a tabletop factory task (TF) and walking motions during an automotive assembly task (AA). Each consists of a set of distinct actions, each with a unique trajectory profile — and, therefore, a representative trajectory, X^R. Consequently, we grouped the trajectories of each action within each dataset as separate subsets during training and evaluation. For each such subset, we split the data into a training set, $\mathcal{D}^T$; validation set, $\mathcal{D}^V$; and evaluation set, $\mathcal{D}^E$.

The TF dataset, originally presented by Lasota and Shah [10], consists of trajectories from a collaborative task during which a person placed fasteners at eight distinct locations on a table while a robot applied a sealant to the fasteners. Human reaching motions, consisting of the three-dimensional coordinates of the wrist, were collected using a PhaseSpace motion capture system and resampled to a rate of 50Hz (τ = 0.02s). The set contains data collected from 20 participants who each performed the eight different motions twice. For each participant, we used the motion that corresponded to the "Human-Aware" mode (see [10]). During each placement, participants were required to twist the fastener into place, resulting in a short pause of roughly 2-3s prior to the person retracting his or her hand.

In the AA dataset, first presented by Unhelkar et al [19], participants walked to four different locations on a simulated factory floor to perform logistics tasks. The walking motion data, consisting of the two-dimensional position of the person's head, were collected using a Vicon motion capture system at a rate of 10Hz (τ = 0.1s). In this dataset, two participants performed each of the four motions 20 times. During each trajectory, the participants performed short tasks that resulted in a roughly 2-4s pause during the walking motion. (Three of the actions involved one such pause; one action included two.)

B. Baseline Methods

In order to evaluate our technique, we compare its alignment performance to three baseline methods: O-DTW [3], OSB [12], and the moving window (MW) approach from [11], which were all briefly introduced in Section II. We selected these baselines based on their capability for partial alignment using low-dimensional time series as input (e.g., trajectories instead of time series of images).

The first baseline method, O-DTW, has two key parameters: the number of new warp matrix values to compute at each iteration (denoted as c) and the maximum number of times the search can move in the same direction of the warp matrix ($MaxRunCount$). We selected values for these parameters using the validation set $\mathcal{D}^V$ in the same way as the hyperparameter tuning described in Section III.

For our second baseline, OSB, we utilized the formulation based on work by Latecki et al [12], but with modifications introduced by Koknar-Tezel [9]. The main parameters include the penalty for skipping elements of either trajectory (denoted as C) the maximum number of elements that can be skipped in a row, and a warping window size that limits how far off the diagonal of the warp matrix the search should be limited to. We set the first parameter, C, via our validation set $\mathcal{D}^V$, using the method suggested by the authors of [12]. For the remaining parameters, we used the authors' guidance, and assigned the same value to each of them. To select that value, we used the same hyperparameter tuning method as that used for O-DTW.

When applied to our problem, as OSB attempts to align a given partial trajectory to any part of the full trajectory, it can sometimes identify a similar subsequence in the wrong section of the trajectory, especially for very short partial sequences. To avoid this problem (and make the comparison fair) we introduced an additional constraint to OSB that forced it to match the first elements of X^R and X^P to each other.

Finally, for the MW baseline, the only parameter was the size of the window over which to conduct the search for the nearest point in X^R. Specifically, we were looking for the number of time steps, w, such that the search was limited to $X^R_{[t-w,t+w]}$. Once again, we trained this parameter in the same manner used to train the other baselines.

C. Evaluation Methodology

For each of the two datasets, we performed leave-one-out cross-validation (LOOCV) to compare the performance of our framework to the baselines. For each iteration of the cross-validation, we took one trajectory as $\mathcal{D}^E$ and performed a random 70%/30% split of the remaining trajectories for use in $\mathcal{D}^T$ and $\mathcal{D}^V$, respectively. We defined our error metric as the time difference (in seconds) between the proposed and ground truth partial trajectory alignment times t^* for each time step t of the evaluation trajectory.

In order to assess the impact of the presence of stop segments on alignment performance, we performed the analysis on both the original trajectories as well as modified ones with stop segments removed for each action of each dataset. Below, we refer to these modified datasets by adding a subscript "NS" for "No Stop".

Dataset	O-DTW	OSB	MW	BEST-PTA	Friedman
TF	0.50±0.23	0.33±0.23	0.35±0.21	**0.28±0.21**	χ^2=108.68
TF_{NS}	0.10±0.05	0.06±0.04	0.04±0.04	**0.02±0.02**	χ^2=339.06
AA	0.28±0.13	0.19±0.15	0.21±0.14	**0.09±0.09**	χ^2=127.06
AA_{NS}	0.15±0.07	0.07±0.05	0.05±0.06	**0.02±0.01**	χ^2=180.7

TABLE I: Mean partial trajectory alignment errors (± standard deviation) in seconds on the original and "No Stop" versions of the TF and AA datasets.

D. Impact of Alignment on Accuracy of Motion Prediction

The second part of the evaluation involved applying our partial trajectory alignment framework to human motion prediction with the Multiple-Predictor System (MPS) [11]. We compared the prediction performance of the MPS when using our technique versus each of the baseline methods in order to investigate the impact of the quality of partial trajectory alignment on the accuracy of human motion prediction. We performed a LOOCV using the same data splits and trained aligners as in the first evaluation. Specifically, for each iteration of the cross-validation, we set the partial trajectory aligner of the MPS to each of the previously-trained aligners, trained the MPS using the same training and validation sets, and then computed the mean prediction error of the MPS on the evaluation trajectory. As the trajectories of the TF dataset are quite short in duration, we utilize the AA dataset instead, and use the MPS to make predictions at time horizons from 0.1s to 6s.

V. Results and Discussion

The results of the first evaluation, which compares the average partial trajectory time alignment errors for each test trajectory using BEST-PTA compared to that of the three baselines, indicate that our method leads to superior alignment for both variations of both datasets. To assess statistical significance of our results, we applied the Friedman test to determine the main effect of the alignment method and the Wilcoxon signed rank test for pairwise comparisons. For the original TF dataset, BEST-PTA reduced the mean alignment error by 44.0%, 15.1%, and 20.4% when compared to O-DTW, OSB, and MW, respectively. The reduction in error for the AA dataset was even more significant, with BEST-PTA outperforming the baselines by 69.4%, 55.6%, and 59.0%. The results were similar for the modified datasets (where stop segments were removed) as well. For TF_{NS}, our approach reduced the mean alignment error by 80.5%, 66.6%, and 46.6%, while for the AA_{NS} dataset, the error was reduced by 85.5%, 68.6%, and 59.2%, when compared to O-DTW, OSB, and MW, respectively. These results were all statistically significant ($p < 0.001$ for all tests), and are summarized in Table I.

The mean alignment error results provide strong evidence for BEST-PTA being an effective partial trajectory alignment

method for trajectories containing stop segments and overlapping regions. As BEST-PTA outperformed the baselines for both datasets, these results also indicate that our approach is suitable for alignment of both ambulatory and manipulative motion trajectories, which amplifies its usability within human-robot interaction contexts.

While the differences in magnitudes of the errors are fairly small, by utilizing a Bayesian formulation, a key benefit of our method is that it reduces the occurrence of large and persistent alignment errors. Figure 5 shows the percentage of the evaluated trajectories from the AA dataset in which for at least 5% of the trajectory's duration the error was above the given range of thresholds. From this figure, it is apparent that BEST-PTA results in fewer instances of large errors. Note, for example, that alignment errors of at least 1.5s occurred in only 3% of trajectories when using BEST-PTA, while they occurred nearly 20% of the time for MW and OSB.

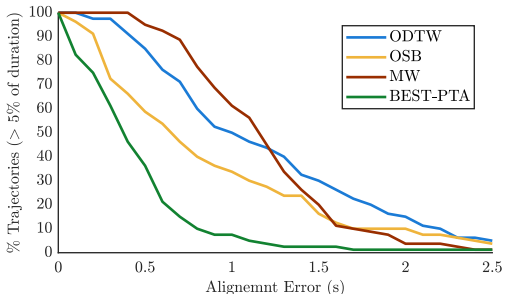

Fig. 5: Percentage of trajectories in the AA dataset for which the alignment error was above the given thresholds for at least 5% of the trajectory.

The fact that our approach leads to significantly fewer large errors has potentially significant implications for the application of partial trajectory alignment to human-robot interaction applications such as activity recognition or motion prediction. In the case of human motion prediction, for example, reducing the number of large alignment errors is likely to result in fewer instances of significant prediction errors.

This idea is supported by the results of the second part of our evaluation. When using BEST-PTA for partial trajectory alignment, the mean prediction errors of the MPS were reduced by 28.0%, 17.7%, and 19.9% when compared to the results when using O-DTW, OSB, and MW, respectively. Once again, a Friedman test and pairwise comparisons with the Wilcoxon signed ranks test showed these results to be statistically significant ($p < 0.001$ for all tests). The mean prediction errors when using BEST-PTA, O-DTW, OSB, and MW were 0.24m, 0.33m, 0.29m, and 0.30m, respectively.

Due to BEST-PTA's quality of having fewer large alignment errors, large prediction errors were also reduced when it was used with the MPS. This trend is displayed in Figure 6, which depicts the percentage of trajectories in the AA dataset for which the error in human motion prediction (at time horizons of 1.5s, 3.1s, and 4.5s) was greater than the given thresholds for at least 5% of the trajectory. From this figure, we can see that BEST-PTA indeed allows the MPS to make fewer large errors. For example, at a prediction time horizon of 3.1s, errors of at least 1.3m appeared in 8.75% of the trajectories when using BEST-PTA, in nearly 30% of the trajectories when using OSB, and in as much as 45% of the trajectories for O-DTW and MW. If a robot's planner is using these predictions to make decisions about its own motions in a shared human-robot workspace, large prediction errors can cause the robot to take inefficient or unsafe actions. Consequently, utilizing BEST-PTA as part of a prediction framework can lead to improvements in safety and efficiency of human-robot interaction.

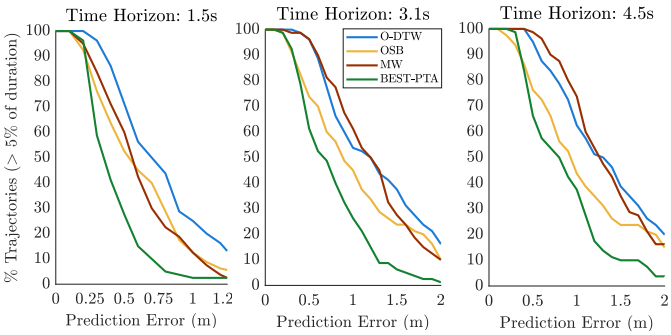

Fig. 6: Percentage of trajectories in the AA dataset for which the MPS error was above the given thresholds for at least 5% of the trajectory for time horizons of 1.5s, 3.1s, and 4.5s.

VI. Conclusion and Future Work

In this work, we introduce BEST-PTA, a Bayesian estimator for partial trajectory alignment specifically designed to accommodate the presence of temporary stops and overlapping trajectory segments. Due to its linear time complexity and the ability to provide distributions over candidate alignment points, our method is suitable for online applications such as activity recognition, action and motion prediction, and gesture recognition. We evaluate our technique against state-of-the-art baselines, and demonstrate that BEST-PTA outperforms these baselines for both ambulatory and manipulative human motions. Furthermore, we show the utility of our method for human motion prediction by integrating it as part of a previously developed prediction framework, and demonstrate that it leads to lower prediction errors.

In the future, we plan to analyze the impact of BEST-PTA on the quality of human-robot interaction by investigating how the improved prediction performance of the MPS due to using BEST-PTA translates to benefits in interaction and co-navigation. We would also like to improve BEST-PTA by incorporating a more robust method of tracking segment transitions, using Hidden Semi-Markov Models, for example, in order to handle trajectories that differ significantly from those observed during training. Lastly, while the majority of our framework is unsupervised, the number of segments m must be provided by the user. We therefore plan to implement a non-parametric approach for discovering the optimal value of m for a given trajectory.

References

[1] Tarik Arici, Sait Celebi, Ali S Aydin, and Talha T Temiz. Robust gesture recognition using feature pre-processing and weighted dynamic time warping. *Multimedia Tools and Applications*, 72(3):3045–3062, 2014. doi: 10.1007/s11042-013-1591-9.

[2] Leo Breiman. Random Forests. *Mach. Learn.*, 45(1): 5–32, October 2001. ISSN 0885-6125. doi: 10.1023/A:1010933404324.

[3] Simon Dixon. Live tracking of musical performances using on-line time warping. In *International Conference on Digital Audio Effects*, pages 92–97. Citeseer, 2005.

[4] Shuonan Dong and Brian Williams. Learning and Recognition of Hybrid Manipulation Motions in Variable Environments Using Probabilistic Flow Tubes. *International Journal of Social Robotics*, 4(4):357–368, Nov 2012. ISSN 1875-4805. doi: 10.1007/s12369-012-0155-x.

[5] Bradley Hayes and Julie A Shah. Interpretable models for fast activity recognition and anomaly explanation during collaborative robotics tasks. In *International Conference on Robotics and Automation (ICRA)*, pages 6586–6593. IEEE, 2017. doi: 10.1109/ICRA.2017.7989778.

[6] Tin Kam Ho. Random decision forests. In *International Conference on Document Analysis and Recognition*, volume 1, pages 278–282. IEEE, 1995.

[7] John-Paul Hosom. Speaker-independent phoneme alignment using transition-dependent states. *Speech Communication*, 51(4):352–368, 2009. doi: 10.1016/j.specom.2008.11.003.

[8] Eamonn J Keogh and Michael J Pazzani. Derivative dynamic time warping. In *International Conference on Data Mining*, pages 1–11. SIAM, 2001. doi: 10.1137/1.9781611972719.1.

[9] Suzan Koknar-Tezel. *Optimal Subsequence Bijection and Classification of Imbalanced Data Sets*. PhD thesis, Philadelphia, PA, USA, 2011. AAI3440089.

[10] Przemyslaw A. Lasota and Julie A. Shah. Analyzing the Effects of Human-Aware Motion Planning on Close-Proximity Human-Robot Collaboration. *Human Factors*, 57(1):21–33, January 2015. ISSN 0018-7208.

[11] Przemyslaw A. Lasota and Julie A. Shah. A multiple-predictor approach to human motion prediction. In *International Conference on Robotics and Automation (ICRA)*, pages 2300–2307, May 2017. doi: 10.1109/ICRA.2017.7989265.

[12] Longin Jan Latecki, Qiang Wang, Suzan Koknar-Tezel, and Vasileios Megalooikonomou. Optimal subsequence bijection. In *ICDM International Conference on Data Mining*, pages 565–570. IEEE, 2007. doi: 10.1109/ICDM.2007.47.

[13] Robert Macrae and Simon Dixon. Accurate Real-time Windowed Time Warping. In *International Society for Music Information Retrieval Conference (ISMIR)*, pages 423–428, 2010.

[14] Claudia Pérez-D'Arpino and Julie A Shah. Fast Target Prediction of Human Reaching Motion for Cooperative Human-Robot Manipulation Tasks using Time Series Classification. In *International Conference on Robotics and Automation (ICRA)*, 2015. doi: 10.1109/ICRA.2015.7140066.

[15] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 26(1):43–49, Feb 1978. ISSN 0096-3518. doi: 10.1109/TASSP.1978.1163055.

[16] Stan Salvador and Philip Chan. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis*, 11(5):561–580, 2007.

[17] Abraham Savitzky and Marcel JE Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical chemistry*, 36(8):1627–1639, 1964.

[18] Wenjun Tan, Chengdong Wu, Shuying Zhao, and Jiang Li. Dynamic hand gesture recognition using motion trajectories and key frames. *International Conference on Advanced Computer Control*, 3:163–167, 2010. doi: 10.1109/ICACC.2010.5486760.

[19] Vaibhav V Unhelkar, Przemyslaw A Lasota, Quirin Tyroller, Rares-Darius Buhai, Laurie Marceau, Barbara Deml, and Julie A Shah. Human-aware robotic assistant for collaborative assembly: Integrating human motion prediction with planning in time. *IEEE Robotics and Automation Letters*, 3(3):2394–2401, 2018. doi: 0.1109/LRA.2018.2812906.

[20] Raviteja Vemulapalli, Felipe Arrate, and Rama Chellappa. Human action recognition by representing 3d skeletons as points in a lie group. In *IEEE conference on computer vision and pattern recognition*, pages 588–595, 2014. doi: 10.1109/CVPR.2014.82.

[21] Huanmei Wu, Betty Salzberg, Gregory C Sharp, Steve B Jiang, Hiroki Shirato, and David Kaeli. Subsequence matching on structured time series data. In *SIGMOD International Conference on Management of Data*, pages 682–693. ACM, 2005. doi: 10.1145/1066157.1066235.

[22] F. Zhou and F. De la Torre. Generalized time warping for multi-modal alignment of human motion. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1282–1289, June 2012. doi: 10.1109/CVPR.2012.6247812.