Optimized Discrete Fréchet Distance between trajectories

Thomas Devogele
University of Tours, Computer Science Laboratory, Tours
64 avenue Jean Portalis
Tours 37200, France
thomas.devogele@univ-tours.fr

Laurent Etienne
University of Tours, Computer Science Laboratory, Tours
64 avenue Jean Portalis
Tours 37200, France
laurent.etienne@univ-tours.fr

Maxence Esnault
University of Tours, Computer Science Laboratory, Tours
64 avenue Jean Portalis
Tours 37200, France
maxence.esnault@etu.univ-tours.fr

Florian Lardy
University of Tours, Computer Science Laboratory, Tours
64 avenue Jean Portalis
Tours 37200, France
florian.lardy@etu.univ-tours.fr

ABSTRACT

Distance computation between polylines or trajectories is a key point to assess similarity between geometrical objects. This paper describes a new optimized algorithm to compute discrete Fréchet distance which aims to lower computation time and improve precision. This algorithm is applied to GPS trajectories. It includes a filtering, pruning and an enhancement process. Thanks to this algorithm, big data trajectory repositories can be mined. This process is validated on a large trajectory dataset.

CCS CONCEPTS

• **Information systems → Geographic information systems**; **Similarity measures**; *Clustering*; Global positioning systems;

KEYWORDS

Fréchet distance, linear distance, similarity measure, GPS dataset, filtering process, trajectory mining

ACM Reference format:
Thomas Devogele, Laurent Etienne, Maxence Esnault, and Florian Lardy. 2017. Optimized Discrete Fréchet Distance between trajectories. In *Proceedings of 6th ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data, Redondo Beach, California, USA, November 2017 (BigSpatial-2017),* 9 pages.
https://doi.org/10.1145/3150919.3150924

1 INTRODUCTION

GPS sensors are helpful to track, analyze and understand moving objects behavior. Theses sensors are nowadays widely embedded into smart-phones, watches, vehicles and data loggers. They produces real time position data flows that can be stored and mined to unveil movement behavior. Thanks to the increasing availability of GPS traces, it is now possible to infer patterns or trends to analyze the behavior of a large set of moving objects. These patterns are very useful to model the behavior of various moving object (people, vehicles, animals or natural events...). Researchers from different fields such as ecology, social science, geography, computer science exploit trajectories. analyzing Human mobility is a key component to understand social networks [2, 26] or to provide travel recommendation [29]. GPS-equipped vehicles give interesting information to plan for resources allocation [25], analyze traffic [24], improve transportation networks [28] and detect outliers [8]. Biologists have been collecting the animals' trajectories to model migration, behavior and habitat [14, 15]. Meteorologists, environmentalists, climatologists and oceanographers are collecting trajectories of natural phenomena, such as hurricanes, tornados, and ocean currents [14, 19]. Thanks to historical trajectory dataset, scientists can capture the climate change. Comprehensive survey [16, 17, 27] thoroughly explores the field of trajectory data mining and its applications. All these collected positions generate big trajectory dataset which require new algorithms optimized for big data.

Many of these applications require to compare and measure similarity between trajectories. Linear distances are effective to measure the similarity between moving objects. This similarity measure can also be employed to compare the trajectories of two different mobile objects or one object over time. However, linear distances highly depends on the collected data features as the number of positions recorded (frequency) and the accuracy of these positions.

Common GPS systems record positions (longitude, latitude and time) every second with a standard accuracy ranging from 1 to 15 meters depending on context and receivers [10]. Accuracy of a positioning system can be measured using a precise known reference position and a set of positions recorded over time. The accuracy is then defined as the 95% (the 95 percentile) of the positioning error in meters measured over time. The accuracy of the different algorithms presented in this article will be measured using the same methodology (95 percentile of the measured errors).

Linear distances can be classified into three different measures. The first one is the minimal distance which returns the distance between the two spatially closest points of two lines. Unfortunately, this measure is not effective to evaluate the similarity between two lines. Indeed, two lines can be very different even if they intersect each other.

Average distance is another linear distance which preform well when the two lines to compare have the same length. Dynamic time warping: DTW [21] is a well known algorithm for measuring similarity between time series. This measure calculates an optimal alignment between two given sequences. The result is the minimal sum of the distances between corresponding points. The time complexity of the original algorithm is $O(nm)$. Some improvement as FastDTW [22] proposes an approximation of DTW that has a linear time and space complexity. Unfortunately, DTW can not be applied on large heterogeneous data sets. Indeed, the sum function is not an good indicator for lines having a very different number of points (long line compared to short one).

Finally, the maximum distance is usually employed to calculate the biggest gap between lines. Hausdorff and Fréchet defined different way to assess this maximal distance.

The Hausdorff distance [1] does not consider lines as an order set of points. The maximum distance is then selected among spatially closest points instead of homologous points. This drawback implies that two lines can have a small Hausdorff distance, without being similar.

On the other hand, the Fréchet distance [1] is a similarity measure between lines which takes into account the location and ordering of the points along lines. This measure performs well on trajectories having different sizes and loops. Unfortunately, the time complexity of this algorithm is $O(nmlog(nm))$ where n and m are the length of the two respective lines to compare [1]. A discrete version was proposed by [7] to reduce the complexity to $O(nm)$. However, the accuracy of this measure is also impacted by this algorithm.

In this article, a new optimized discrete Fréchet distance is proposed. This distance metric aims to fulfill two properties: efficacy (accurate measures) and efficiency for large GPS dataset (low computation time). Unfortunately, accuracy of this measure is bound to the computation time as detailed in section 3. Nevertheless, quality of the measure have to fit with GPS device accuracy. This algorithm is meant to handle big datasets and require lower computation time. The process was optimized for similar trajectories. Indeed, to detect dissimilar linear objects, bounding box or epsilon band approach are more efficient and faster than distance approaches.

The remainder of this article is organized as follows. Section 2 presents regular discrete Fréchet distance. In section 3, several optimizations are introduced. To validate this new measure, the optimized discrete Fréchet distance is applied to a big trajectory datasets. In section 4, the impact of the different algorithm parameters are detailed. Section 5 concludes this paper.

2 DISCRETE FRÉCHET DISTANCE

The optimized measure presented in this paper relies on the Fréchet distance. A usual illustration of Fréchet distance is the example of a man walking a dog using a leash. The man moves on one math (L_1) and the dog moves on another path (L_2). Both the man and dog are allowed to move on their corresponding path at varying speeds, but they can not walk backward. The Fréchet distance between the two paths is the shortest leash required to connect the dog and the man walking together.

Definition 2.1. The Fréchet distance is the maximal distance between two oriented lines. Each oriented line is equivalent to a continuous function $f : [a,a'] \rightarrow V(g : [b,b'] \rightarrow V)$ where $(a,a',b,b' \in \mathcal{R}, (a < a'), (b < b')$ and (V,d) is a metric space. Then d_F denotes their Fréchet distance defined in equation 1.

$$d_F(f,g) = \underset{\substack{\alpha:[0,1]\rightarrow[a,a'] \\ \beta:[0,1]\rightarrow[b,b']}}{Inf} \; \underset{t\in[0,1]}{Max} \; (d(f(\alpha(t)), g(\beta(t)))) \tag{1}$$

Eiter and Mannila [7] give a good approximation of the Fréchet distance named the discrete Fréchet distance (d_{dF}). The discrete Fréchet distance between two polylines can be computed in $O(nm)$ time. Polylignes (L_1 and L_2) are defined as ordered sets of n vertexes $< L_{1.0}...L_{1.n-1} >$ and m vertexes $< L_{2.0}...L_{2.m-1} >$. The d_{dF} is calculated using dynamic programming to fill two matrices.

(1) First of all, the distance matrix (DM) is computed. The value of each cell $DM_{i,j}$ is the distance between vertexes $L_{1.i}$ and $L_{2.j}$. In this article, the distance between vertexes is the Euclidean distance (d_E). For GPS data using latitude and longitude values, Haversine distance (d_H) is computed.
(2) Then, the Fréchet matrix (FM) is filled up iteratively to compute the discrete Fréchet distance [7]. To compute $FM_{i,j}$, formula 2 is applied. In the discrete version of the algorithm, only vertexes of polylines are taken into account. The discrete Fréchet distance (d_{dF}) is held in the last cell of the matrix ($FM_{n-1,m-1}$).

$$FM_{i,j} = Max(DM_{i,j}, Min(FM_{i-1,j}, FM_{i,j-1}, FM_{i-1,j-1})) \tag{2}$$

To illustrate this approach, the example of figure 1 is introduced. The example line 1 has $n = 19$ vertexes and the line 2 has $m = 12$ vertexes. Distance and Fréchet matrices have 228 (19x12) cells each. These matrices were completely filled up using equation 2.

The discrete Fréchet distance between these two lines is represented by the green dashed line in figure 1. The computed value of d_{dF} for this example is 1.612m. This maximum distance between the two lines corresponds to the distance between vertexes $L_{1.13}$ and $L_{2.9}$. Matrices are too large to be detailed in this article. Smaller matrices are described in next section (figure 2).

The discrete Fréchet distance is also adapted to complex trajectories with loops [1] or to compute partial distance between one full trajectory and an incomplete one [5].

3 OPTIMIZATIONS

Three different optimizations are now introduced to improve lower the required CPU time and get a balanced accuracy.

3.1 Filtering process

The first optimization focuses on the number of vertexes of each trajectory. The recording process of GPS traces produces large datasets. The number of vertexes is a key factor driving the accuracy of the measure. Two highly sampled trajectories with an important numbers of vertexes (n and m) will produce an accurate d_{dF} distance. However, the required CPU time to compute this distance is directly bound to these numbers as the complexity of the d_{dF} distance computation is $O(nm)$.

L_1 num	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
x	1.1	2.0	3.0	4.0	5.0	6.0	7.0	8.0	9.0	10.0	11.0	12.0	13.2	14.2	15.0	16.0	17.0	18.0	19.0
y	1.0	1.1	1.2	1.3	1.2	1.1	0.9	1.1	1.2	1.3	1.4	1.5	1.6	1.0	1.0	1.1	1.2	1.0	1.0

L_2 num	0	1	2	3	4	5	6	7	8	9	10	11
X	1	2.6	4.2	5.8	7.4	9	11	12	14	15	17	19
Y	1.6	1.7	2	2.1	2	1.8	2.1	2.4	2.6	2.4	2.2	2

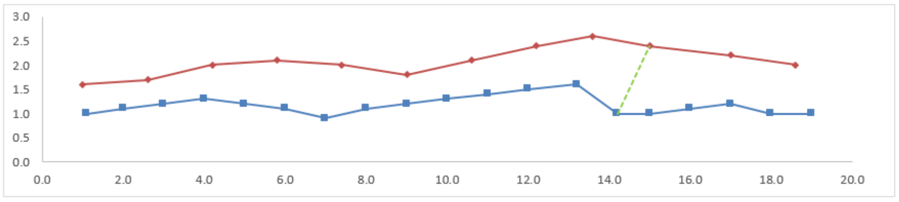


Figure 1: Two example lines and discrete Fréchet distance

The following part of the process aims to reduce the number of vertexes to lower the CPU time required to compute the d_{dF} distance and analyze the impact on the d_{dF} distance.

If a trajectory is smooth and straight, many vertexes can be filtered out. Several articles describe online data reduction processes [4, 13, 18]. These processes determine whether a newly acquired spatial position should be retained in a trajectory and recorded or filtered out. Only few systems implement this kind of data reduction processes. For example, some Garmin GPS systems propose two modes for position recording: every second or smart recording.

Offline trajectory filtering process consider the full trajectory. According to Shi and Cheung [23] among the different line simplification algorithms, the Douglas-Peucker algorithm [6] is the most accurate with a minimum value of maximum dissimilarity. This algorithm is used to approximate the original trajectory by a subset of its positions. User defines a tolerance distance value (ε). The bigger ε is, the smaller number of vertexes of result is.

The initial approximate trajectory is the straight line from the first position to the last one. The position contributing the biggest error (distance between original vertexes and the approximate trajectory) is kept into the filtered polyline. For this example, the perpendicular Euclidean distance projected on the simplified line segment is used to calculate the error. The algorithm recursively divides the approximate line in two segments according to this new position. For each part, new position is added if the error is superior to ε. For a trajectory having n positions, the expected complexity of the Douglas-Peucker algorithm is $O(nlogn)$.

Figure 2 gives the results of the filtering process applied to the example trajectories of Figure 1 with $\varepsilon = 0.2m$ and the two matrices DM and FM. The number of vertexes decreases (from 19 to 6 for L_1 and from 12 to 5 for L_2). The d_{dF} between these two filtered trajectory is 2.193 m and represents the distance between $L_{1.6}$ and $L_{2.5}$ (dashed green line in Figure 2).

As the number of vertexes of L_1 and L_2 were reduced, the size of the matrices shrink from 228 cells to 30 cells (-86.84%). The CPU time required to compute these matrices is then lower. However, the filtering process induced a decrease in the distance accuracy. The vertexes associated with d_{dF} changed as the initial vertexes

$f_{0.2}DM$

L1 num	x	y	L2 num: 0	3	5	8	11
		X	1	5.8	9	13.6	18.6
		Y	1.6	2.1	1.8	2.6	2
0	1.1	1.0	0.608	4.827	7.940	12.602	17.529
3	4.0	1.3	3.015	1.970	5.025	9.688	14.617
6	7.0	0.9	6.041	1.697	**2.193**	6.815	11.652
12	13.2	1.6	12.200	7.417	4.205	1.077	5.415
13	14.2	1.0	13.214	8.472	5.261	1.709	4.512
18	19.0	1.0	18.010	13.246	10.032	5.632	1.077

$f_{0.2}FM$

L1 num	x	y	L2 num: 0	3	5	8	11
		X	1	5.8	9	13.6	18.6
		Y	1.6	2.1	1.8	2.6	2
0	1.1	1.0	0.608	4.827	7.940	12.602	17.529
3	4.0	1.3	3.015	1.970	5.025	9.688	14.617
6	7.0	0.9	6.041	1.970	2.193	6.815	11.652
12	13.2	1.6	12.200	7.417	4.205	2.193	5.415
13	14.2	1.0	13.214	8.472	5.261	2.193	4.512
18	19.0	1.0	18.010	13.246	10.032	5.632	**2.193**

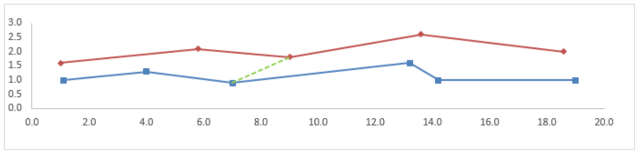
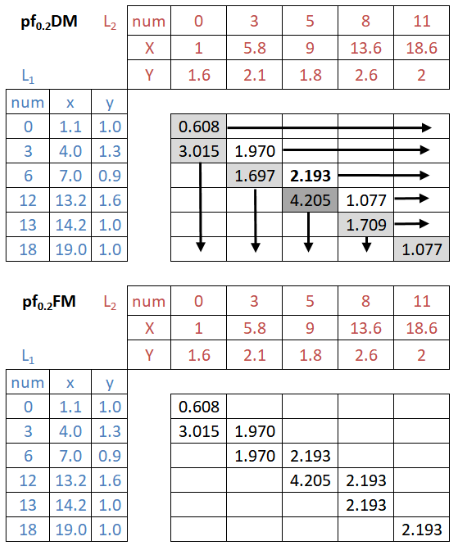


Figure 2: The two lines after the filtering process and the discrete Fréchet distance

were filtered out by the simplification process. The d_{dF} raised from 1.612 m to 2.193 m (+36.04%).

3.2 Partial computation of matrix

The second optimization focuses on limiting the number of computations required to build up the of two matrices. The values of some cells are not required to compute the Fréchet distance. Indeed, only cells close to the main diagonal should be considered. For DTW, Sakoe and Chiba [21] and Itakura [11] already propose to focus only on a subset of the matrices called the warping window. This subset is a band [21] or a parallelogram [11]. User needs to define the width of these windows. For d_{dF} the proposed idea is the same but it does not require the user to set up a parameter. In a first time, cell values from a potential interesting way from the start vertexes


pf$_{0.2}$DM
L$_2$
L$_1$

L$_2$					
num	0	3	5	8	11
X	1	5.8	9	13.6	18.6
Y	1.6	2.1	1.8	2.6	2

num	x	y					
0	1.1	1.0	0.608	→	→	→	→
3	4.0	1.3	3.015	1.970	→	→	→
6	7.0	0.9	↓	1.697	2.193	→	→
12	13.2	1.6	↓	↓	4.205	1.077	→
13	14.2	1.0	↓	↓	↓	1.709	→
18	19.0	1.0	↓	↓	↓	↓	1.077

pf$_{0.2}$FM
L$_2$
L$_1$

L$_2$					
num	0	3	5	8	11
X	1	5.8	9	13.6	18.6
Y	1.6	2.1	1.8	2.6	2

num	x	y					
0	1.1	1.0	0.608				
3	4.0	1.3	3.015	1.970			
6	7.0	0.9		1.970	2.193		
12	13.2	1.6			4.205	2.193	
13	14.2	1.0				2.193	
18	19.0	1.0					2.193

Figure 3: Matrices with almost diagonal and pruned cells

couple $(L_{1.0}, L_{2.0})$ to the end vertexes $(L_{1.n}, L_{2.m})$ are computed. For square matrices, the main diagonal is selected. For non square matrices, an "almost diagonal" is defined.

If dimension n (number of lines) is superior to dimension m (number of columns), the quotient (q) and the reminder (r) of n divided by m is used to defined this "almost diagonal". The first q lines are associated with $r+1$ columns. The next lines are associated with r columns. For example, for a matrix with 4 lines and 6 columns. The quotient (q) equals 1 and the remainder (r) equals 2. So, the almost diagonal is $[0,0], [0,1], [1,2], [1,3], [2,4], [3,5]$. If dimension m is superior to dimension n, dimensions are inverted. Distances between vertexes on this almost diagonal are computed and the maximal value is returned as a temporary value of the leash. This value is called $Diag_{Max}$. Indeed, one way from the vertexes couple $(L_1.0, L_2.0)$ to the vertexes couple $(L_1.n, L_2.m)$ is known and the length of this leash is $Diag_{Max}$. This value can then be used to prune the matrix computation as soon as one cell on a row or column of the matrix is greater than $Diag_{Max}$.

For our example, the almost diagonal cells (gray colored in figure 3) are initially computed. The $Diag_{Max}$ value (dark gray cell) equals 4.205 m. In a second time, the left down part and the right up part of the matrix are computed. For the right up part, for each line, distances between vertexes are computed until one value is superior to $Diag_{Max}$. For the first line, the value of the first candidate cell of DM is 4.827 m (see figure 2). As this value is greater than $Diag_{Max}$, the process is pruned and no other cell of this row are computed. For the second line, the value of the first candidate cell is 1.970 m. This value below $Diag_{Max}$ is stored in the pDM (Figure 3). The next one (5.025) is above, so the process is pruned for this line. As shown in Figure 3, only two other cells are recorded. To stop the process a second condition must be reached. Indeed, if for line i, the process stop at column j, then, for line $i+1$, the computing process can not stop before column j. This condition is required as the next value can be below $Diag_{Max}$ and the minimum distance path could be at the right of this cell. The algorithm 1 summarize this process to compute the optimized partial Distance Matrix pDM with $n >= m$.

Algorithm 1 Optimized Distance Matrix computation

```
1: procedure COMPUTE DISTANCE MATRIX($L_1, L_2$)
2:     $i \leftarrow 0$ ; $j \leftarrow 0$ ; $Diag_{Max} \leftarrow 0$
3:     $n \leftarrow Length(L_1)$ ; $m \leftarrow Length(L_2)$ where $n >= m$
4:     $q \leftarrow n/m$ ; $r \leftarrow n\%m$
5:     $Diag_{Max} \leftarrow 0$
6:     for $i \leftarrow 0$ to $(r*(q+1))$ do                ▷ Almost diagonal
7:         $j \leftarrow (i/(q+1))$
8:         $DM_{i,j} \leftarrow d_E(L1.i, L2.j)$
9:         if $DM_{i,j} > Diag_{Max}$ then
10:            $Diag_{Max} \leftarrow DM_{i,j}$
11:    for $i \leftarrow (r*(q+1))$ to $n-1$ do              ▷ End of diagonal
12:        $j \leftarrow ((i-r)/q)$
13:        $d \leftarrow d(L1.i, L2.j)$
14:        $DM_{i,j} \leftarrow d_E(L1.i, L2.j)$
15:        if $DM_{i,j} > Diag_{Max}$ then
16:            $Diag_{Max} \leftarrow DM_{i,j}$
17:    $j_{min} \leftarrow 0$
18:    for $i \leftarrow 0$ to $n-1$ do                      ▷ Upper right matrix
19:        $j \leftarrow i$
20:        repeat
21:            $d \leftarrow d(L1.i, L2.j)$
22:            if $d < Diag_{Max}$ then
23:                $DM_{i,j} \leftarrow d$
24:                $j \leftarrow j+1$
25:        until $(j \geq m \vee d > Diag_{Max}) \wedge (j \geq j_{min})$
26:        $j_{min} \leftarrow j$
27:    $i_{min} \leftarrow 0$
28:    for $j \leftarrow 0$ to $m-1$ do                      ▷ Lower left matrix
29:        $i \leftarrow j$
30:        repeat
31:            $d \leftarrow d(L1.i, L2.j)$
32:            if $d < Diag_{Max}$ then
33:                $DM_{i,j} \leftarrow d$
34:                $i \leftarrow i+1$
35:        until $(i \geq n \vee d > Diag_{Max}) \wedge (i \geq i_{min})$
36:        $i_{min} \leftarrow i$
37:    return $DM, Diag_{Max}$
```

In the same way, for the left down part of pDM, for each column, distances between vertexes are computed until one value is greater than $Diag_{Max}$. For this example all values of this part are greater than $Diag_{Max}$ so that all this part of the matrix is pruned. Pruned cells are set up with *null* value in the pDM matrix (see figure 3). We can notice that 21 cells out of 30 were pruned for this simple example. Finally, the partial Fréchet matrix pFM is calculated using only the non null cells of the distance matrix. This algorithm gives the same results than the regular discrete Fréchet distance algorithm.

The proposed algorithm to compute this optimized discrete Fréchet distance is more complicated than the regular one but its complexity is smaller. The complexity is proportional to the length of the "almost diagonal" $o(\sqrt{(n^2 + m^2)})$.

Indeed, k almost diagonals close to the almost diagonals are computed. Thanks to the $Diag_{max}$ pruning value, the width of the required cells to compute around the diagonal does not need to be set up by the user. To compute the partial Fréchet Matrix, only non null cells of pDM and pFM are considered. If one cell is null, it is discarded from the min and max computation. The dynamic programming formula used to compute the pFM is presented in equation 3.

$$pFM_{i,j} = \max(pDM_{i,j}; \min(pFM_{i-1,j}; pFM_{i,j-1}; pFM_{i-1,j-1})) \tag{3}$$

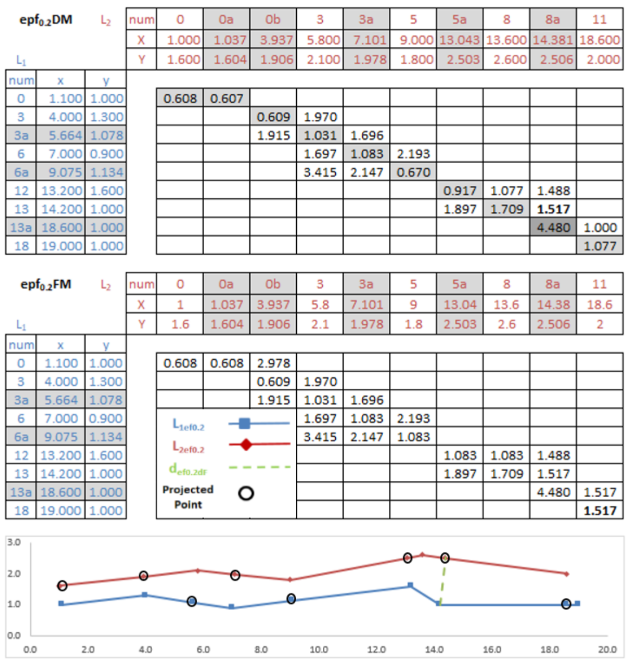
Algorithm 2 Optimized Frechet Matrix computation

```
1:  procedure COMPUTE FRECHET MATRIX($L_1, L_2$)
2:      $i \leftarrow 0$ ; $j \leftarrow 0$ ; $jmin \leftarrow 0$
3:      $n \leftarrow Length(L_1)$ ; $m \leftarrow Length(L_2)$
4:      $(DM, Diag_{Max}) \leftarrow ComputeDistanceMatrix(L_1, L_2)$
5:      for $i \leftarrow 0$ to $n - 1$ do
6:          $j \leftarrow jmin$
7:          while $DM_{i,j} = null$ do $j \leftarrow j + 1$
            $jmin \leftarrow j$
8:          while $DM_{i,j} \neq null \wedge j < m$ do $mini \leftarrow \infty$
9:              if $i > 0 \wedge j > 0 \wedge DM_{i-1,j-1} \neq null$ then
10:                 $mini \leftarrow FM_{i-1,j-1}$
11:             if $i > 0 \wedge DM_{i-1,j} \neq null$ then
12:                 $mini \leftarrow min(mini, FM_{i-1,j})$
13:             if $j > 0 \wedge DM_{i,j-1} \neq null$ then
14:                 $mini \leftarrow min(mini, FM_{i,j-1})$
15:             $FM_{i,j} \leftarrow max(mini, DM_{i,j})$
16:     return $FM$
```

3.3 Enhancement of trajectory

The d_{dF} accuracy is bounded by the maximum length of segments [7]. To improve the d_{dF}, some significant points of segments must be added. Two kinds of points are significant: the vertexes of polylines and the projected points. Projected points are the projection of the perpendicular dropped from one vertex of one trajectory and one segment of the other trajectory. Indeed, the shortest distance from a vertex to a segment is either the distance between two vertexes or the distance between this vertex and its projected point in Euclidean geometry. For each vertex, only segments with one or two points are associated with this vertex. Point is associated if the cell of pDM between this point of this vertex has a value. Figure 4 describes the new projected points added to the enhanced polylines. The eight black circles surround new points. Point $L_2.0a$ is very close to point $L_2.0$. For L_1, three projected points are created. For L_2, five projected points are created.

With a first d_{dF} computation process, only a small number of projected points were added to the filtered lines. The computation of this projected points require CPU time so the creation of projected points should be limited to a small number of segments with long length. Minimizing this number of projected points is a key factor to limit the required CPU time.



epf$_{0.2}$DM

num	x	y	0	0a	0b	3	3a	5	5a	8	8a	11
		X	1.000	1.037	3.937	5.800	7.101	9.000	13.043	13.600	14.381	18.600
		Y	1.600	1.604	1.906	2.100	1.978	1.800	2.503	2.600	2.506	2.000
0	1.100	1.000	0.608	0.607								
3	4.000	1.300			0.609	1.970						
3a	5.664	1.078			1.915	1.031	1.696					
6	7.000	0.900				1.697	1.083	2.193				
6a	9.075	1.134				3.415	2.147	0.670				
12	13.200	1.600							0.917	1.077	1.488	
13	14.200	1.000							1.897	1.709	**1.517**	
13a	18.600	1.000									4.480	1.000
18	19.000	1.000										1.077

epf$_{0.2}$FM

num	x	y	0	0a	0b	3	3a	5	5a	8	8a	11
		X	1	1.037	3.937	5.8	7.101	9	13.04	13.6	14.38	18.6
		Y	1.6	1.604	1.906	2.1	1.978	1.8	2.503	2.6	2.506	2
0	1.100	1.000	0.608	0.608	2.978							
3	4.000	1.300			0.609	1.970						
3a	5.664	1.078			1.915	1.031	1.696					
6	7.000	0.900				1.697	1.083	2.193				
6a	9.075	1.134				3.415	2.147	1.083				
12	13.200	1.600							1.083	1.083	1.488	
13	14.200	1.000							1.897	1.709	1.517	
13a	18.600	1.000									4.480	1.517
18	19.000	1.000										**1.517**

Figure 4: Enhanced polylines and the associated discrete Fréchet distance

Finally, the discrete Fréchet distance is computed again between the enhanced trajectories. First, the almost diagonal is computed. The $Diag_{Max}$ value of the diagonal is now 4.480 m. The value of d_{dF} computed for polylines without enhancement was 2.193 m. So, if a better path exists, its maximum value should be below or equal 2.193 m. To calculate it, the right up and left down parts of pDM are explored, the pruning value is the minimum value between $Diag_{Max}$ and the previously computed d_{dF}. Third, pFM is computed. The new d_{dF} is 1.517 m. This value is the distance between $L_{1.13}$ and its projected point ($L_{2.8a}$) on segment between $L_{2.8}$ and $L_{2.11}$. This value (1.517 m) is closer to the initial value of d_{dF} computed for L_1 and L_2 without filtering (1.612m) than the d_{dF} between filtered without enhancement lines (2.193 m).

This new data flow to compute d_{dF} described in figure 5 is more complex than the original one. Nevertheless, the computation is faster with filtering and pruning sub-processes if the number of positions is large. On the other hand, enhancement sub-process improved accuracy for trajectories with long segments. To validate this optimized discrete Fréchet distance algorithm, one experimentation on real data was conducted.

4 EXPERIMENTATION FOR LARGE DATASET

To validate this optimized process, The discrete Fréchet Distances between trajectories from a real dataset were computed. This dataset was recorded by Biro et al. [3] to analyze pigeon behavior [9]. Freeman et al. explore usual route structure and leadership phenomenon. Birds are released in pairs in the Oxford area.

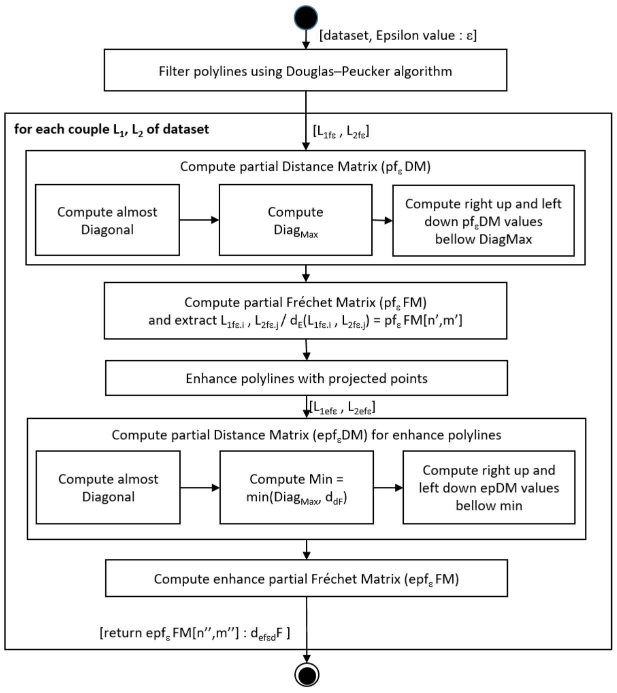

[dataset, Epsilon value : ε]
Filter polylines using Douglas–Peucker algorithm
for each couple L1, L2 of dataset
[L1fε , L2fε]
Compute partial Distance Matrix (pfεDM)
Compute almost Diagonal
Compute DiagMax
Compute right up and left down pfεDM values bellow DiagMax
Compute partial Fréchet Matrix (pfεFM) and extract L1fε.i , L2fε.j / dE(L1fε.i , L2fε.j) = pfεFM[n',m']
Enhance polylines with projected points
[L1efε , L2efε]
Compute partial Distance Matrix (epfεDM) for enhance polylines
Compute almost Diagonal
Compute Min = min(DiagMax, ddF)
Compute right up and left down epDM values bellow min
Compute enhance partial Fréchet Matrix (epfεFM)
[return epfεFM[n'',m''] : defεdF]

Figure 5: Optimized Discrete Fréchet distance processing (UML activity diagram)

This dataset is chosen for two reasons. Trajectories are complex with several loops in an open area. Trajectories are very similar: the same beginning point (the release site) and the same end point (the home loft). For this experimentation 495 trajectories from 23 different pigeons were studied. The dataset is made up of 1 533 521 points. The measured GPS accuracy during the experiment was 4m in the latitude/longitudinal plane [9]. Indeed, for pigeon flying in the air, the sky view is clear. The average distance between consecutive points was 4m. Figure 6.a illustrates this dataset.

The optimized process is tested with all possible couples of trajectories (122 265 trajectory couples). Several different configurations are analyzed. Different ε values (0 m, 25 cm, 50 cm, 1m, 2m, 3m, 4m, 5m and 10m) are investigated. In the same way, results with or without projected point are studied. Table 1 shows the filtering ability of the Douglas Peucker filter applied to the pigeon dataset. The initial number of points (1 533 521) is drastically reduced by this process even at 25 cm which retains only 26.95% of the points. This filtering results confirm the results of Shi and Cheung [23].

Average distances between consecutive points are small so the process without filtering process and projected points (d_{edF}) are not taken into account. Indeed, differences between regular d_{dF} and the enhanced version d_{edF} are very small (95% of differences are below 10cm). So, regular configuration (d_{dF} or 0m) is considered as the reference. Accuracy of configurations is deduced according to this reference. Without filtering process, the discrete Fréchet distance process takes 8 hours 36 min and 53 seconds (31 014 seconds) for an Intel Core 2 Duo E8400 3 GHz Processor and 8Go RAM computer (see table 2). With the diagonal pruning approach (but without filtering) the CPU time is reduced to 6 hours 9 minutes and 12 seconds (22 152 seconds).

Table 1: Selected points number according to the filtering parameter: ε values

ε	nb points	% selected points
0 cm	1 533 521	100.00%
25 cm	413 287	26.95%
50 cm	280 964	18.32%
1 m	187 904	12.25%
2 m	124 122	8.09%
3 m	96 640	6.30%
4 m	81 080	5.29%
5 m	70 713	4.61%
10 m	45 277	2.95%

Then, filtering process is still required to shrink down the CPU time. Figures 7 and 8 summarize different configurations and relation between efficacy and efficiency. Three important criterion are analyzed:

- 95 percentile error;
- maximum error;
- CPU time.

Indeed, median errors are very similar. These errors are below 1 m for configurations with ε below or equal 5 m. In figure 7, we can notice two curves. The first one regroups configurations without projected points. The second one regroups configurations with projected points. Configurations without projected points are very fast compared to configurations with the same epsilon value and projected points. If we focus only on accuracy (95 percentile), configurations 3 m, 4 m, 5 m and 10 m with projected points are incompatible with the GPS accuracy (4 m). The fastest compatible configuration is ε = 2 m without projected points (2 m or d_{df2F}). In 157 seconds, d_{df2F} gives measures with 95 percentile error equals to 3.75 m. Unfortunately, for this configuration, the maximal error is very large (158.07 m). Only, few d_{df2F} measures are erroneous but these errors are too large. If 95 percentile error and max error are considered by the user, configuration 4 m with projected points (d_{ef4dF} or 4 m with pp) is a good compromise. The 95 percentile error is 2.63 m and maximal error is 25.20 m. However, 1 070 seconds are required to compute d_{ef4dF}. To conclude this analysis, thanks to the filtering process, the number of points decrease drastically and configurations are efficient. Without projected points, configurations are effective according to 95 percentile error but not according to max error. To improve efficacy, projected points must be added. This process is CPU expensive but improve the accuracy even if larger ε values are selected.

To illustrate this improvement, the maximal error of d_{f2dF} (ε = 2 without projected points) is presented in figure 9. This error measure the difference between d_{f2dF} and d_{dF} for trajectory with number 167 and 168. These two trajectories describes two pigeons released together. Trajectories are very close to each other. Small circle points represent the original GPS points. Only square points are selected by the filtering process (ε = 2 m). For the straight part of lines (see figure 9.a), a small number of points are chosen and

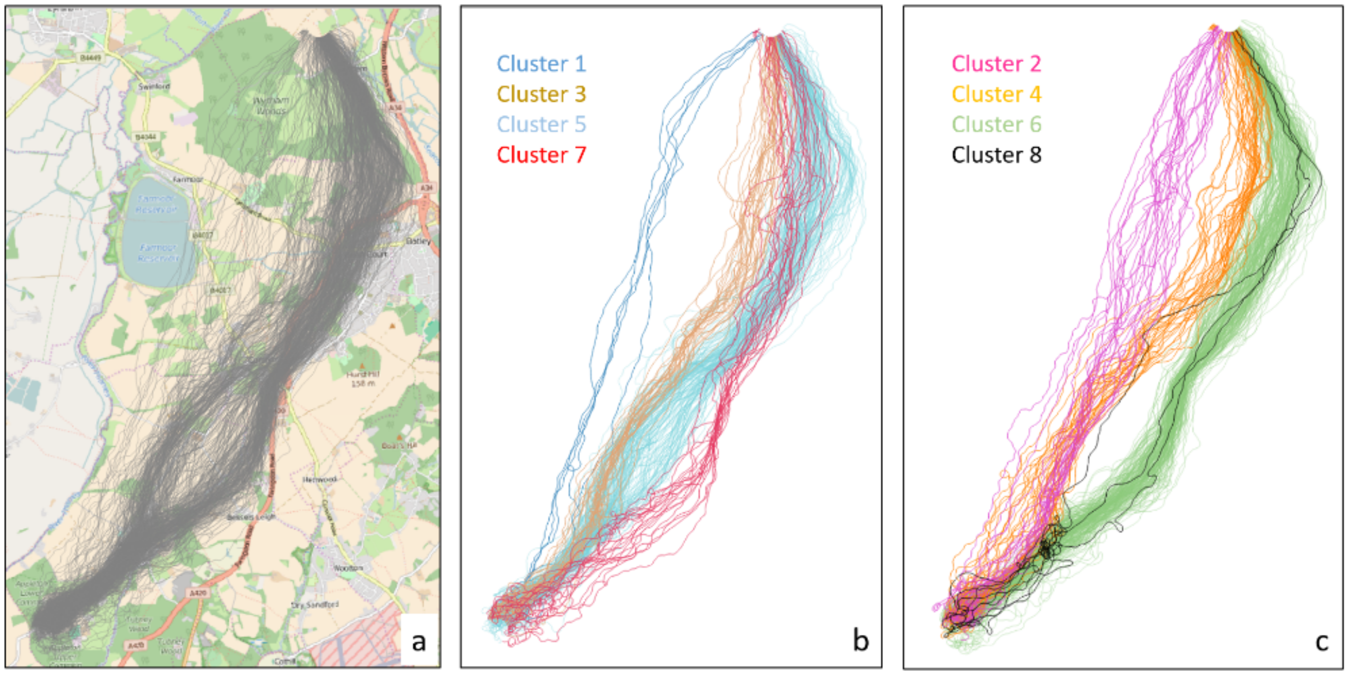


Figure 6: Pigeon dataset (a) with clustering process result (b and c)

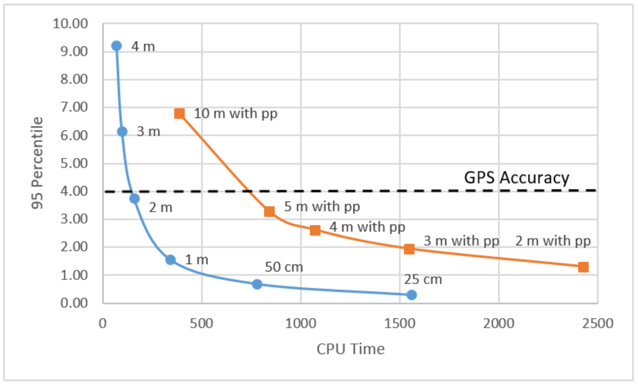


Figure 7: Relation between Time CPU and 95 percentile error according to ϵ value and with or without projected points (pp)

Table 2: CPU time and number of points of the different configurations of optimized Fréchet Distance

		Errors		
ε (m)	CPU Time (sec)	median (m)	95 percentile (m)	max (m)
0 (regular)	31 014	0.00	0.00	0.00
0 (optimized)	22 152	0.00	0.00	0.00
0.25	1560	0.04	0.31	39.56
0.50	779	0.09	0,70	53.73
1	339	0.18	1.57	83.32
2	157	0.35	3,75	158.07
3	96	0.53	6.17	138.89
4	69	0.73	9,22	213.62
5	55	0.96	13.13	222.51
10	27	3.07	45.26	466.05

distances between same line points are important. The selected big white circle point are far away from blue square points. Then, the distance to the closer one (the big blue circle) is 178.36 m. This distance is the d_{f2dF} between tracks 167 and 168. The regular d_{dF} for this trajectory couple is smaller (20.29 m). So, the error is 158.07 m. The figure 9.b shows the enhanced lines with selected and projected points (the stared points). Distance between stared points is smaller (21.35 m) and error is reduced to 1.06 m. This example demonstrates why enhancement process drastically improve the distance measure. For this example, only 0.36 % of distances are greater than 25 m. This percentage value is very small but it represents 436 distances with very large errors.

Similarity measure in a data mining context is a key point. To illustrate this sentence, figure 6.b and 6.c. show 8 clusters defined by hierarchical clustering [12]. This method used average linkage criteria and a 1 000 m distance to split the dendrogram. Optimized Fréchet distance can deal with this large dataset. These clusters highlight the different pigeon behaviors. For example, the two main clusters 5 and 6 describe two different behaviors. We can notice that the upside parts of these two clusters are similar. On the other

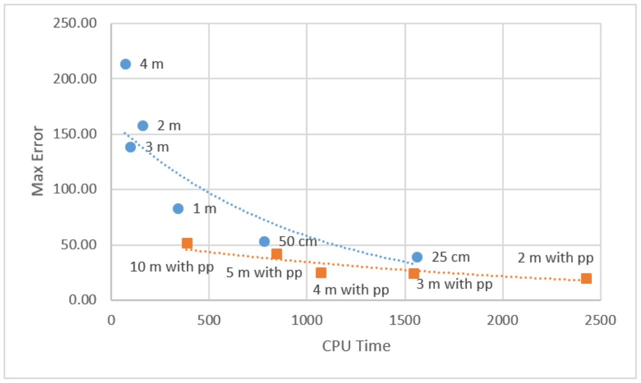


Figure 8: Relation between Time CPU and Max error and according to ϵ value and with or without projected points (pp), dot lines show the exponential trend line

Table 3: CPU time and number of points of the different configurations of enhanced optimized Fréchet Distance with projected points

		Errors		
ε (m)	CPU Time (sec)	median (m)	95 percentile (m)	max (m)
0	22 152	0.00	0.00	0.00
0.25	27 189	0.04	0.17	7.38
0.50	12 484	0.08	0.34	8.30
1	5 748	0.16	0.67	10.81
2	2 424	0.29	1.31	19.99
3	1 545	0.42	1.95	24.42
4	1 070	0.55	2.63	25.20
5	841	0.68	3.29	42.29
10	384	1.46	6.81	52.07

hand, the downside parts is different. For these one, cluster 6 is at the right of cluster 5.

A fast $d_{f\varepsilon dF}$ with small ε value can be selected if the user needs to define median trajectories [20] or more complex patterns like Trajectory Box Plot [8] for each cluster. On the other hand, if user needs to classify each trajectory in one cluster a similarity measure with a small maximum error has to be used. Slower enhanced configurations such as $d_{ef\varepsilon dF}$ with medium ε value gives better classification result.

5 CONCLUSION

Compute Fréchet distance for large trajectory datasets consume a large amount of CPU time (more than heigh hours and half for our experimentation). In this article, a new optimized process is introduced. This one includes three different improvements. The first one is a Douglas and Peucker filtering process. Indeed, for GPS trajectories, the number of positions can be dramatically reduced. In our experience, fewer than 10% of points are selected. The second one is a pruning process. Only a small part of the two matrices required to compute the discrete Fréchet distance are computed. The last one is an enhancement process. Remarkable projected points are added to the two polylines. This new optimized process is more complex than regular discrete Fréchet distance, but CPU Time is reduced to few minutes or less for our experimentation. In terms of accuracy, a balance between CPU time and precision is required. The accuracy highly depends on the filtering parameters and the number of projected points added to the enhanced trajectories. Nevertheless, accuracy of this new process has to match GPS accuracy. If users tolerate few large errors, the adding of projected points is not required. Otherwise, if large errors must be avoided, the enhancement is recommended. This optimized similarity measure is very useful for trajectory mining.

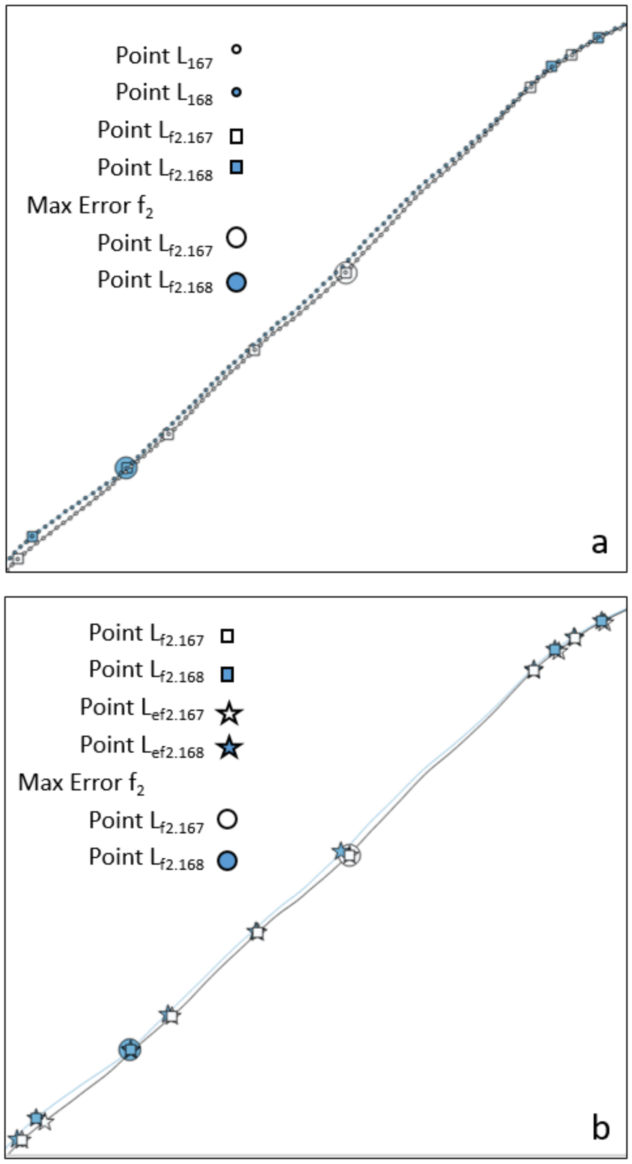


Figure 9: Max error between lines 167 and 168 and improvement with projected points

In a short term, we plan to further validate this process on different datasets (pedestrians, cars, etc.) and different application domains (animal study, traffic analyzis, etc.). Other clustering methods such as k-means will be investigated using this similarity distance. To reduce the CPU Time, this optimized process will be customized to run in parallel computing environments.

In the middle term, several research directions are planned. Firstly, a hybrid method has to be defined. This one adds only few projected points. The main difficulty resides to define the fit criteria (local or global) before the enhancement process. Several criteria have to be tested. Trajectory has spatial and temporal dimension. So, this spatial similarity measure has to be extended to spatio-temporal dimensions. In the same way, this optimized distance can be extended to define a data-alignment process and to compute average distance between trajectories.

REFERENCES

[1] H. Alt and M. Godau. 1995. Computing the FrÃl'chet Distance Between two Polygonal Curves. *International Journal of Computational Geometry and Applications* 5, 1 (1995), 75–91.

[2] Jie Bao, Yu Zheng, David Wilkie, and Mohamed Mokbel. 2015. Recommendations in location-based social networks: a survey. *GeoInformatica* 19, 3 (2015), 525–565. https://doi.org/10.1007/s10707-014-0220-8

[3] Dora Biro, David JT Sumpter, Jessica Meade, and Tim Guilford. 2006. From compromise to leadership in pigeon homing. *Current Biology* 16, 21 (2006), 2123–2128.

[4] Alminas Civilis, Christian S Jensen, and Stardas Pakalnis. 2005. Techniques for efficient road-network-based tracking of moving objects. *IEEE Transactions on Knowledge and Data Engineering* 17, 5 (2005), 698–712.

[5] Thomas Devogele. 2002. A New Merging Process for Data Integration Based on the Discrete FrÃl'chet Distance. In *Advances in Spatial Data Handling*, DianneE. Richardson and Peter van Oosterom (Eds.). Springer Berlin Heidelberg, 167–181. https://doi.org/10.1007/978-3-642-56094-1_13

[6] D.H. Douglas and T.K. Peucker. 1973. 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 (1973), 112–122.

[7] T. Eiter and H. Mannila. 1994. *Computing Discrete FrÃl'chet Distance.* Technical Report. Technische Universitat Wien. 64 pages.

[8] Laurent Etienne, Thomas Devogele, Maike Buchin, and Gavin McArdle. 2016. Trajectory Box Plot: a new pattern to summarize movements. *International Journal of Geographical Information Science* 30, 5 (2016), 835–853.

[9] Robin Freeman, Richard Mann, Tim Guilford, and Dora Biro. 2011. Group decisions and individual differences: route fidelity predicts flight leadership in homing pigeons (Columba livia). *Biology Letters* 7, 1 (2011), 63–66. https://doi.org/10.1098/rsbl.2010.0627

[10] E. Grimson, X. Wang, G.W. Ng, and K.T. Ma. 2008. Trajectory Analysis and Semantic Region Modeling Using A Nonparametric Bayesian Model. (2008).

[11] F. Itakura. 1975. Minimum Prediction Residual Principle Applied to Speech Recognition. *IEEE Transactions on Acoustics, Speech and Signal Processing* 23, 1 (1975), 67–72.

[12] L. Kaufman and P.J. Rousseeuw. 1990. *Finding groups in data: an introduction to cluster analysis.* Vol. 39. Wiley Online Library.

[13] E. Keogh, K. Chakrabarti, M. Pazzani, and S. Mehrotra. 2001. Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and Information Systems* 3, 3 (2001), 263–286.

[14] Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. 2007. Trajectory Clustering: A Partition-and-group Framework. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data (SIGMOD '07)*. ACM, New York, NY, USA, 593–604. https://doi.org/10.1145/1247480.1247546

[15] Zhenhui Li, Bolin Ding, Jiawei Han, Roland Kays, and Peter Nye. 2010. Mining periodic behaviors for moving objects. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 1099–1108.

[16] Miao Lin and Wen-Jing Hsu. 2014. Mining GPS data for mobility patterns: A survey. *Pervasive and Mobile Computing* 12 (2014), 1–16.

[17] Jean Damascène Mazimpaka and Sabine Timpf. 2016. Trajectory data mining: A review of methods and applications. *Journal of Spatial Information Science* 2016, 13 (2016), 61–99.

[18] Nirvana Meratnia and Rolf A. de By. 2004. Spatiotemporal Compression Techniques for Moving Point Objects. In *Advances in Database Technology - EDBT 2004 (Lecture Notes in Computer Science)*, Vol. 2992. Springer Berlin / Heidelberg, 561–562. https://doi.org/10.1007/b95855

[19] Mahsa Mirzargar, Ross T Whitaker, and Robert M Kirby. 2014. Curve boxplot: Generalization of boxplot for ensembles of curves. *IEEE Transactions on Visualization and Computer Graphics* 20, 12 (2014), 2654–2663.

[20] François Petitjean, Alain Ketterlin, and Pierre Gançarski. 2011. A global averaging method for dynamic time warping, with applications to clustering. *Pattern Recognition* 44, 3 (2011), 678–693.

[21] H. Sakoe and S. Chiba. 1978. Dynamic Programming Algorithm Optimization for Spoken Word Recognition. *IEEE Transactions on Acoustics, Speech, and Signal Processing* 26 (1978), 43–49.

[22] S. Salvador and P. Chan. 2007. Toward accurate dynamic time warping in linear time and space. *Intelligent Data Analysis* 11, 5 (2007), 561–580.

[23] Wenzhong Shi and ChuiKwan Cheung. 2006. Performance evaluation of line simplification algorithms for vector generalization. *The Cartographic Journal* 43, 1 (2006), 27–44.

[24] Yilun Wang, Yu Zheng, and Yexiang Xue. 2014. Travel time estimation of a path using sparse trajectories. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining.* ACM, 25–34.

[25] Nicholas Jing Yuan, Yu Zheng, Liuhang Zhang, and Xing Xie. 2013. T-finder: A recommender system for finding passengers and vacant taxis. *IEEE Transactions on Knowledge and Data Engineering* 25, 10 (2013), 2390–2403.

[26] Yu Zheng. 2011. Location-based social networks: Users. In *Computing with spatial trajectories.* Springer, 243–276.

[27] Yu Zheng. 2015. Trajectory data mining: an overview. *ACM Transactions on Intelligent Systems and Technology (TIST)* 6, 3 (2015), 29.

[28] Yu Zheng, Yanchi Liu, Jing Yuan, and Xing Xie. 2011. Urban computing with taxicabs. In *Proceedings of the 13th international conference on Ubiquitous computing.* ACM, 89–98.

[29] Yu Zheng and Xing Xie. 2011. Learning travel recommendations from user-generated GPS traces. *ACM Transactions on Intelligent Systems and Technology (TIST)* 2, 1 (2011), 2.