Model-Driven Matching and Segmentation of Trajectories [*]

Swaminathan Sankararaman
Akamai Technologies

Pankaj K. Agarwal
Duke University

Thomas Mølhave
Duke University

Jiangwei Pan
Duke University

Arnold P. Boedihardjo
U.S. Army Corps of Engineers

ABSTRACT

A fundamental problem in analyzing trajectory data is to identify common patterns between pairs or among groups of trajectories. In this paper, we consider the problem of matching similar portions between a pair of trajectories, each observed as a sequence of points sampled from it. We present new measures of trajectory similarity — both local and global — between a pair of trajectories to distinguish between similar and dissimilar portions. We then use this model to perform segmentation of a set of trajectories into *fragments*, contiguous portions of trajectories shared by many of them.

Our model for similarity is robust under noise and sampling rate variations. The model also yields a score which can be used to rank multiple pairs of trajectories according to similarity, e.g. in clustering applications. We present quadratic time algorithms to compute the similarity between trajectory pairs under our measures together with algorithms to identify fragments in a large set of trajectories efficiently using the similarity model.

Finally, we present an extensive experimental study evaluating the effectiveness of our approach on real datasets, comparing it with earlier approaches. Our experiments show that our model for similarity is highly accurate in distinguishing similar and dissimilar portions as compared to earlier methods even with sparse sampling. Further, our segmentation algorithm is able to identify a small set of fragments capturing the common parts of trajectories in the dataset.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*data mining*; F.2.2 [**Analysis of Algorithms and Problem Complexity**]: Nonnumerical Algorithms and Problems—*computations on discrete structures*

General Terms

Algorithms, Experimentation, Performance

[*]This work was done while Swaminathan Sankararaman was at Duke University. Work on this paper is supported by NSF under grants CCF-09-40671, CCF-10-12254, and CCF-11-61359, by ARO grants W911NF-07-1-0376 and W911NF-08-1-0452, and by an ERDC contract W9132V-11-C-0003.

(c) 2013 Association for Computing Machinery. ACM acknowledges that this contribution was authored or co-authored by an employee, contractor or affiliate of the United States government. As such, the United States Government retains a nonexclusive, royalty-free right to publish or reproduce this article, or to allow others to do so, for Government purposes only.
SIGSPATIAL'13 , November 05 - 08 2013, Orlando, FL, USA
Copyright 2013 ACM 978-1-4503-2521-9/13/11 ...$15.00
http://dx.doi.org/10.1145/2525314.2525360.

Keywords

GPS trajectories, trajectory matching, trajectory segmentation

1. INTRODUCTION

Trajectories are functions from a time domain—an interval on the real line—to $\mathbb{R}^d$ with $d > 1$, and are observed as sequences of points sampled from them. They arise in the description of any system that evolves over time and are being recorded or inferred from hundreds of millions of sensors nowadays, from traffic monitoring systems [19] and recordings of GPS sensors on cell phones [17] to cameras in surveillance systems and smart phones, helmets of soldiers in the field [7], medical devices, and scientific experiments and simulations such as molecular dynamic simulations [13].

A central problem in analyzing trajectory data is to identify common patterns between pairs or among a group of trajectories. Besides being interesting in its own right, this problem often lies at the core of classifying, clustering, and computing mean trajectories. Motivated by this goal, we study two problems in this paper. The first problem is to develop a model and algorithms for matching similar portions between two trajectories that is effective despite issues present in the data due to real-world constraints. This problem is important both by itself and as a building block for clustering trajectories. The second problem is to segment a large set of trajectories into *fragments*, contiguous portions shared by many trajectories. Such a segmentation is often key to identifying the characteristics of the dataset. For instance, in GPS traces, fragments may correspond to important road sections.

Problem statement. We first study the problem of matching similar portions between two trajectories. If we are given two trajectories γ_1, γ_2, then their similar portions can be defined by reparameterizing them using two functions $\alpha : [0,1] \to \mathbb{R}^d$ and $\beta : [0,1] \to \mathbb{R}^d$ and identifying the sub-intervals of $[0,1]$ over which the two of them are similar. However, we only have finite sample points from each trajectory with additional noise. More precisely, let $P = \langle p_1, \dots, p_m \rangle$ and $Q = \langle q_1, \dots, q_n \rangle$ be two sequences of points in $\mathbb{R}^d$, sampled from γ_1 and γ_2 with added noise. We match similar portions between γ_1 and γ_2 by computing *correspondences* between sample points in P and Q.

Such a set of correspondences must be able to match similar portions in the presence of noise/outliers, even if the sampling rates of the two trajectories are different, and the trajectories are partially observed (i.e., portions of trajectories are missing). For any given set of correspondences between two trajectories, it is also advantageous to provide a score indicating their degree of similarity. Such a score is not only useful in identifying a good set of correspondences but also a way to rank multiple pairs of trajectories according to similarity, e.g., in clustering applications.

Our objective is to define an appropriate model for correspondences together with a scoring function to capture the above requirements and to compute correspondences between two trajectories with the optimal score efficiently. We are also interested in identifying the most similar contiguous sub-trajectories of two given trajectories, i.e., computing *local* similarity between trajectories.

Next, we study the problem of segmenting a collection of trajectories into a small set of *fragments*, using which large portions of all trajectories may be described compactly, i.e., these fragments capture the characteristics of the dataset. Each fragment represents a contiguous portion of a trajectory and the same fragment on two trajectories corresponds to a sub-trajectory common to both of them. Such a form of compression or abstraction will be useful for detecting patterns of behavior more efficiently and also for efficient clustering of trajectories. The objective is to represent large portions of trajectories with as few fragments as possible — there is a tradeoff between the number of fragments used and the number of trajectories sharing each fragment.

Related work. There is extensive work on computing similarity between two objects (or rather points sampled from the objects) in many fields, including computational geometry, computer vision, computer graphics, and statistical learning. A commonly used measure of similarity is the so-called Fréchet distance [2]. Informally, consider a person and a dog connected by a leash, each walking along a curve from its starting point to its end point. Both are allowed to control their speed, but they cannot backtrack. The Fréchet distance between the two curves is the minimal length of a leash that is sufficient for traversing both curves in this manner. The Fréchet distance between two polygonal curves with m and n vertices respectively can be computed in $O(mn \log(m+n))$ time [2]. If we only have samples of points on each trajectory, then a simpler variant, called *discrete Fréchet distance*, is used, where the dog and its owner jump from one sample point to the next without backtracking.The discrete Fréchet distance can be computed in $O(mn)$ time by a straightforward dynamic programming algorithm. Recently Agarwal *et al.* [1] have presented a sub-quadratic time algorithm for the same.

For a given pair of trajectories, there may exist a large number of possible matchings which yield the optimal Fréchet distance. Hence, the correspondences provided by any matching is not necessarily a good indicator of trajectory similarity; see Fig. 1(a). To circumvent this issue, instead of minimizing the length of the leash required (which is equivalent to finding a matching where the maximum distance between matched points is minimized), we may try to minimize the average distance. This falls under the popular *dynamic time warping* framework, which was originally developed for matching speech signals in speech recognition [14]. It matches two similar trajectories effectively even if the sampling rates are different. Since DTW tries to match all points, the results are not meaningful when significant deviations occur between the two trajectories; see Fig. 1(b). It is even more difficult to distinguish actual deviations from outliers caused by measurement errors. Approaches which aim to be robust to outlier include using the longest common sub-sequence [18], an adaptation of the edit-distance measure [6] for sequences, where trajectory points are designated a match if they are closer than a threshold distance, and the edit-distance with real penalties measure [5]. However, these approaches do not distinguish outliers from actual deviations.

On the other hand, there is a rich body of literature on pairwise sequence alignment in computational biology, where the goal is to identify similar portions between two DNA or protein sequences even in the presence of significant dissimilar portions; see [8, cf. Chapter 2]. These methods have also been extended to aligning two polygonal curves such as protein backbones and may be easily applied to the matching of trajectories with the choice of an appropriate scoring function. Fig. 1(c) shows, non-uniform sampling rates, however, cause similar portions to be designated as gaps because correspondences are restricted to be one-to-one.

When considering a large dataset of trajectories, there exists a large body of work on identifying patterns in such datasets. In [10, 20], similarity of soccer player trajectories is considered based on significant events such as passing or shooting. In [12], the authors propose the simplification of trajectories into a set of segments which are then clustered; a similar approach is taken in [16]. Perhaps most similar to the objectives in this paper are those in [3], where the Fréchet distance is used to cluster popular subtrajectories in a given trajectory dataset, and in [4], where a pathlet dictionary is learned via solving an optimization problem. However, [3] aims at finding only one optimal subtrajectory cluster and [4] needs to first convert the trajectories into paths on the underlying roadmap graph before running their optimization algorithms.

Contributions. Our first contribution is the introduction of a new model for matching similar portions of trajectories as defined using the notion of *assignments* in Section 2. Together with this model, we design an appropriate way to score assignments thus providing a unified framework encompassing previous approaches such as dynamic time warping, sequence alignment, edit-distance and others, while handling the issues present in real data. Our model can be adapted to perform the so-called *local assignment* for identifying most similar sub-trajectories between two trajectories. We also consider a *semi-continuous model* in which we interpolate each trajectory between consecutive samples, say, by a linear function, and allow matching samples to interpolated points. This approach adapts the model to sparsely sampled data.

Our second contribution is a quadratic-time dynamic programming based algorithm for computing optimal assignments between two trajectories under our scoring function (Section 3). Our algorithm builds on ideas from sequence alignment and dynamic time warping. Furthermore, our algorithm can be adapted to compute local and semi-continuous assignments as well.

Our third contribution is an efficient algorithm to segment a set of trajectories into fragments using pairwise matchings. The algorithm is designed in conjunction with our assignment model but can be adapted to other matching models as well.

We present an extensive experimental study that demonstrates the effectiveness of our matching and segmentation algorithms. We evaluate the algorithms on a number of real datasets and compare them with algorithms based on existing similarity models. We show that, in practice, our model captures the advantages of both dynamic time warping and sequence-alignment based approaches with none of their drawbacks. We also show that the segmentation algorithm is successful in capturing the shared portions of the trajectories using a small number of fragments relative to the number of sample points. Moreover, our experiments are highly indicative of the fact that the segmentation is most effective using the assignment model for matching trajectories.

2. MODEL FOR MATCHING

Let P and Q be sequences of points as defined above. We describe our model for measuring the similarity of P and Q and matching their common portions. Our model builds on the strengths of both dynamic time warping (DTW) and sequence alignment without their drawbacks: it handles non-uniform sampling of points by allowing multiple points of P to match with one point of Q, and it distinguishes deviation from noise by using the gap model.

DEFINITION 1. *An **assignment** for P and Q is a pair of functions $\alpha : P \to Q \cup \{\perp\}$ and $\beta : Q \to P \cup \{\perp\}$ for the points of P and Q respectively. If $\alpha(p_i) = \perp$ (or $\beta(q_j) = \perp$), then p_i (or q_j) is called a **gap point**. A maximal contiguous sequence of gap points in P or Q is called a **gap**.*

*An assignment is **monotone** if it satisfies the following conditions: (i) if $\alpha(p_i) = q_j$ then for all $i' > i$, $\alpha(p_{i'}) = \perp$ or $\alpha(p_{i'}) = q_{j'}$ for some $j' > j$, and (ii) if $\beta(q_j) = p_i$ then for all $j' > j$, $\beta(q_{j'}) = \perp$ or $\beta(q_{j'}) = p_{i'}$ for some $i' > i$.*

Intuitively, if a point $p_i \in P$ lies on a similar portion of the two trajectories then $\alpha(p_i)$ defines the point on Q to which p_i corresponds; p_i is a gap point otherwise. A similar interpretation holds for $\beta(\cdot)$. Unlike traditional alignment/matching models, our assignments are asymmetric, which allows it to better adapt to trajectories with different sampling rates. The notion of gaps is introduced to identify deviations between the two trajectories. Using gaps enables the identification of a "good" assignment even if there are only partial observations on any of the trajectories; we can compute an assignment for the observed portions. Distinguishing between noise and dissimilarity can be accomplished by restricting to assignments where gaps are sufficiently long (short deviations are more likely to be due to noise and thus, the underlying portions of the trajectories are similar).

It will be easier to view an assignment of P and Q in terms of the complete directed bipartite graph $G := G(P, Q) = (P \cup Q, P \times Q \cup Q \times P)$, i.e., G has a directed edge (p_i, q_j) and another directed edge (q_j, p_i) for every pair $p_i \in P$ and $q_j \in Q$. We say that a pair of edges (p_i, q_j) (or (q_j, p_i)) and (p_k, q_l) (or (q_l, p_k)) ***cross*** if $i < k$ and $j > l$ or vice versa. Under our definition, the opposite edges (p_i, q_j) and (q_j, p_i) do not cross each other.[1]

In this perspective, a ***monotone*** assignment is a set E of pairwise non-crossing edges in G so that E has at most one outgoing edge from every point in $P \cup Q$. Points with no outgoing edges are gap points and, as in Def. 1, a maximal contiguous sequence of gap points in P or Q is called a ***gap***, and the length of the sequence is called the length of the gap. Let $\Gamma(E)$ denote the set of gaps in P and Q for the assignment E. We define the ***score*** of E, denoted by $\sigma(P, Q; E)$, as

$$\sigma(P, Q; E) = \sum_{(u,v)\in E} \frac{1}{\lambda + \|u - v\|^2} + \sum_{g\in\Gamma(E)} (\theta + \Delta|g|), \quad (1)$$

where $\theta < 0, \Delta > 0$ and $\lambda > 0$ are carefully chosen parameters, as described in Section 3, $\|\cdot\|$ is the L_2-norm and $|g|$ is the length of a gap g. For appropriate comparison between different pairs of trajectories and their corresponding assignments, the above score may be normalized in a straightforward manner to provide a score between 0 and 1. We can rewrite the score in terms of the functions α, β from Def. 1 as well:

$$\sigma(P, Q; \alpha, \beta) = \sum_{\substack{p_i \in P \\ \alpha(p_i)\neq\perp}} \frac{1}{\lambda + \|p_i - \alpha(p_i)\|^2} + \sum_{\substack{q_j \in Q \\ \beta(q_j)\neq\perp}} \frac{1}{\lambda + \|q_j - \beta(q_j)\|^2} + \sum_{g\in\Gamma(\alpha,\beta)} (\theta + \Delta|g|), \quad (2)$$

where $\Gamma(\alpha, \beta)$ is the set of gaps in P and Q given α, β. We define the similarity between P and Q as

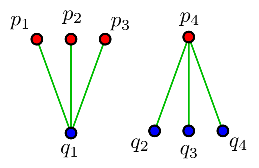
$$\sigma(P, Q) = \max_{\alpha,\beta} \sigma(P, Q; \alpha, \beta),$$

and the corresponding assignment identifies the similar portions.

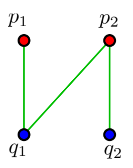
Why the directed graph? We now explain why we chose a directed graph model and an assignment. Both DTW and sequence alignment can be formulated as computing a subset of non-crossing edges in the complete *undirected* bipartite graph $P \times Q$. In particular, DTW finds a subset $D \subset P \times Q$ of non-crossing edges such that each vertex of $P \cup Q$ is incident on at least one edge of D and the total "length" of edges in D is minimum. In some applications, an appropriate choice for the length of edges would be the Euclidean distance whereas in other applications, a different function is used. On the other hand, the sequence-alignment model asks for computing a set of vertex-disjoint edges $M \subset P \times Q$ whose score is maximum, where the score is similar to (1).

It is tempting to describe an assignment as well with respect to the undirected graph but this leads to difficulties. For example, we may relax the vertex-disjoint condition from the sequence-alignment model allowing multiple points of P to match with one point of Q, (see Fig. 2(a)) and simply ask for finding a set of non-crossing edges whose score is maximum, but this is not always meaningful and introduces additional edges that are redundant. For example, in Fig. 2(b), this approach will find three edges — the diagonal edge (p_2, q_1) is spurious and is an artifact of the model because it is more meaningful to match p_1 with q_1, p_2 with q_2 and vice versa.

The directed graph model avoids this problem by not requiring the functions α and β to be symmetric; see Fig. 3.



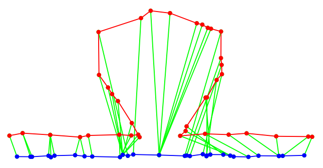
(a)



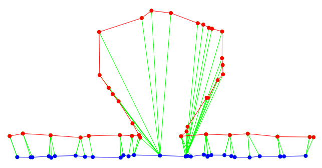
(b)

Figure 2. Pros and cons of allowing multiple matches in an undirected graph:(a) A situation where allowing multiple matches is logical. (b) A situation where allowing multiple matches does not allow us to obtain a clear correspondence.

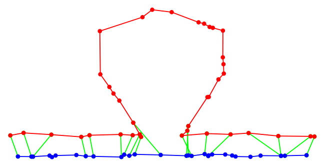
[1]Note that our definition of the crossing is topological, defined in terms of the graph G. the line segments $\overline{p_i q_j}$ and $\overline{p_k q_l}$ corresponding to two non-crossing edges (p_i, q_j) and (p_k, q_l) may cross each other (geometrically), and may be disjoint even if the graph edges are crossing.



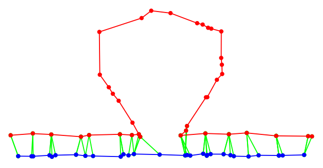
(a)



(b)



(c)



(d)

Figure 1. Matching of two trajectories using different approaches: (a) Fréchet distance, (b) dynamic time warping or average Fréchet distance, (c) sequence alignment based method, (d) our model. The green edges are the correspondences between points.

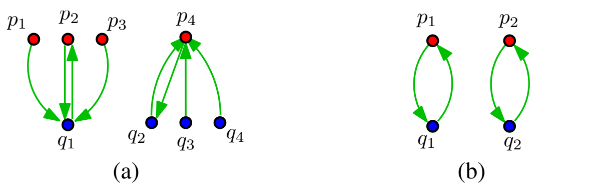


Figure 3. Using a directed graph provides a logical set of correspondences always. Comparison with the examples of Fig. 2.

There are a few other subtle advantages of using the directed graph model, omitted from this paper due to lack of space.

Remark. (i) Our framework is not limited to the scoring function (1). For example, the sequence-alignment based approach, DTW or other measures such as edit-distance are easily incorporated into our model. (ii) Note that according to our definition of assignments, vertices in G which have incoming edges but no outgoing edges are designated as gap points. This may be modified so gap points have no incoming edges.

Local assignment model. We now describe how we modify our model for finding the most similar sub-trajectories between two trajectories, or so-called local assignments. Specifically, we are only interested in computing correspondences for points along the most similar portions and scoring these. At a high level, instead of examining only the complete trajectories, we wish to examine all possible pairs of sub-trajectories and choose the one which is most similar. However, choosing the most similar sub-trajectories requires a modification of the scoring function since the pair of sub-trajectories which achieves the maximum score is always going to be the complete trajectories themselves due to the fact that all terms are positive. More precisely, given two trajectories P and Q, for any pair of sub-trajectories P', P'' of P and Q', Q'' of Q such that P' is contained within P'' and Q' is contained within Q'', we always have $\sigma(P'', Q'') \geq \sigma(P', Q')$.

We therefore, modify the score of an assignment E, represented as a set of non-crossing edges, as follows:

$$\sigma_l(P, Q; E) = \sum_{(u,v)\in E} \left[\frac{1}{\lambda + \|u - v\|^2} - \tau\right] + \sum_{g\in\Gamma(E)} (\theta + (\Delta - \tau)|g|). \tag{3}$$

Here, $\tau > \Delta$ is a threshold parameter that we subtract from each term in (1). The parameter τ defines a lower bound on the score of assignments being considered and we only consider pairs of sub-trajectories P', Q' of P and Q respectively such that $\sigma(P', Q')$ is at least this lower bound. Formally, for any two sub-trajectories P' of P and Q' of Q, let $E_{P',Q'} = P' \times Q' \cup Q' \times P'$ denote the set of edges in the complete directed bipartite graph between P' and Q'. Our goal is to compute

$$\sigma_l(P, Q) = \max_{P',Q'} \max_{\substack{E\subset E_{P',Q'} \\ E \text{ is monotone}}} \sigma_l(P', Q'; E),$$

where the maximum is taken over all subtrajectories P' and Q' of P and Q respectively, and over all monotone assignments of P', Q'.

Semi-continuous assignment model. The model discussed so far is designed to be fairly robust to both noise and sampling rate differences but still may break down when the noise or sampling rate difference are too large. By modeling the continuous trajectory from which the sample points are generated, one may handle these issues better. Specifically, instead of scoring based on distances between sample points, we consider the uncertainty inherent in them and use distances between the *most likely* locations of the sample points.

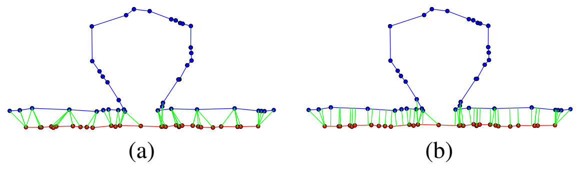


Figure 4. Discrete (a) vs semi-continuous (b).

We achieve this by interpolating between consecutive sample points and considering all points on this interpolation as candidate locations for the samples. For example, consider Fig. 4. Due to the different sampling rates of the trajectories, it is better to assign endpoints in one trajectories to points on a continuous curve obtained by a linear interpolation between endpoints of the other trajectory. Such a linear interpolation obtained by connecting two consecutive sampled points by a line segment is a common way to model the underlying trajectory. Let $\overline{P}$ and $\overline{Q}$ denote these curves for P and Q respectively. We next modify the scoring function by replacing the distance term with a function $f(\cdot)$ as follows:

$$\sigma_s(P, Q; E) = \sum_{(u,v)\in E} \frac{1}{\lambda + f(u, v)} + \sum_{g\in\Gamma(E)} (\theta + \Delta|g|). \tag{4}$$

We set $f(u, v) = \|u - v'\|^2$ where v' is the closest point to u on the segment connecting v to its preceding point on P or Q (if v is the first point, then $v' = v$). As before, we are interested in the optimal assignment $\Sigma_s(P, Q) = \arg\max_E \sigma_s(P, Q; E)$ and its score $\sigma_s(P, Q) = \max_E \sigma_s(P, Q; E)$. Using v' is a simple method of specifying a likely location for v given the linear interpolation model for the underlying trajectory. Alternatively, we may use a more sophisticated predictor for v' if we have prior knowledge of how the sample points where generated.

3. ALGORITHMS FOR MATCHING

We now describe an algorithm for computing the optimal score $\sigma(P, Q)$ and the corresponding assignment. Our algorithm, similar to that for sequence alignment [8], runs in $O(mn)$ time. Because of the asymmetric nature of the definition of assignments, the recurrence relation is complex, and requires a few auxiliary assignments and score functions.

Auxiliary functions. For $1 \leq i \leq m$, let $P_i = \langle p_1, \ldots, p_i\rangle$ denote the prefix of P of length i, and for $1 \leq j \leq n$, let $Q_j = \langle q_1, \ldots, q_j\rangle$ denote the prefix of Q of length j. The algorithm computes the similarity score $\sigma(P_i, Q_j)$ for all $1 \leq i \leq m$ and $1 \leq j \leq n$. For brevity, let $\sigma(i, j)$ denote $\sigma(P_i, Q_j)$. Let $\Sigma(i, j)$ denote the (optimal) monotone assignment corresponding to the score $\sigma(i, j)$. With a slight abuse of notation, we use both the graph representation — a set of non-crossing edges — as well as the pair of functions α, β. It will be clear from the context which of the two representations we are referring to.

For each pair i, j, to compute $\sigma(i, j)$ and $\Sigma(i, j)$ efficiently, we also compute a set of auxiliary functions described below:

- $\sigma_{\perp *}(i, j)$ and $\sigma_{*\perp}(i, j)$: $\sigma_{\perp *}(i, j)$ denotes the score of the best monotone assignment, denoted by $\Sigma_{\perp *}(i, j)$, for P_i and Q_j with the restriction that p_i is a gap point. That is, there is no outgoing edge from p_i, but our model allows $\Sigma_{\perp *}(i, j)$ to have incoming edges to p_i. We similarly define $\sigma_{*\perp}(i, j)$ and $\Sigma_{*\perp}(i, j)$.
- $\sigma_{\perp\perp}(i, j)$: the score of the best monotone assignment, denoted by $\Sigma_{\perp\perp}(i, j)$, for P_i and Q_j with the restriction that both p_i and q_j are gap points.
- $\sigma_{\phi *}(i, j)$ and $\sigma_{*\phi}(i, j)$: $\sigma_{\phi *}(i, j)$ is the score of the best monotone assignment for P_i and Q_j, denoted by $\Sigma_{\phi *}(i, j)$,

$$\sigma(i,j) = \max\Big\{\sigma_{\perp *}(i,j), \sigma_{*\perp}(i,j), \sigma_{\phi *}(i,j) + \delta(i,j), \sigma_{*\phi}(i,j) + \delta(i,j)\Big\}, \tag{5}$$

$$\sigma_{\perp *}(i,j) = \max\Big\{\sigma(i-1,j) + \theta + \Delta, \sigma_{\perp *}(i-1,j) + \Delta, \sigma_{\perp\phi}(i,j) + \delta(i,j), \sigma_{\perp\perp}(i,j)\Big\}, \tag{6}$$

$$\sigma_{\phi *}(i,j) = \max\Big\{\sigma_{\phi\perp}(i,j), \sigma_{*\phi}(i-1,j) + \delta(i,j), \sigma_{\phi *}(i,j-1) + \delta(i,j), \sigma(i-1,j)\Big\}, \tag{7}$$

$$\sigma_{\phi\perp}(i,j) = \max\Big\{\sigma_{\phi *}(i,j-1) + \theta + \Delta, \sigma_{\phi\perp}(i,j-1) + \Delta, \sigma_{*\perp}(i-1,j)\Big\}, \tag{8}$$

$$\sigma_{\perp\perp}(i,j) = \max\Big\{\sigma_{*\perp}(i-1,j) + \theta + \Delta, \sigma_{\perp\perp}(i-1,j) + \Delta, \sigma_{\perp *}(i,j-1) + \theta + \Delta, \sigma_{\perp\perp}(i,j-1) + \Delta\Big\}. \tag{9}$$

Figure 5. Recurrence relations for σ and each of the auxiliary functions. The relations for $\sigma_{*\phi}$, $\sigma_{\perp\phi}$ are symmetric to those above for $\sigma_{\phi *}$ and $\sigma_{\phi\perp}$ respectively. Here, $\delta(i,j) = 1/(\lambda + \|p_i - q_j\|^2)$.

with the restriction that p_i is not assigned to any point of Q_j but points of Q_j can be assigned to p_i, i.e., there is no outgoing edge from p_i but there can be incoming edges to p_i. The difference between $\sigma_{\phi *}(i,j)$ and $\sigma_{\perp *}(i,j)$ is that p_i is considered as a gap point in the latter and $\sigma_{\perp *}(i,j)$ includes the gap score corresponding to p_i, namely $\theta + \Delta$ if a new gap starts at p_i in $\Sigma_{\perp *}(i,j)$ and Δ otherwise, while in the former no score is added corresponding to p_i. We define $\sigma_{*\phi}(i,j)$ and $\Sigma_{*\phi}(i,j)$ analogously.

- $\sigma_{\phi\perp}(i,j)$ and $\sigma_{\perp\phi}(i,j)$: $\sigma_{\phi\perp}(i,j)$ is the score of the best monotone assignment for P_i and Q_j, denoted by $\Sigma_{\phi\perp}(i,j)$, with the restriction that p_i is not assigned to any point of Q_j as in the previous case, and q_j is a gap point. Note that there are no outgoing edges from p_i or q_j but there may be incoming edges to one or both of them, and that $\sigma_{\phi\perp}(i,j)$ does not include any score corresponding to p_i but does include a gap score for q_j. We define $\sigma_{\perp\phi}(i,j)$ and $\Sigma_{\perp\phi}(i,j)$ analogously.

For $i = j = 0$, we set $\sigma(i,j) = 0$ and all other auxiliary functions to $-\infty$. For $i > 0$ or $j > 0$, the recurrence relations for each of the auxiliary score functions are described in Fig. 5, and each value can be computed in $O(1)$ time using dynamic programming. For brevity, we set $\delta(i,j) = 1/(\lambda + \|p_i - q_j\|^2)$. We refer to [15] for details on their derivation.

We maintain a separate table for each auxiliary function and compute the entries in increasing order of i and j. For a fixed pair i, j, we compute them in the following order: $\sigma_{\perp\perp}, \sigma_{\perp\phi}, \sigma_{\phi\perp}, \sigma_{\phi *}, \sigma_{*\phi}, \sigma_{\perp *}, \sigma_{*\perp}, \sigma$. It can be verified from (5)–(9) that each of them can be computed in $O(1)$ time. Hence, the total time spent in computing the final $\sigma(P,Q)$ and $\Sigma(P,Q)$ is $O(mn)$. If we maintain the entire tables, the space used is also $O(mn)$ but it can be reduced to $O(m+n)$ [11]. We thus obtain the following:

THEOREM 3.1. *Given two sequences of points P and Q in $\mathbb{R}^d$ of lengths m and n respectively, $\sigma(P,Q)$ and the corresponding assignment $\Sigma(P,Q)$ can be computed in $O(mn)$ time using $O(m+n)$ space.*

Local assignment. For local assignment, following the same ideas as in the algorithm for local sequence alignment [8], we can compute $\sigma(P,Q)$ as defined in Section 2 in $O(mn)$ time. Intuitively, during the course of the execution of the algorithm described above, when we find that the score of aligning initial portions of the trajectories is too small, we should discard them from further consideration and start afresh. More precisely, we modify the recurrences in (5) by adding the 0 term:

$$\sigma(i,j) = \max\big\{\sigma_{\perp *}(i,j), \sigma_{*\perp}(i,j), \sigma_{\phi *}(i,j) + \delta(i,j), \\ \sigma_{*\phi}(i,j) + \delta(i,j), 0\big\}.$$

The others recurrences in (6)–(9) remain the same. Finally, instead of returning $\sigma(m,n)$, we return the score $\max_{i,j} \sigma(i,j)$. Omitting details, we get the following:

THEOREM 3.2. *Given two sequences of points P and Q in $\mathbb{R}^d$ of lengths m and n respectively, $\sigma_l(P,Q)$ and the corresponding local assignment $\Sigma_l(P,Q)$ can be computed in $O(mn)$ time using $O(m+n)$ space.*

Semi-continuous assignment. Since the semi-continuous model entails only a modification of the scoring function as described in (4), it may be computed using the same procedure as in the general case with the same time and space complexity.

THEOREM 3.3. *Given two sequences of points P and Q in $\mathbb{R}^d$ of lengths m and n respectively, $\sigma_s(P,Q)$ and the corresponding semi-continuous assignment $\Sigma_s(P,Q)$ can be computed in $O(mn)$ time using $O(m+n)$ space.*

Parameter selection. Parameter selection is important when choosing the scoring function. During the course of the algorithm, when examining a pair of points $p_i \in P$ and $q_j \in Q$, the difference in values $1/(\lambda + \|p_i - q_j\|^2)$ versus Δ dictate the choice of whether to assign $\alpha(p_i) = q_j$ or $\beta(q_j) = p_i$ versus assigning one or both as gap points.

We work with the hypothesis that all points which are "matched" have roughly the same distance. Let r be the threshold on the distance beyond which points are dissimilar. We suggest the following choices of the parameters:

$$\Delta = \frac{1}{\lambda + r^2}, \quad \theta = -l\Delta,$$

where l indicates a minimum gap length. The algorithm only chooses to start gaps when at least l points are farther than r apart since, if not, $\theta + \Delta \cdot |g|$ will be negative.

The choice of r is clear if we have semantic information about the trajectories such as what type of entities generated them. For example, if we have GPS trajectories and wish to classify similar portions as those following the same road, then the width of the road, sampling rate and GPS accuracy would determine r. On the other hand, in many situations, the choice of r is not clear. In such cases, we suggest the following simple iterative procedure: (i) Start with a rough guess of the upper bound $\hat{r}$ and compute an assignment with $r = \hat{r}$, (ii) Discard a percentage of the larger distances in this assignment and compute the root mean square (rms) of the remaining distances, (iii) Choose a new threshold which is a small factor of the rms, say $r = c_1 \cdot \mathsf{rms}$, and (iv) Repeat steps (ii) and (iii) until convergence.

When we identify all similar portions and dissimilar portions, we do not expect the rms to change significantly when we discard some of the larger distances. Hence, the assignments should converge and the algorithm should terminate.

4. SEGMENTATION OF TRAJECTORIES

In this section we describe our segmentation algorithm. Let $\mathcal{T} = \{T_1, \cdots, T_k\}$ be a set of trajectories, where each trajectory $T_i = \langle p_{i1}, p_{i2}, \cdots, p_{in_i} \rangle$. Set $\sum_{i=1}^{k} n_i = n$. Our goal is to segment each T_i into *fragments* and to identify the fragments that are shared by many trajectories; we wish to cover all the trajectories by as few fragments as possible.

Our segmentation algorithm works in two stages. The first stage assigns a label $L(p) \subseteq \{1, \cdots, k\}$ to each point p on every trajectory. Intuitively, if $j \in L(p)$ for a point $p \in T_i$, then p is a point on a subtrajectory of T_i that "spatially" overlaps with a subtrajectory of T_j and there is a point $p' \in T_j$ that corresponds to p. The labeling process is accomplished by computing the assignment between every pair of trajectories in $\mathcal{T}$. The second stage identifies maximal contiguous portions of trajectories with the same labels — they correspond to fragments. To make the step robust, we actually find portions of trajectories where the labels are "roughly" the same. We describe the complete details of these steps below.

Assignment graph. For each pair $1 \leq i < j \leq k$, using the `Assignment` algorithm described in Section 3, we compute assignment functions $\alpha_{ij} : T_i \to T_j \cup \{\perp\}$ and $\alpha_{ji} : T_j \to T_i \cup \{\perp\}$. We construct a directed graph $G = (V, E)$ for vertex set $V = \bigcup_{i=1}^{k} T_k$. We connect a pair $p_{ir}, p_{js} \in V$ by the directed edge (p_{ir}, p_{js}) if either of the following satisfies:

(i) $j = i$ and $s = r + 1$, i.e., p_{js} is the successor of p_{ir} in T_i

(ii) $j \neq i$ and $p_{js} = \alpha_{ij}(p_{ir})$.

We say that a point $p \in T_i$ is *close* to trajectory T_j if $i = j$ or there is an edge (p, p_{js}) for some $p_{js} \in T_j$. Note that our definition of closeness is not a metric definition, in which one says that p is close to q if $||p - q|| \leq r$ for some parameter $r > 0$. The metric based definition is frequently used in trajectory segmentation. There are two advantages of our approach. First, we allow an individual point p to be not very spatially close to q to be assigned to q, which can thus handle noise and outliers in a more principled way. Second, p is assigned to q by the `Assignment` algorithm if there is a subtrajectory that is common to the two trajectories — see Section 5 for more discussion on this.

Labeling. We now define a labeling function $L : V \to 2^{\{1,\cdots,k\}}$. For a vertex $p \in V$, its label $L(p) \subseteq \{1, \cdots, k\}$ is defined as follows:

$$L(p) = \{j \mid \exists p_{js} \in T_j \text{ s.t. } (p, p_{js}) \in E\}$$

That is, $L(p)$ is the set of all trajectories that are close to p. The label of all vertices in V can be computed by performing a depth first search of G.

Identifying fragments. Given labels for all trajectory points in V, we can now cluster them into fragments. Our algorithm proceeds by iteratively picking a point p as a new fragment center and assigning all the unassigned points with label similar to $L(p)$ to the new fragment, until all points in V have been assigned. More specifically, define the similarity between two labels $L(p)$ and $L(q)$, denoted as $\mathsf{sim}(L(p), L(q))$, as follows:

$$\mathsf{sim}(L(p), L(q)) = \frac{|L(p) \cap L(q)|}{|L(p) \cup L(q)|}$$

At each iteration, let G_{rem} denote the undirected subgraph of G induced by the unassigned trajectory points and removing edge directions. The algorithm first chooses an arbitrary unassigned point p as the center of a new fragment. Then it traverses G_{rem} starting from p to identify a maximal connected component C such that for each point q in C, $1 - \mathsf{sim}(L(p), L(q)) \leq \gamma$, where $\gamma \in (0, 1]$ is a parameter controlling how similar two labels should be to be considered as from the same fragment. Finally, we set a new fragment f to be the vertices in C.

In some rare cases, the algorithm may produce fragments that are not contiguous along some trajectories. In such situations, we split such a fragment into smaller fragments. We omit the routine details of this procedure from this paper. (In all the datasets we experimented with, all fragments were contiguous along each trajectory.)

This completes the description of the segmentation algorithm. The most expensive step is constructing the assignment graph, which takes $O(n^2)$ time. We conclude by making a few remarks on the fragments identified by the algorithm.

Direction of fragments. Since each trajectory is oriented and the assignment function is monotone, each fragment is also directed. If there are trajectories that share a common "curve" that is traversed in opposite directions, e.g. two way traffic on a road, our algorithm will generate two fragments for the curve — one in each direction.

Ordering of fragments. The monotonicity of the assignment function also ensures that if a pair of fragments f, f' appear on two trajectories, then they appear in the same order on both of them. This ordering is natural for many data sets. In some cases, especially when the trajectories have loops and these loops are broken independently, one may want to relax this property, but this requires defining the assignment function differently.

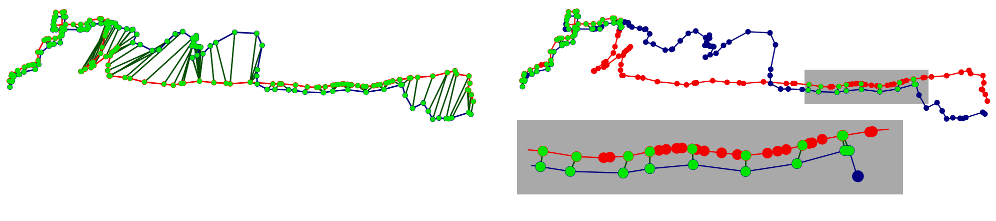
Shape of fragments. Given all the points belonging to a fragment, we can compute a curve that best describe the fragment. The curves of the fragments identified by our algorithm can be of arbitrary shape and length, which makes our model more flexible in describing the trajectory data and discovering patterns.

5. EXPERIMENTS

In this section, we present the results of an experimental study on real datasets to evaluate the effectiveness of our model and algorithms, as well as to compare them with previous approaches. We present two types of results in this section: (i) an analysis of the effectiveness of our matching model in comparison to previous approaches, and (ii) an analysis of the behavior of the segmentation algorithm from Section 4 when used in conjunction with different matching models. The goal of our experiments was not only to present a qualitative study but also to observe the characteristics of the data and how they impact the model parameters.

Datasets. We have used three datasets in our experiments: (i) 143 trajectories of school buses in Athens, Greece [9] (we call this the BUS dataset), (ii) the GeoLife project by Microsoft Research Asia [21–23] consisting of 17,621 trajectories of 182 users in China (we call this the GEOLIFE dataset), and (iii) 330 trajectories from road cycling and running exercises captured by a fitness GPS device at a constant one second sample rate (we call this the WORKOUT dataset). A subset of the trajectories in the GEOLIFE dataset are labeled with the mode of transportation used from the set {biking, walking, running, bus, car, taxi, train, subway, airplane}. From these, we extracted the trajectories with labels in {bus, car, taxi} and used only those trajectories which were sufficiently long (at least 20 sample points).

Although all three datasets are GPS trajectories, they differ significantly from each other due to the different modes of transportation and sensor equipment as well as due to road network characteristics. The WORKOUT dataset, being highly accurate and finely sampled, represents ideal conditions. The BUS dataset contains significant measurement noise but is more uniformly sampled than the GEOLIFE dataset. The GEOLIFE dataset contains significant sampling rate differences between trajectories while the GPS noise is less significant when compared with the BUS dataset. In addition,


(a) `DTW` (b) `Seq-Align`

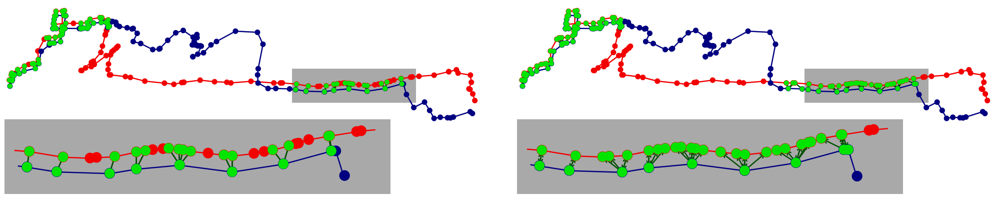

(c) `DTW-Pruned` (d) `Assignment`

Figure 6. Results on a trajectory pair from the BUS dataset.

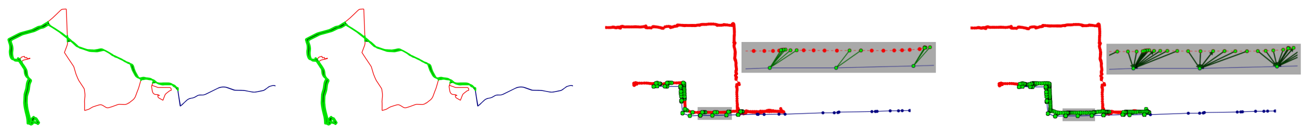

(a) WORKOUT, `DTW-Pruned` (b) WORKOUT, `Assignment` (c) GEOLIFE, `DTW-Pruned` (d) GEOLIFE, `Assignment`

Figure 7. Results on trajectory pairs from the WORKOUT and GEOLIFE datasets.

it also contains many partially observed trajectories (long stretches with no samples).

Algorithms tested. We used the DTW and sequence-alignment based approaches to provide a basis for comparison of our approach. We refer to these algorithms as `DTW` and `Seq-Align` and to our algorithm from Section 3 as `Assignment` throughout this section.

Since `DTW` tries to find correspondences for every point on a trajectory, we apply a simple heuristic to make its comparison with `Assignment` and `Seq-Align` fairer: after computing the `DTW` correspondences, we prune all correspondences where the distance between the two constituent points is greater than a threshold. Recall that the parameter selection for our scoring function (cf. Sec. 3) uses a similar distance threshold r. We call this modified `DTW` heuristic `DTW-Pruned`.

Because of lack of space, we omit our experiments on local and semi-continuous assignments (see [15] for these results).

Pairwise matching results. To show the effectiveness of our algorithms, we first present results on a single pair of trajectories from each of the datasets. We chose pairs which exhibited significant similar portions as well as dissimilar portions and have sufficiently dissimilar sampling rates in the case of the pairs from the BUS and GEOLIFE datasets (the WORKOUT dataset trajectories have uniform sampling rates). The results exhibited on these pairs are representative of the results on other trajectories as well. We chose a distance threshold $r = 25$ m for `Seq-Align`, `DTW-Pruned` and `Assignment` and a minimum gap length $l = 2$ for `Seq-Align` and `Assignment`. In all cases, we perturbed one of the trajectories slightly in the figures to present the results more clearly.

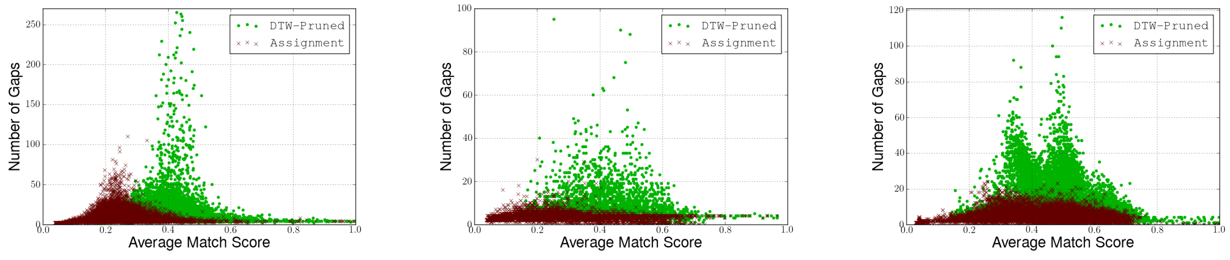
In the case of both `Seq-Align` and `DTW-Pruned`, the similar portions between trajectories are often fragmented, i.e., small gaps are present in between the matched portions. In the former, this is due to the fact that that correspondences are one-to-one while in the latter, it is due to its sensitivity to the distance threshold (even distances slightly larger than r are discarded). This phenomenon does not happen in the case of `Assignment` since it is robust to both outliers and sampling rate variations. The imposition of a minimum gap length generates correspondences which may be desired despite having distance larger than the threshold due to the characteristics of the trajectories in their neighborhood. Thus, it is more robust to uncertainties in the choice of the distance threshold. In the WORKOUT dataset, both `DTW-Pruned` and `Assignment` perform well due to the uniform sampling rate, velocity and accuracy of sampling. See Figures 6 and 7 for results showing this behavior. The correspondences computed by `DTW` are clearly meaningless and we include it only for the sake of completeness. Due to lack of space, we omit both `DTW` and `Seq-Align` from Fig. 7.

We next examined the number of gaps obtained by matching all pairs of trajectories in the datasets using `DTW-Pruned` and `Assignment`. We measured this number against an average match score computed by taking only the first term in (1) and normalizing it, i.e., we use the score

$$\sigma_m(P,Q;E) = (\lambda/|E|) \sum_{(u,v)\in E} (1/(\lambda + \|u-v\|^2)),$$

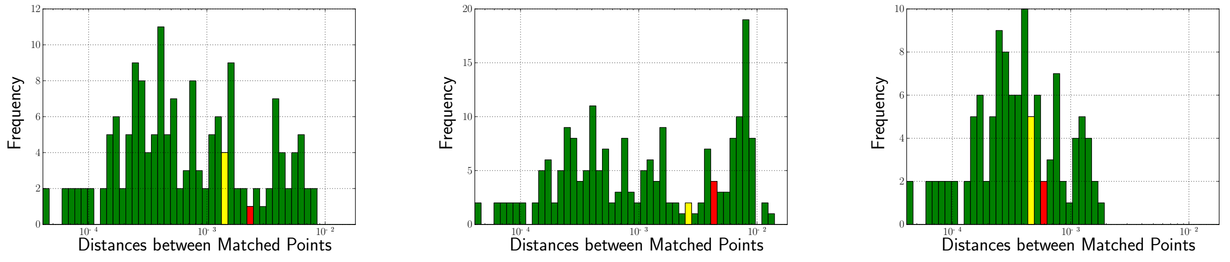
where E is the edge set of the optimal matching. Fig. 8 shows these results. Here, there is a point for each pair of trajectories such that $|E| > 0$ with the average match score and number of gaps being the x and y coordinates respectively. Clearly, the number of gaps in `DTW-Pruned` is significantly higher on average than in `Assignment`, due to the fragmentation of matched portions present in `DTW-Pruned`. Since `DTW-Pruned` uses a strict threshold, the score variations are lower than in the case of `Assignment`, although the differences are not too significant.

Parameter selection. We chose an initial distance threshold $r = 500$ m for our iterative parameter selection process (see Section 3), and λ was set to $(0.4r)^2$ as we found it to be the most suitable.



(a) BUS dataset (b) GEOLIFE dataset (c) WORKOUT dataset

Figure 8. Variations in average match score and number of gaps over all trajectory pairs in the three datasets.



(a) (b) (c)

Figure 9. The assignments and corresponding distance distributions computed at various stages of the iterative algorithm on the BUS dataset. (a) is at the first iteration, (b) at an intermediate stage and (c) at the point of convergence.

Fig. 9 shows the assignment and distribution of distances between matched points for the pair from the BUS dataset at the beginning, end and an intermediate step of the iterative approach. As we can see, larger distances are slowly pruned away until we reach the optimal assignment. The variance in the final distribution is due to the sparse sampling and noise in the data.

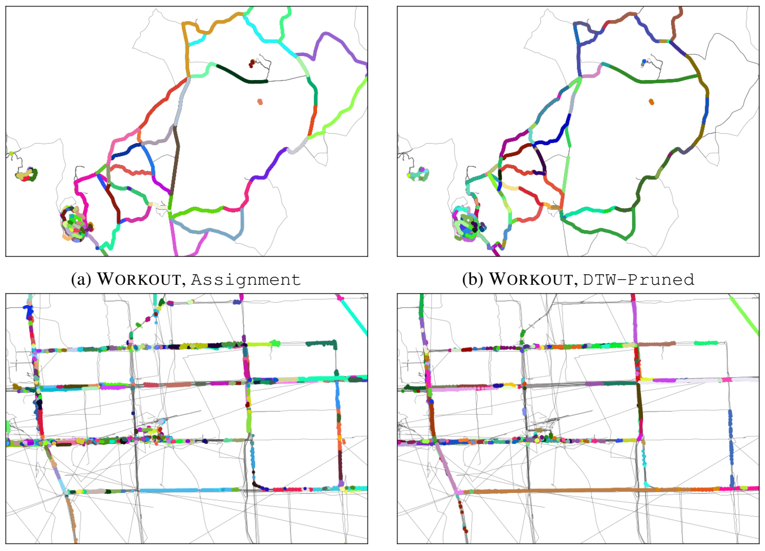
Segmentation results. We now present results of the segmentation algorithm from Section 4 on the three datasets. In addition to the threshold γ on the similarity of labels assigned by the labeling process, we used two thresholds η and ρ on the length of fragment labels and number of points belonging to a fragment respectively. By filtering fragments whose label size is less than η and/or number of points is less than ρ, we can visualize the important fragments. Due to the way the labeling process is executed, η provides an indication of the number of trajectories sharing the fragment while ρ is more direct.

Our segmentation algorithm (see Section 4) can be extended to use any algorithm that computes a matching between the points in two trajectories, and its effectiveness depends on the effectiveness of the underlying matching model. We first compared the effectiveness of `DTW-Pruned` and `Assignment`. The segmentation results are quite different between the two models with significantly different split points along trajectories. This is expected in the case of the BUS and GEOLIFE datasets due to the presence of larger numbers of outliers and sparse sampling but is more surprising for the WORKOUT dataset since `DTW-Pruned` and `Assignment` appear to perform identically for individual pairs. This suggests that even very few outliers have significant effects on the segmentation when using `DTW-Pruned`. Fig. 10 shows results for the WORKOUT and GEOLIFE datasets with thresholds $\eta = 2$, $\rho = 10$ and $\gamma = 0.4$.

Next, we examine two measures of the quality of the fragmentation: (i) fraction of sample points captured, and (ii) number of fragments obtained. Fig. 11 shows the variation of these measures with the thresholds η, ρ and γ. In all cases where the parameters are fixed, $\eta = 0$, $\rho = 10$ and $\gamma = 0.4$. The number of points captured by using `Assignment` is consistently higher than when using `DTW-Pruned` while the number of fragments is consistently lower. Even when the number of fragments is higher for `Assignment` (as sometimes for the GEOLIFE dataset), the fraction of points captured is significantly higher than `DTW-Pruned`. Since the number of road sections is relatively small in all the datasets as compared to the number of points, we conclude that `Assignment` is consistently better at representing the shared portions of the trajectories with fewer fragments. Finally, Fig. 12 shows the results of the segmentation algorithm on the WORKOUT dataset using the `Assignment` model for $\gamma \in \{0.1, 0.4, 0.7\}$. Here, $\gamma = 0.1$ provides a reasonable segmentation of the trajectories while $\gamma = 0.4$ does not vary much. Setting $\gamma = 0.7$, however, merges some of the fragments and the segmentation starts to fall apart. We note that the threshold γ provides a natural way to compute segmentations at different hierarchies which may be useful for clustering purposes.

6. DISCUSSION

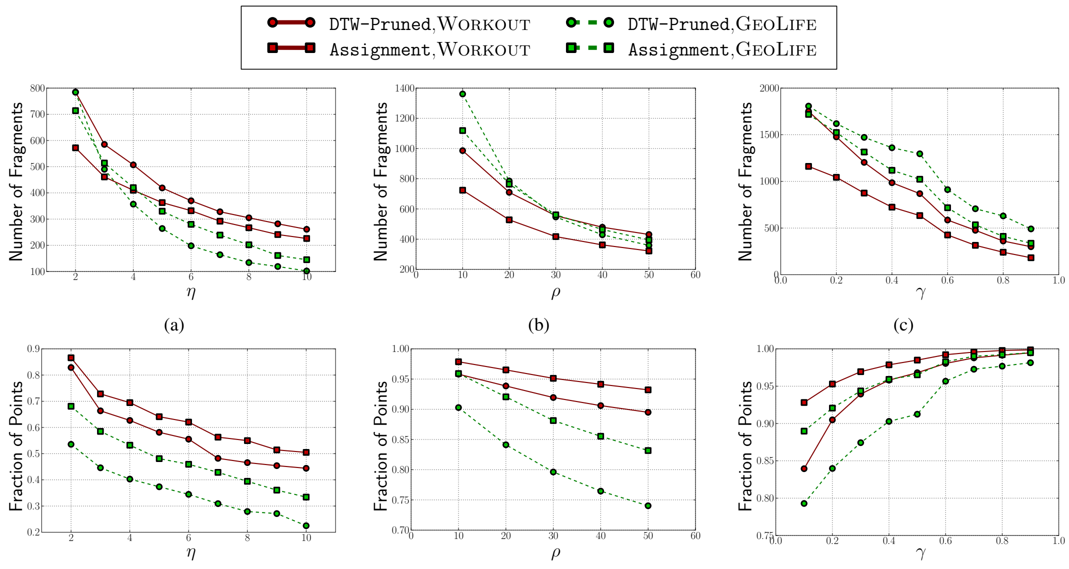
We have shown that our matching framework captures the advantages of both DTW and sequence alignment based approaches for identifying trajectory similarity, and that it is able to exceed their accuracy. Experiments show that the approach is highly accurate in identifying similar portions of trajectories from real datasets. Even without an accurate prior knowledge of distances between points based on which to compute similarity, our iterative procedure is able to converge at the point where similar portions are identified and distinguished accurately from dissimilar portions. Further, the segmentation algorithm is able to capture the shared portions of the trajectories in the datasets better using our model as compared to other approaches. This indicates that our model and segmentation algorithm are effective at capturing the characteristics of the data.



(a) WORKOUT, `Assignment` (b) WORKOUT, `DTW-Pruned`

(c) GEOLIFE, `Assignment` (d) GEOLIFE, `DTW-Pruned`

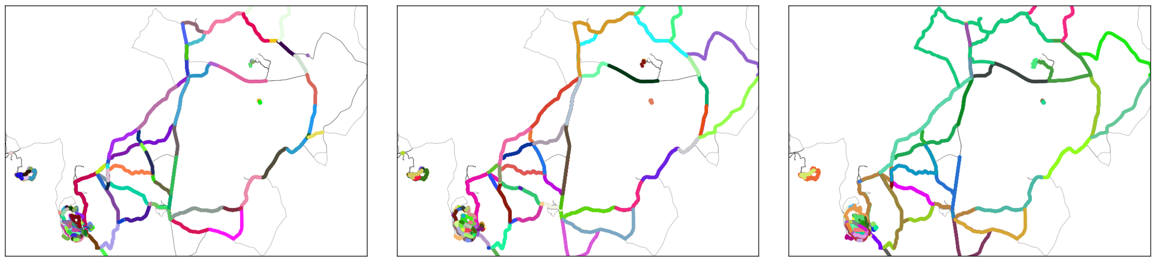
Figure 10. Visual comparison of segmentation algorithm using `DTW-Pruned` and `Assignment` matching models.



(a) (b) (c)

(d) (e) (f)

Figure 11. Quantitative comparison of segmentation algorithm using `DTW-Pruned` and `Assignment` matching models. (a)-(c) and (d)-(f) show the variation in number of fragments and fraction of points captured with η, ρ and γ.



(a) WORKOUT dataset, $\gamma = 0.1$ (b) WORKOUT dataset, $\gamma = 0.4$ (c) WORKOUT dataset, $\gamma = 0.7$

Figure 12. Visual comparison of segmentation using the `Assignment` model for different thresholds γ. In all cases, $\eta = 2$ and $\rho = 10$.

There are many directions for future research. One direction is to develop algorithms to handle extremely sparse sampling in the trajectory dataset. Such algorithms could explore the fact that there are usually sufficient sample points along a road section even though there might be very few samples along any particular trajectory. After running our segmentation algorithm, each trajectory can be represented more succinctly as a sequence of fragments, and another interesting direction is to cluster the trajectories based on this new representation.

7. REFERENCES

[1] P. K. Agarwal, R. B. Avraham, H. Kaplan, and M. Sharir. Computing the Discrete Fréchet Distance in Subquadratic Time. In *Proc. 24th ACM-SIAM SODA*, pages 156–167, 2013.

[2] H. Alt and M. Godau. Computing the Fréchet Distance between Two Polygonal Curves. *Int. J. Comput. Geom. Appl.*, 5:75–91, 1995.

[3] K. Buchin, M. Buchin, J. Gudmundsson, M. Löffler, and J. Luo. Detecting Commuting Patterns by Clustering Subtrajectories. *Int. J. Comput. Geom. Appl.*, 21(3):253–282, 2011.

[4] C. Chen, H. Su, Q. Huang, L. Zhang, and L. Guibas. Pathlet Learning for Compressing and Planning Trajectories. In *this proceedings*, 2013.

[5] L. Chen and R. Ng. On the Marriage of Lp-norms and Edit Distance. In *Proc. VLDB*, pages 792–803, 2004.

[6] L. Chen, M. T. Özsu, and V. Oria. Robust and Fast Similarity Search for Moving Object Trajectories. In *Proc. ACM SIGMOD*, pages 491–502, 2005.

[7] Defense Systems. Soldiers gain combat edge with smart helmets . `http://defensesystems.com/articles/2009/05/06/defense-it-3-helmet-networks.aspx`, 2009.

[8] R. Durbin, S. R. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.

[9] E. Frentzos, K. Gratsias, N. Pelekis, and Y. Theodoridis. Nearest Neighbor Search on Moving Object Trajectories. In C. Bauzer Medeiros, M. Egenhofer, and E. Bertino, editors, *Advances in Spatial and Temporal Databases*, volume 3633 of *Lecture Notes in Computer Science*, pages 328–345. Springer Berlin Heidelberg, 2005.

[10] S. Hirano and S. Tsumoto. A Clustering Method for Spatio-temporal Data and Its Application to Soccer Game Records. In D. Ślezak, G. Wang, M. Szczuka, I. Düntsch, and Y. Yao, editors, *Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing*, volume 3641 of *Lecture Notes in Computer Science*, pages 612–621. Springer Berlin Heidelberg, 2005.

[11] D. S. Hirschberg. A Linear Space Algorithm for Computing Maximal Common Subsequences. *Commun. ACM*, 18(6):341–343, 1975.

[12] J.-G. Lee, J. Han, and K.-Y. Whang. Trajectory Clustering: A Partition-and-Group Framework. In *Proc. ACM SIGMOD*, pages 593–604, 2007.

[13] T. Meyer, M. D'Abramo, A. Hospital, M. Rueda, C. Ferrer-Costa, A. Pérez, O. Carrillo, J. Camps, C. Fenollosa, D. Repchevsky, J. L. Gelpí, and M. Orozco. MoDEL (Molecular Dynamics Extended Library): A Database of Atomistic Molecular Dynamics Trajectories. *Structure*, 18(11):1399–1409, 2010.

[14] L. Rabiner and B. Juang. *Fundamentals of Speech Recognition*. PTR Prentice Hall, 1993.

[15] S. Sankararaman, P. K. Agarwal, T. Mølhave, and A. P. Boedihardjo. Computing Similarity between a Pair of Trajectories. *arXiv*, 1303.1585, 2013.

[16] C. Sung, D. Feldman, and D. Rus. Trajectory clustering for motion prediction. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1547–1552, 2012.

[17] The Wall Street Journal. Apple, Google collect user data . `http://online.wsj.com/article/SB10001424052748703983704576277101723453610.html`, 2011.

[18] M. Vlachos, D. Gunopoulos, and G. Kollios. Discovering Similar Multidimensional Trajectories. In *Proc. 18th IEEE ICDE*, pages 673–684, 2002.

[19] Y. Wang, Y. Zhu, Z. He, Y. Yue, and Q. Li. Challenges and opportunities in exploiting large-scale gps probe data. Technical Report HPL-2011-109, HP Laboratories, 2011.

[20] T. Wolle and J. Gudmundsson. Towards Automated Football Analysis: Algorithms and Data Structures. In *Proc. 10th Australasian Conference on Mathematics and Computers in Sport*, 2010.

[21] Y. Zheng, Q. Li, Y. Chen, X. Xie, and W.-Y. Ma. Understanding Mobility Based on GPS Data. In *Proc. 10th ACM UbiComp*, pages 312–321, 2008.

[22] Y. Zheng, X. Xie, and W.-Y. Ma. GeoLife: A Collaborative Social Networking Service among User, Location and Trajectory. *IEEE Data Eng. Bull.*, 33(2):32–39, 2010.

[23] Y. Zheng, L. Zhang, X. Xie, and W.-Y. Ma. Mining Interesting Locations and Travel Sequences from GPS Trajectories. In *Proc. 18th ACM WWW*, pages 791–800, 2009.