Similarity Search for Spatial Trajectories Using Online Lower Bounding DTW and Presorting Strategies

Marie Kiermeier[1] and Martin Werner[2]

1 Mobile and Distributed Systems Group, Ludwig-Maximilians-Universität, Munich, Germany
marie.kiermeier@ifi.lmu.de

2 Institute of Cartography and Geoinformatics, Leibniz-University, Hanover, Germany
martin.werner@ikg.uni-hannover.de

Abstract

Similarity search with respect to time series has received much attention from research and industry in the last decade. Dynamic time warping is one of the most widely used distance measures in this context. This is due to the simplicity of its definition and the surprising quality of dynamic time warping for time series classification. However, dynamic time warping is not well-behaving with respect to many dimensionality reduction techniques as it does not fulfill the triangle inequality. Additionally, most research on dynamic time warping has been performed with one-dimensional time series or in multivariate cases of varying dimensions. With this paper, we propose three extensions to $\mathrm{LB}_{\mathrm{Rotation}}$ for two-dimensional time series (trajectories). We simplify $\mathrm{LB}_{\mathrm{Rotation}}$ and adapt it to the online and data streaming case and show how to tune the pruning ratio in similarity search by using presorting strategies based on simple summaries of trajectories. Finally, we provide a thorough evaluation of these aspects on a large variety of datasets of spatial trajectories.

1998 ACM Subject Classification H.2.8 Database Applications

Keywords and phrases Trajectory Computing, Similarity Search, Dynamic Time Warping, Lower Bounds, k Nearest Neighbor Search, Spatial Presorting

Digital Object Identifier 10.4230/LIPIcs.TIME.2017.18

1 Introduction

Dynamic time warping (DTW) is a family of algorithms designed to compare time series with each other. Though dynamic time warping lacks several important properties such as the triangle inequality, it has seen a wide adoption in many fields. This might be due to the very intuitive definition, relatively efficient calculations, and, of course, successful applications. In the domain of time series classification, each sequence in a set of time series has an associated class label and the task is to predict class labels on unknown instances given a labeled training dataset. In this setting, one nearest neighbor using dynamic time warping with warping constraints calibrated from cross-validation (DTWCV) on the training set is surprisingly hard to beat [7]. A recent paper of Bagnall et al. confirms this observation [2]. They evaluate on a large set of datasets and are able to beat DTWCV with one nearest neighbor, but only with a large ensemble of data transformations. This paper impressively confirms the importance, effectiveness, and efficiency of dynamic time warping similarity search in time series classification.

Nowadays, large datasets of time series are available to evaluate and discuss results on. However, one-dimensional time series predominate these datasets. With this paper, however, we want to concentrate on the special area of spatial time series, often called trajectories. In this area, trajectories are considered as time series in a two-dimensional space. The two-dimensional space has important characteristics allowing for a novel type of lower bound, $\text{LB}_{\text{Rotation}}$, and are very important for practical applications such as tracking, navigation, and location-based services. With this paper, we evaluate and improve the lower bound $\text{LB}_{\text{Rotation}}$ proposed by Gong et. al [10] and extend it with efficient presorting strategies for scalable nearest neighbor search.

More concretely, the main contributions of this paper are as follows:

- We show that $\text{LB}_{\text{Rotation}}$ can be calculated more efficiently without least square fitting and without loss of tightness.
- We show that $\text{LB}_{\text{Rotation}}$ can be calculated in a data stream setting.
- We define three presorting strategies leading to considerable increase in pruning power.
- We provide a thorough evaluation of $\text{LB}_{\text{Rotation}}$ variants and presorting strategies on datasets of very different characteristics.

The remainder of the paper is structured as follows: Section 2 reviews related work on dynamic time warping including speedup techniques. In Section 3, we explain how to adapt the lower bound $\text{LB}_{\text{Rotation}}$ for an online algorithm. Additionally, we introduce presorting strategies in Section 3.3.

In Section 4, we provide a detailed analysis of four variants of $\text{LB}_{\text{Rotation}}$ on several datasets and discuss the impact and caveats with presorting strategies for similarity search. Finally, Section 5 concludes the paper with some hints on possible future work.

2 Related Work

Dynamic time warping is a distance definition which has found wide adoption in various domains. Historically, dynamic time warping originates in the area of speech recognition [23], but has soon been used in many other domains including geometric recognition tasks such as handwriting recognition and signature verification [6], in computer vision [1], in shape retrieval [18], in biology and medicine [14], pattern recognition [4], and recently similarity search for spatial trajectories [24].

The simplest way of calculating dynamic time warping by an algorithm is given by dynamic programming [29]:

$$\delta_{DTW}(a_{1..n}, b_{1..m}) = \begin{cases} 0, \text{ if } a \text{ and } b \text{ are both empty,} \\ \infty, \text{ if only one of } a \text{ and } b \text{ is empty,} \\ \delta(a_n, b_m) + \min \begin{cases} \delta_{DTW}(a_{1..n-1}, b_{1..m-1}) \\ \delta_{DTW}(a_{1..n-1}, b_{1..m}) \\ \delta_{DTW}(a_{1..n}, b_{1..m-1}) \end{cases} \end{cases} \tag{1}$$

This distance δ_{DTW} can be computed in $\mathcal{O}(mn)$ time [4] where m and n are the lengths of the trajectories.

One way to achieve this is by first evaluating a complete distance matrix

$$D_{i,j}(a_{1..n}, b_{1..m}) = (\delta(a_i, b_j))_{i=1..n, j=1..m}$$

between all points of the trajectories. Then, the problem of calculating dynamic time warping is reduced to finding the shortest path through this matrix from the lower left corner to the upper right corner using only movements to the right, up or diagonally up and summing up all the entries in the distance matrix. The sequence of coordinates in this distance matrix gives the warping path.

Due to its quadratic complexity in time and space for equal-length trajectories this measure is only applicable to short time series. There are two different approaches to speeding up dynamic time warping calculation [22]: either one creates additional constraints on the warping path that allow to only reduce the number of evaluated cells in the distance matrix [21, 11] or one carefully reduces the size of the input trajectories beforehand. However, neither of these approaches will be strictly correct as some warping paths cannot be found in both approaches.

2.1 Lower Bounds

For speeding up similarity search performance under DTW, lower bounds can be used for pruning candidate trajectories. A lower bound in this context is a function taking two sequences and calculating a lower bound to their true DTW distance, that is $\mathrm{LB}(Q, R) \leq \delta_{DTW}(Q, R)$.

The most popular lower bounding measures for dynamic time warping are $\mathrm{LB}_{\mathrm{Kim}}$ [13], $\mathrm{LB}_{\mathrm{Yi}}$ [29], $\mathrm{LB}_{\mathrm{Keogh}}$ [12], and $\mathrm{LB}_{\mathrm{Improved}}$ [15].

Even if there exist multidimensional versions of them like $\mathrm{LB}_{\mathrm{MV}}$ [20], they have actually been designed for one dimensional time series. In contrast, Gong et al. propose $\mathrm{LB}_{\mathrm{Rotation}}$ as a first approach which also takes into account special characteristics of trajectories, i.e. 2-dimensional time series [10].

The idea of $\mathrm{LB}_{\mathrm{Rotation}}$ is to first divide the query trajectory $Q = q_1, q_2, ..., q_n$ in consecutive and non-overlapping segments s_i which are as straight as possible. Then, each segment is rotated such that it is parallel to the X-axis. By doing so, the area of the bounding envelopes which are defined as $U_i = \max(q_{i-c} \dots q_{i+c})$ and $L_i = \min(q_{i-c} \dots q_{i+c})$, where c is the global warping constraint, is reduced. With these small envelopes, we get a more accurate, i.e. greater, lower bound, since the second trajectory $R = r_1, r_2, ..., r_n$ pokes out of the envelopes more often. Note that the rotation of the points in this trajectory is defined from the rotation of the matching point candidates in the query. If the candidates hit several segments, we are free to use the minimum of the resulting numbers.

Then, similar to the construction of $\mathrm{LB}_{\mathrm{Keogh}}$, the lower bound $\mathrm{LB}_{\mathrm{Rotation}}$ is defined to be the sum of the distances d to the segments' envelopes:

$$\mathrm{LB}_{\mathrm{Rotation}}(Q, R) = \sqrt{\sum_{i=1}^{n} \min_{s_j \in S_i} d(r_i, \mathrm{Env}(s_j))}, \tag{2}$$

with S_i is the set of segments of Q which are hit by the candidates and $\mathrm{Env}(s_j) = (U_1, U_2, ..., U_m, L_1, L_2, ..., L_m)$ the corresponding envelopes.

Figure 1 illustrates this situation: a sinus wave is considered as a two-dimensional spatial trajectory (i.e., time is moving forward along the trajectory and not along the X-axis). First, this trajectory is split into linear segments (lower left of Figure 1), then these linear segments are rotated to become parallel to the X-axis resulting in quite small envelopes (upper right in Figure 1). Finally, this trajectory is compared to the graph of a cosinus wave. Note that the points of this graph have to be rotated with the rotations prescribed from the query rotations. Down right in Figure 1, one sees that the intersection of the query and the rotated example is quite small leading to efficient pruning. For reference, Figure 2 depicts the same situation for the multivariate version of $\mathrm{LB}_{\mathrm{Keogh}}$.

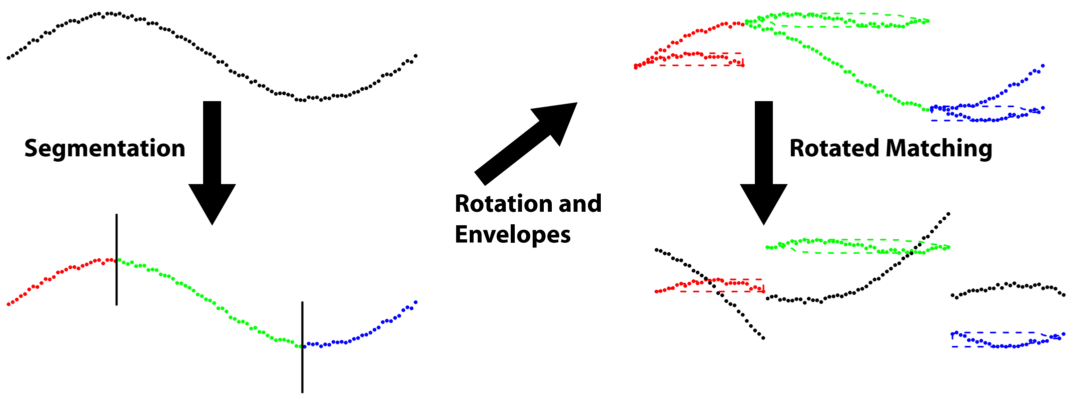


Figure 1 Principles of $\mathrm{LB}_{\mathrm{Rotation}}$ illustrated with a sinus graph.

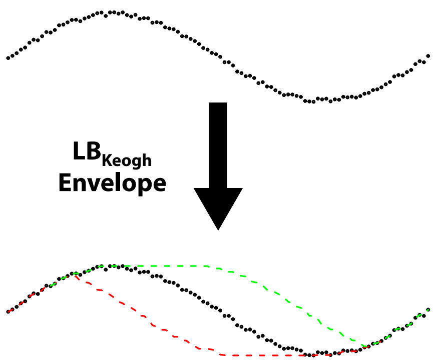


Figure 2 Envelope of $\mathrm{LB}_{\mathrm{Keogh}}$ for a sinus graph.

Note, however, that the version of $\mathrm{LB}_{\mathrm{Rotation}}$ as proposed by Gong et al. is an offline algorithm and cannot be applied in streaming settings. It is based on using the Douglas-Peucker algorithm [8] for segmenting the query trajectory.

3 Online Similarity Search

In this section, we first present our efficient online solution for $\mathrm{LB}_{\mathrm{Rotation}}$ and show, how it can be used for k-nearest neighbor search. Then, four presorting strategies based on extremely simple trajectory summaries for increasing the pruning power when searching trajectory databases are explained in detail.

3.1 Improvements for Lower Bounds Based on Segment Rotation

Facing the challenge of *online* similarity search, the current version of $\mathrm{LB}_{\mathrm{Rotation}}$ can not be used as a lower bound for DTW, since it is not applicable to data streams. Therefore, in the following, we propose an online version of $\mathrm{LB}_{\mathrm{Rotation}}$ which can be applied for speeding up online similarity search. Since we do not have access to the complete trajectory when working with data streams, the segmentation step of $\mathrm{LB}_{\mathrm{Rotation}}$ has to work online, too. Therefore, we replace Douglas Peucker with the Opening Window Algorithm [9, 33], which segments the incoming data stream online in subsequences as straight as possible. By doing this, every time the segmentation algorithm provides a segment, the corresponding rotation angle and envelope can be computed and recorded. Thanks to the sum in Eq. 2, the lower bound can then easily be updated online by computing for the incoming data points the matching points in the query and summing the minimal distances.

In addition, not only the method for the segmentation step is interchangeable, but also the rotation angles can be determined in different ways. For our implementation, we decided not to use *least square fitting* as suggested by Gong et al., but the arcus tangens of the segment's start and end point. This simplification is possible, since the segmentation algorithms are already supposed to find straight line segments and so the subsequence can be approximated by the line from the start to the end point. With this simplification, the rotation angle θ is:

$$\theta = -\arctan\left(\frac{\Delta y}{\Delta x}\right),$$

where $\Delta x = x_{last} - x_{start}$ and $\Delta y = y_{last} - y_{start}$ with x_{start} (y_{start}) are the first x-(y)-coordinates of the segment and x_{end} (y_{end}) the last x-(y)-coordinates, respectively.

3.2 k-Nearest Neighbor Search

For similarity search applications, especially for k-nearest neighbors, lower bounds can be used to speed up the process of searching. Therefore, a search by example strategy is employed: one proceeds with a linear search over the complete dataset remembering the best result (e.g., nearest neighbor) found so far. For each new element in the linear search, the lower bound is calculated. If the lower bound is higher than the current nearest neighbor distance, we can avoid calculating the true distance and prune the element. If, however, the lower bound is smaller than the current nearest neighbor distance, the true distance must be computed in order to decide, whether the current element is better than the nearest neighbor found so far.

In the context of this algorithm, the two metrics *pruning ratio* and *tightness* are widely used to assess the quality of lower bounds.

Tightness is a measure comparing the lower bound with the true distance and we define it, according to [12], as follows:

$$T = \frac{\text{Value of the Lower Bound}}{\text{True Value of DTW}}.$$

This creates a value between zero and one, the larger, the better. It summarizes how much the lower bound resembles the true distance. For a tightness of one, the lower bound coincides with the distance, for a tightness of zero, the lower bound is the trivial lower bound given by a the constant zero function.

Though tightness is a good measure for comparing different lower bounds, it is not directly related to search speed. Therefore, a more practicable metric called *pruning ratio* has been defined. This measure is based on counting how often the lower bound has successfully avoided the calculation of the true distance.

Formally, the *pruning ratio* can be defined as follows:

$$P = \frac{\text{Number of Omitted Elements}}{\text{Total Number of Elements in the Dataset}}.$$

This measure creates a value between zero and one with one representing the best case.

Note that tightness is a quite general measure and only influenced by the dataset and the lower bound algorithm. The pruning ratio also depends on the best known candidate in a linear search at a given iteration, hence, on the ordering of the dataset. As an extreme example, consider the nearest neighbor being the first element to be inspected. Then, the lower bound has a higher chance of pruning elements as if the best example is very far away from the beginning of the linear scan.

We will exploit this fact by proposing some presorting strategies for trajectories in order to increase the pruning ratio while the tightness keeps constant.

3.3 Presorting Strategies

A central idea of this paper is to increase the pruning ratio by presorting strategies for two-dimensional time series. Even though very simple summaries of trajectories such as the centroid of the set of points do not carry much information about trajectory similarity with respect to DTW distance, they can be used to reorder the dataset or to prepone the analysis of promising candidates in order to be able to more often use the lower bound for pruning. Note that we do not want to sort the complete database in order to have the majority of the linear scan access data in its ordering in memory or on disk. We just prepone a set of examples, before we enter the classical linear scan.

The main idea is to summarize a complete trajectory (e.g., the sequence of spatial points) into a single indexable object. For example, we can summarize the point set of a trajectory by the centroid of the point set. As the centroid maps a trajectory database into a database of points of the same size, existing efficient point indexing strategies such as R^*-trees can be used to efficiently retrieve k-nearest neighbors.

The following sections introduce three presorting strategies for trajectory data in the same framework: first, a single object summary is calculated and then similarity of these summaries can be used to prepone some examples to increase pruning power. Of course, many other variants can be easily defined and it depends on the application and datasets how to choose such a summary.

3.3.1 Presorting Using Centroids

For all datasets, we used the centroid of the points of the given trajectory (i.e., the mean of the coordinates) as a summary. For increasing pruning ratio, we first looked at a given number of nearest neighbors in the set of centroids (which is a simple point set to be indexed for example by an R^* tree). The idea is that similarity of centroids is a necessary condition for similarity with respect to DTW: if the centroids are far from each other, the average distance between points of two trajectories will be large, hence, DTW distance will be large.

3.3.2 Presorting Using the Last Point

Another point-based presorting strategy is given by bringing forward some trajectories whose end points are near each other. This is especially useful in online scenarios in which similarity is in any way searched with respect to the immediate past.

3.3.3 Presorting Using the Jaccard distance of Geohash sets

The third presorting strategy is given by encoding every point of every time series using the Geohash mechanism [30, 19] and considering the Jaccard distance of the involved sets as an indicator of nearness. This strategy is especially interesting as the Jaccard distance can be indexed by using Bloom filters in an extremely memory- and time-efficient way [26].

4 Evaluation

We implemented the lower bounds LB_{Keogh} and four different variants of $\text{LB}_{\text{Rotation}}$ as a library for the statistical computing environment R using C++[1]. We appreciate that the authors of the initial paper on $\text{LB}_{\text{Rotation}}$ provided us with their implementation, which we used as a reference for reconstructing some details of their approach.

In a preparation step, we preprocess all trajectories of a dataset to contain the same number of points (see Equation 2). Therefore, we use piecewise linear interpolation (PLI) for trajectories with fewer points than desired as well as picewise aggregate approximation (PAA) for trajectories which are too long. Optionally, we independently normalize trajectories by subtracting the mean and dividing by the standard deviation.

For the segmentation of trajectories, we implement three approaches: the first approach is, of course, Douglas Peucker simplification with a given threshold. Note that the original paper prescribed the number of segments to generate. Due to the variations in our datasets, the approach using a threshold in terms of the distance in the dataset is better. However, we do not know the number of segments for a given trajectory beforehand. For being able to process $\text{LB}_{\text{Rotation}}$ in an online manner, we used the Opening Window Algorithm. Though the Opening Window Algorithm has quadratic complexity, it is often used as a replacement for Douglas Peucker for the case of data streams [33].

For assessing the orientation of segments, we used the linear models provided by the R environment with the `lm` function. As this function rejects to fit lines to sequences of equal points, we completed the linear model fitting by setting an angle of zero in any case, where `lm` rejects the creation of a linear model for a segment. This amounts to not rotating these segments. This is a minor difference to the original version, which tries to derive an angle also in numerically unstable situations. Additionally, we argue that the segmentation algorithm should have already taken care of creating linear segments and just assess the orientation based on the angle formed by the first and last point of the segments.

Presorting has been implemented by an appropriate index for the given presorting strategy: An R^* tree as provided by the `boost::geometry::index` library for the case of points and Bloom Aggregated Cell Representations for the case of Jaccard distance of Geohash sets [26].

4.1 Data Sets

We evaluate our approach against the original $\text{LB}_{\text{Rotation}}$ and LB_{Keogh} and use several variants of segmentation and line fitting algorithms on datasets of varying type.

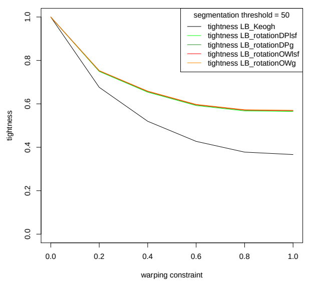
4.1.1 Geolife GPS Trajectories

The Geolife dataset is a large GPS dataset containing 18,670 GPS trajectories with 24,876,978 points. It contains the trajectories of 178 users in a period of over four years from April 2007 to October 2011. The data is given in WGS84 using latitude and longitude [31, 32].

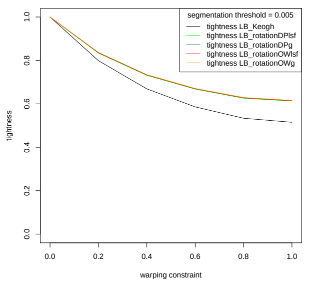
4.1.2 Character Trajectories

The Character dataset, available from the UCI Machine Learning Repository, contains 2858 trajectories of writing 20 different letters on a digital tablet [16]. Note that the samples in this dataset are given as the smoothed derivative of location. This makes the letters

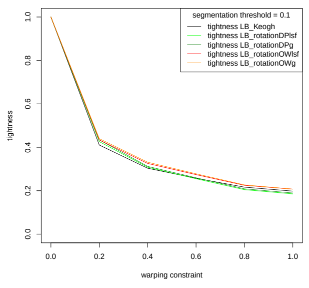
[1] We plan to publish all the source code via CRAN and on our web pages.



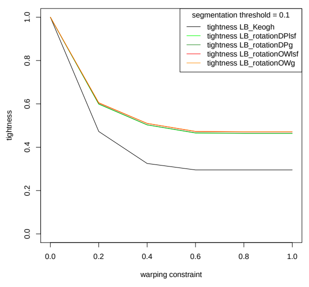
(a) Prague



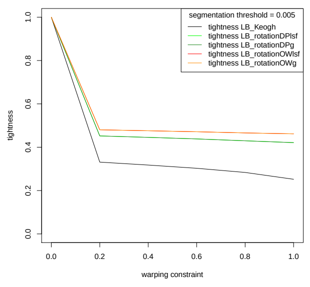
(b) San Francisco



(c) Roma



(d) Character



(e) Geolife

Figure 3 Tightness of $\mathrm{LB}_{\mathrm{Keogh}}$ and variants of $\mathrm{LB}_{\mathrm{Rotation}}$.

independet from the actual pen location on the tablet. For our experiments, we calculate the cumulative sums of these discrete derivatives and perform the matching of characters on their actual shape. Note that by this procedure, all characters start at the zero point.

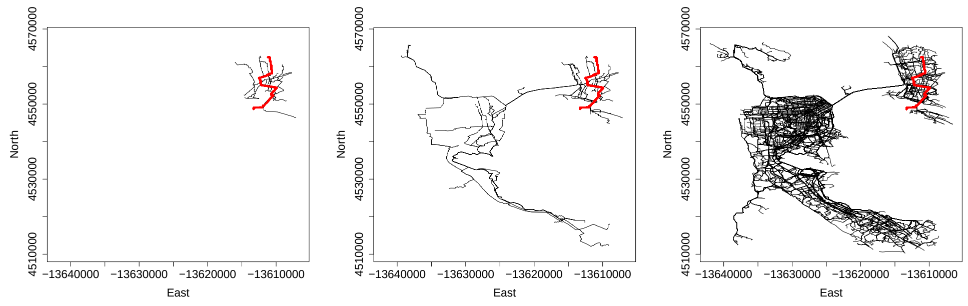
4.1.3 Prague Ego-Shooter Dataset

The Prague Ego-Shooter Dataset is a dataset containing 50 minutes of five players playing Urban Terror in Capture and Hold Mode on the map Prague (ut4_prague) [25]. In this game mode, players of two teams start in two different areas and the goal of the game is to "hold" the flag. The flag is positioned at a third fixed location on the map and the team who was the last walking over the flag is "holding" the flag. The team scores, who is holding the flag at specific time points. This map data has a lot of internal structure including the three locations and other tactically important spots around the map.

The dataset contains 275 trajectories for a total of 244,675 samples of player locations taken every 50ms. The coordinate system is similar to an orthogonal coordinate system with a unit of meters.

4.1.4 Roma Taxi Dataset

The Roma Taxi Dataset contains trajectories of 315 taxi drivers working in the center of Rome [5]. The traces contain the position of the taxis roughly every 7 seconds. The dataset was acquired with various Android tablets and is given in WGS84 coordinates. Note that the getAccuracy method of the Android location API has been used to reject positions for which this API method estimates a precision worse than 20m. In other words, only good signal situations have been kept. The dataset contains 21,743,005 points.



(a) First 10 trajectories **(b)** First 25 trajectories **(c)** All trajectories

Figure 4 A reordering of the San Francisco dataset.

4.1.5 Fastest Paths in San Francisco Dataset

The Fastest Paths in San Francisco dataset is a synthetic dataset we created from a street network of greater San Francisco area [28]. The street network was extracted from OpenStreetMap data and used in the ACM SIGSPATIAL GIS Cup 2015 [27]. The dataset provides the coordinates of nodes in the street network as modelled in the OpenStreetMap and calculates the fastest way between two random nodes using a weighting based on speed annotations in OpenStreetMap. It contains 539 trajectories with 136,978 points. Trajectory length ranges from 4 to 681 with a mean length of 254.1 points per trajectory.

4.2 Experiments

In this section, we present several experiments, we performed on the datasets. First, we evaluate tightness as a function of the warping constraint. Then, we look into pruning ratio and presorting.

4.2.1 Tightness

First, we calculate average tightness of five lower bounds: LB_{Keogh} and four version of LB_{Rotation}, namely for Douglas Peucker and Opening Window together with least square fitting and simple segment fitting. Figure 3 depicts the results for the five datasets Prague, Roma, San Francisco, Character and Geolife. We conclude that all variants of LB_{Rotation} perform similar, however, we also note that the gain of LB_{Rotation} is quite small for Roma as you can see in Figure 3c. This is to be expected as the trajectories of the Roma dataset are quite long (2,994 km on average; each trajectory is the complete trace of a Taxi driver during a month) and similar with each other. Interestingly, the GeoLife dataset shows nearly identical tightness for both Douglas Peucker variants and significantly better tightness for the Opening Window algorithm. This is particularly interesting as the Douglas Peucker algorithm is often used as a reference for segmentation algorithms due to its mathematical and perceptive quality and this can be seen as a warning that these two properties are not always deciding.

In summary, we conclude that LB_{Rotation} can be calculated more efficiently without loss of tightness by using the segment start and end points instead of fitting a linear model. Additionally, we conclude that an online segmentation such as Opening Window is applicable. We remark, however, that the quality of the segmentation has a severe impact on LB_{Rotation}

Table 1 Pruning Ratio and Speedup.

Dataset	Strategy	Average Pruning Ratio with Strategy	Average Pruning Ratio with Random Ordering	Speedup
San Fransisco A	Centroid	0.9925651	0.9849442	2.03
San Fransisco A	Geohash-Jaccard	0.8461538	0.8038462	1.27
San Fransisco A	Last Point	0.9925373	0.9826493	2.32
San Fransisco B	Centroid	0.9944238	0.9830855	3.03
San Fransisco B	Geohash-Jaccard	0.9868421	0.9236842	5.8
San Fransisco B	Last Point	0.9962687	0.9809701	5.1
Roma Taxi Dataset	Centroid	0.4493631	0.4407643	1.02
Roma Taxi Dataset	Geohash-Jaccard	0.5306122	0.4938776	1.08
Roma Taxi Dataset	Last Point	0.4423567	0.4417197	1.0011
Prague Dataset	Centroid	0.9817518	0.970073	1.64
Prague Dataset	Last Point	0.9817518	0.9689781	1.7
Character Trajectories	Centroid	0.8989899	0.8484848	1.5
Character Trajectories	Last Point	0.9292929	0.8565657	2.0
Geolife Dataset	Centroid	0.9985719	0.9985438	1.02
Geolife Dataset	Geohash-Jaccard	0.998602	0.9984555	1.11
Geolife Dataset	Last Point	0.9836735	0.98322	1.03

and that very long trajectories such as those of the Roma dataset degrade the usefulness of all variants of $\mathrm{LB}_{\mathrm{Rotation}}$ altogether.

4.2.2 Pruning Ratio and Presorting

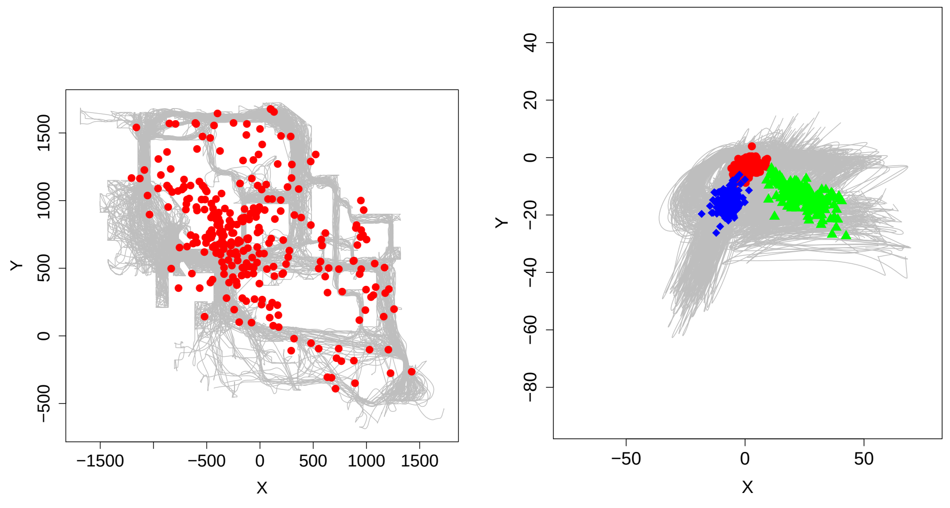
For evaluating the presorting strategies, we select a query object in each dataset and calculate ten reorderings of the dataset in which each time the ten nearest-neighbors have been brought forward.

Figure 4 illustrates this approach for the San Francisco dataset. In this Figure, the trajectory centroid was used as a presorting strategy and we select the ten nearest neighbors with respect to the centroids in order to bootstrap similarity search from them (see Figure 4a). After that, as you can see in Figure 4b, trajectories arrive in the ordering they are stored on disk in order to reduce the amount of random access to the dataset. In Figure 4c, one can see the full dataset.

Additionally, we calculate ten fully random orderings of the same dataset. In any case, the query was removed from the dataset for a more realistic setting.

In general, any of the presorting strategies showed a good increase in pruning power. Table 1 gives results on the datasets.

For the San Francisco dataset, two queries have been used. One on the continent (San Francisco A) and one on the island (San Francisco B). The probability of finding a good example by chance is larger for the island as the dataset contains only a small part of the continent (see Figure 4). This can be seen in the pruning ratio of random orderings. On average, it is easier to find a good candidate for the island case. This is to be expected, as the dataset has a bias towards more trajectories on the island due to the fact that more graph nodes have been modelled on the island. With respect to the pruning strategies, San Francisco shows very good centroid and last point performance, but only moderate Geohash set similarity performance. This is to be expected as the trajectories are shortest paths and the centroid as well as the last point of a shortest path is a good summary of the path. Furthermore, the trajectories are relatively short and, hence, the centroid is a good summary



(a) All Centroids **(b)** Letters “a”, “p”, “z”

Figure 5 Centroid Distribution of (a) Prague Ego-Shooter Dataset and (b) Character Trajectories Dataset.

of the trajectory. Anyways, the tightness of all variants is quite good (only few percents of the dataset are actually analyzed with full DTW calculations). The speedup illustrates the number of DTW calculations, that have to be used without presorting. It is calculated as follows:

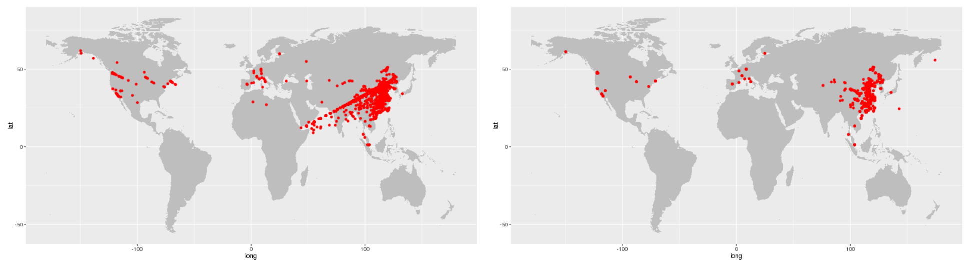
$$\text{Speedup} = \frac{1 - \text{Pruning Ratio with Presorting}}{1 - \text{Pruning Ratio without Presorting}}.$$

With an average speedup of 3.25, the classical approach needs 3.25 times as many DTW calculations as the presorting algorithms.

The Roma dataset contains long trajectories of taxis driving around the center area of Rome. Due to the taxi nature of the trajectories, the centroids are not very informative, as they are inside the center of Rome for all trajectories.

Therefore, the speedup of the point-based strategies is quite poor. The Jaccard distance of Geohash sets strategy, however, performs better. This is due to the fact that the center of Rome, which is part of all trajectories, does not contribute to the dissimilarity of two trajectories and they are indexed by their set of extreme locations (i.e., those Geohash cells that are not in other trajectories). This results in a considerable speedup. Remember that this dataset is a hard dataset for lower bounds (see Figure 3c) due to the very long trajectories traveling around the city. This is expressed by the low general pruning ratio. In this situation, the speedup of 8% is to be understood relative to the large fraction of roughly half of the dataset, which has not been pruned. In this way, our pruning strategy prunes a large absolute number of DTW calculations in comparison to random orderings.

For the Prague Ego-Shooter dataset, the centroid information was quite useful and well-distributed. We expected this behavior as the game rules let players of both teams start in two isolated regions and they aim to quickly reach and defend the flag. But as teams can only score on specific points in time, players tend to look for a safe place on the map and wait for the right moment for a team attack. Figure 5a depicts the distribution of centroids on the Prague Ego-Shooter Dataset.



(a) Centroids **(b)** Last Point

Figure 6 Geolife Point Summary Distributions.

The Character Trajectories dataset is another example of presorting as a powerful tool if the summary is carefully selected. Recall that we integrate the Character dataset such that the shapes of the characters are used (the dataset contains the first derivative of the location and is often used in this modality). By the integration process, all our character strokes start at coordinate $(0, 0)$. Hence, the centroid partly covers the shape of the letters. This leads to a considerable speedup of 1.5 for the centroid strategy. This means, that without presorting, we would have to calculate 1.5 times as many true DTW as with the simple presorting strategy. Figure 5b illustrates examples of the centroid behaviour on the character dataset for all samples of the letters “a”,“p”, and “z”. You can clearly see that the centroids of those characters form separated clusters.

The Geolife dataset behaves similar to the Roma dataset, however, the pruning power is much higher. This is due to the fact that the Geolife dataset is split into many smaller trajectories. Again, and similar to Roma, the centroid does not cover enough information while the Geohash Jaccard distance is able to ignore the common parts of trajectories and concentrate on significantly different locations. Figure 6a depicts the distribution of centroids for this dataset, which is clearly centered on Beijing. Similarly, the distribution of last points of trajectories is also not very informative as depicted in Figure 6b.

In order to illustrate the power of using Geohash Jaccard distance, we provide an embedding based on multidimensional scaling (MDS) for a subset of 400 trajectories of Geolife with respect to the Jaccard distance. In multidimensional scaling, a distance matrix is given and MDS tries to find a set of points such that the relative distance between points is preserved [17].

Figure 7 depicts the pairwise Geohash distances embedded into two-dimensional space as good as possible. Therefore, we employed the SMACOF algorithm minimizing the metric stress

$$\sigma(X) = \sum_{i<j\leq n} w_{ij}(d_{ij}(X) - \delta_{ij})^2$$

between the Geohash distances δ_{ij} and the Euclidean distances $d_{ij}(\mathrm{X})$ in the two-dimensional embedding X. The weights for the individual terms w_{ij} have all been set to one as there is no uncertainty in the distances in our case. We added some jitter to the plot just to make the number of identical points visible. You can clearly see that the points are distributed around the model space without much clustering structure. So this summaries created from Jaccard distance of Geohash cells actually contain useful information for preponing candidates explaining the very good presorting power in San Francisco B, Roma, and Geolife.

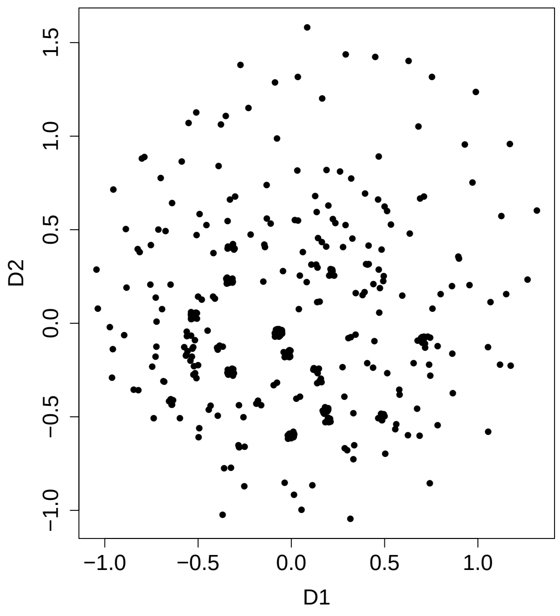


Figure 7 Metric Embedding of 400 Geolife trajectories with respect to Jaccard distance of Geohash sets.

5 Conclusion

With this paper, we have thoroughly evaluated the novel lower bound for dynamic time warping called LB_{Rotation} on several datasets. Additionally, we show that it can be extended to an online algorithm without severe degradation in performance. Third, we showed that a presorting strategy with simple summaries of trajectories such as the last point, the centroid, or the set of visited spatial cells creates better pruning power, sometimes considerably, sometimes marginally.

The most important distinction in this context is the question, how a trajectory can be aggregated in a useful way as a single object. One way to do that is calculating some aggregate. The last point strategy and the centroid strategy are examples for that. The centroid is clearly an “average” as it can be calculated as the mean of the coordinates. The last point also describes the average movement, especially, for datasets where all trajectories start at the same point or are normalized to do so. Another way of presorting is describing the extreme behavior of a single trajectory compared to others.

We have done so – to some extent – by measuring the dissimilarity of the sets of Geohash cells each trajectory hits. Future work could include other ways of extracting the most extreme behavior from trajectories, such as extracting features from the convex hull or performing archetypal analysis [3]. Additionally, ensembles of summaries can be constructed and evaluated. However, the most interesting problem in this area would be, how to select and configure summaries in order to gain the highest pruning power from presorting for similarity search.

References

1 Assaf Almog, Avi Levi, and Alfred M. Bruckstein. Spatial de-interlacing using dynamic time warping. In *IEEE International Conference on Image Processing*, volume 2, pages 1006–1010. IEEE, 2005.

2 Anthony Bagnall, Jason Lines, Jon Hills, and Aaron Bostrom. Time-series classification with COTE: the collective of transformation-based ensembles. *IEEE Transactions on Knowledge and Data Engineering*, 27(9):2522–2535, 2015.

3 Christian Bauckhage and Christian Thurau. Making archetypal analysis practical. In *Pattern Recognition*, pages 272–281. Springer, 2009.

4 Donald J. Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In *KDD workshop*, pages 359–370. Seattle, WA, 1994.

5 Lorenzo Bracciale, Marco Bonola, Pierpaolo Loreti, Giuseppe Bianchi, Raul Amici, and Antonello Rabuffi. CRAWDAD dataset roma/taxi (v. 2014-07-17). Downloaded from `http://crawdad.org/roma/taxi/20140717`, July 2014. `doi:10.15783/C7QC7M`.

6 Won-Du Chang and Jungpil Shin. Modified dynamic time warping for stroke-based on-line signature verification. In *Ninth International Conference on Document Analysis and Recognition (ICDAR)*, volume 2, pages 724–728. IEEE, 2007.

7 Hui Ding, Goce Trajcevski, Peter Scheuermann, Xiaoyue Wang, and Eamonn Keogh. Querying and mining of time series data: experimental comparison of representations and distance measures. *Proceedings of the VLDB Endowment*, 1(2):1542–1552, 2008.

8 David H. Douglas and Thomas K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10(2):112–122, 1973.

9 Fosca Giannotti and Dino Pedreschi. *Mobility, data mining and privacy: Geographic knowledge discovery*. Springer Science & Business Media, 2008.

10 Xudong Gong, Yan Xiong, Wenchao Huang, Lei Chen, Qiwei Lu, and Yiqing Hu. Fast similarity search of multi-dimensional time series via segment rotation. In *Database Systems for Advanced Applications*, pages 108–124. Springer, 2015.

11 Fumitada Itakura. Minimum prediction residual principle applied to speech recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 23(1):67–72, 1975.

12 Eamonn Keogh and Chotirat Ann Ratanamahatana. Exact indexing of dynamic time warping. *Knowledge and information systems*, 7(3):358–386, 2005.

13 Sang-Wook Kim, Sanghyun Park, and Wesley W. Chu. An index-based approach for similarity search supporting time warping in large sequence databases. In *Proceedings of the 17th International Conference on Data Engineering*, pages 607–614. IEEE, 2001.

14 Benoit Legrand, C. S. Chang, S. H. Ong, Soek-Ying Neo, and Nallasivam Palanisamy. Chromosome classification using dynamic time warping. *Pattern Recognition Letters*, 29(3):215–222, 2008.

15 Daniel Lemire. Faster retrieval with a two-pass dynamic-time-warping lower bound. *Pattern recognition*, 42(9):2169–2180, 2009.

16 M. Lichman. UCI machine learning repository. `http://archive.ics.uci.edu/ml`, 2013.

17 Patrick Mair, Jan de Leeuw, and Patrick J. F. Groenen. Multidimensional Scaling in R: SMACOF. URL: `https://cran.r-project.org/web/packages/smacof/vignettes/smacof.pdf`.

18 Andrés Marzal, Vicente Palazón, and Guillermo Peris. Contour-based shape retrieval using dynamic time warping. In *Current Topics in Artificial Intelligence*, pages 190–199. Springer, 2005.

19 Gustavo Niemeyer. Geohash. `http://geohash.org/`, 2008.

20 Toni M. Rath and R. Manmatha. Lower-bounding of dynamic time warping distances for multivariate time series. Available at `http://works.bepress.com/r_manmatha/19/`, 2003.

21 Hiroaki Sakoe and Seibi Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 26(1):43–49, 1978.

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

23 V. M. Velichko and N. G. Zagoruyko. Automatic recognition of 200 words. *International Journal of Man-Machine Studies*, 2(3):223–234, 1970.

24 Haozhou Wang, Han Su, Kai Zheng, Shazia Sadiq, and Xiaofang Zhou. An effectiveness study on trajectory similarity measures. In *Proceedings of the Twenty-Fourth Australasian Database Conference-Volume 137*, pages 13–22. Australian Computer Society, Inc., 2013.

25 Martin Werner. Dataset containing trajectories of four players playing one hour of Urban Terror Capture and Hold on the map Prague. Available at `http://trajectorycomputing.com/datasets/prague-ego-shooter-dataset/`, 2014.

26 Martin Werner. BACR: Set Similarities with Lower Bounds and Application to Spatial Trajectories. In *23rd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL 2015)*, 2015.

27 Martin Werner. GISCUP 2015: Notes on Routing with Polygonal Constraints. In *SIGSPATIAL GIS CUP 15, in conjunction with 23rd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL 2015)*, 2015. URL: `http://www.martinwerner.de/preprint/GISCUP.pdf`.

28 Martin Werner. Dataset containing 500 random shortest paths in greater San Francisco Area. Available at `http://www.martinwerner.de/files/fastest_sanfrancisco.tgz`, 2016.

29 Byoung-Kee Yi, H. V. Jagadish, and Christos Faloutsos. Efficient retrieval of similar time sequences under time warping. In *Proceedings of the 14th International Conference on Data Engineering*, pages 201–208. IEEE, 1998.

30 Chun-Ting Zhang, Ren Zhang, and Hong-Yu Ou. The Z curve database: a graphic representation of genome sequences. *Bioinformatics*, 19(5):593–599, 2003.

31 Yu Zheng, Quannan Li, Yukun Chen, Xing Xie, and Wei-Ying Ma. Understanding mobility based on GPS data. In *Proceedings of the 10th international conference on Ubiquitous computing*, pages 312–321. ACM, 2008.

32 Yu Zheng, Xing Xie, and Wei-Ying Ma. GeoLife: A Collaborative Social Networking Service among User, Location and Trajectory. *IEEE Data Engineering Bulletin*, 33(2):32–39, 2010.

33 Yu Zheng and Xiaofang Zhou. *Computing with spatial trajectories.* Springer Science & Business Media, 2011.