J Vis (2018) 21:457–470
https://doi.org/10.1007/s12650-017-0471-1

REGULAR PAPER

Hutong Wu · Shicao Jia · Jinsong Wang · Jiawan Zhang

M3: visual exploration of spatial relationships between flight trajectories

Received: 31 August 2017 / Accepted: 5 November 2017 / Published online: 5 January 2018

Abstract Outlier detection and clustering are important to analyze trajectory. While many algorithms have been developed to tackle these issues, they lack the combination with visualization to enable the involvement of human intelligence during the analyzing process. We propose a visual framework called M3, which combines data mining algorithms with visualization technique through three coordinated views: Map, MST, and FSDMatrix. Map view displays the spatial information of trajectories. MST is a minimum spanning tree, which presents the relationships between trajectories. In MST, each node represents a trajectory; edges between nodes denote the Fréchet distance between trajectories. FSDMatrix shows a matrix of pairwise Free Space Diagram to assist in detecting outliers and clustering trajectories. The three views are interacted with each other. Through case studies, we discuss the applicability of our framework and demonstrate the convenience brought by it.

Keywords Visual analytics · Trajectory outlier detection · Trajectory clustering · Fréchet distance

1 Introduction

With the advances in mobile communication and positioning technology for tracking the positions of continuously moving objects, the trajectories of moving objects such as civil airplanes can be obtained easily. Since the amount of trajectory data increases year by year, application system managers must regularly improve their system to rapidly and efficiently analyze data. Analysis of trajectory data has recently gained great interest (Lee et al. 2007). To explore and analyze trajectories, two typical data analysis tasks are indispensable. One is outlier detection, the other is trajectory clustering.

Trajectory outliers are those that are grossly different from others in a set of trajectories. They may be inconsistent as a whole or just in sub-trajectories with most of the other trajectories. There are two reasons that result in the occurrence of outliers. One comes from the quality of data, such as missing data, accuracy problems, and precision deficiency (Andrienko et al. 2016). Detecting these outliers is a cornerstone of data-cleaning or clustering. Otherwise, the outliers may interfere with the analysis methods. Another reason comes from the moving objects. For example, a driver may make a detour because of the road congestion and an airplane may deviate from its normal flight path because of bad weather. Detecting these outliers is the first step to interpret the patterns of moving objects.

H. Wu (✉) · S. Jia · J. Zhang
Tianjin University, No.135, Yaguan Road, Jinnan District, Tianjin, China
E-mail: wht@tju.edu.cn

J. Wang
Southwest Electric & Telecom Engineering Institute, Chengdu, China
E-mail: jinsong.wang@126.com

Trajectory clustering is the process of grouping a set of trajectories into classes of similar objects. There are two kinds of clustering methods in terms of completeness of trajectory, including whole trajectory clustering and sub-trajectory clustering. Sometimes, trajectories even in same category may show some difference in details. Clustering these trajectories can find the move patterns for intention analysis, and it has more difficulties which means clustering needs to be fine grained.

Many algorithms have been developed for outlier detection and trajectory clustering. Although these algorithms have been widely used in many fields, they also have limitations: they are not flexible enough and usually have a gap between human and data. It is meaningful to combine those algorithms with human intelligence through visualization. That is, we need a visual framework to explore and analyze trajectories.

During analyzation, users may explore movement data from different levels and perspectives. Take trajectories for example, users may explore the whole set of trajectories, subset of trajectories or single trajectory. Therefore, visual framework should provide the flexibility for users to choose.

Some problems should be noticed in existing visualization methods when analyzing trajectories:

- The most commonly used way to visualize trajectories is to present them with lines on map in 2D or 3D. In order to check the details of spatial characteristics of trajectories, we usually need to zoom in. That usually makes us lost focus in the crossover lines. It would be difficult to decide which trajectories should be explored next, and which ones have been already explored.
- To present the clusters and outliers of trajectories, we can distinguish them by different colors. But what about the relationships between the clusters? What about the differences between two clusters? To discern the spatial differences between two clusters or two trajectories, it is not necessarily enough to explore trajectories only in map view, and it also brings visual burdens on analysts.
- Interaction with a group of trajectories is not easy. Choosing one trajectory of interest from a group of lines is more like pulling a thread through a needle, and it is not convenient and easy at all. Furthermore, choosing sub-group of trajectories can also be difficult. It is not practicable to select lines one by one on the map.

One of the reasons causing the problems above is that trajectories are unstructured and existing visualization methods do not show the spatial relationships between trajectories or clusters. If trajectories with high similarity are connected with each other to form an organized community, it will be much more easier to choose the trajectories and clusters based on the topological structure of trajectories, and users will constantly remain focus without losing context during exploring and interaction.

Based on the considerations above, we propose a visual framework called M3, which consists of three views: Map, MST, and FSDMatrix. Map view provides substantial spatial information. MST presents a set of trajectories as a minimum spanning tree in which each node stands for one trajectory and nodes are connected with respect to the distance between trajectories. FSDMatrix shows a matrix of pairwise Free Space Diagram and it arranges the outliers at the top left corner, and clustering phenomenon can be recognized through it.

In summary, the contributions of this paper are as follows:

- We propose a method to show the topology of trajectories using MST, which also shows the relationships between trajectories or clusters. Moreover, MST provides another visual tool for convenient interaction with trajectories, giving more choice besides most commonly used methods like picking lines on a map.
- We propose a visual tool called FSDMatrix to detect outlier trajectories and clusters by some simple interactions with it, which makes it possible to involve human intelligence during the analyzing process.
- We design a visual framework, called M3, which links the Map view, MST view and FSDMatrix view together for better exploration of trajectories.

The rest of the paper is organized as follows. After an overview of related work in Sect. 2, the pipeline of our framework is described in Sect. 3. Methods used in M3 are discussed in detail in Sect. 4. Two case studies are shown in Sect. 5 and finally we draw our conclusions in Sect. 6.

2 Related work

Existing studies relevant to our research are divided into four categories, namely trajectory distance and similarity, trajectory clustering, trajectory outlier detection, and visualization of movement. In this section, we briefly summarize the previous work in each category.

2.1 Trajectory distance and similarity

To detect trajectory outliers and cluster trajectories, it is very important to efficiently and effectively calculate the similarity between trajectories. Trajectory distance is frequently used to represent the similarity between trajectories.

There are several ways to measure the distance between each pair of elements in trajectory data set. Those methods fall into two types, one considers the geometry of the path but ignores the temporal aspects, and the other focuses not only on the moving entities but also the time attribute (Buchin 2010). In this paper, we only focus on the first type.

One of the most widely used distance function is Hausdorff distance (Alt et al. 1995). While it is appropriate in many applications, Hausdorff distance only takes into account the discrete point sets of curves without consideration of the path of curves. Since the ordering of the points along the curves is important for trajectory outlier detection and clustering, Hausdorff distance is not applicable for trajectory analysis in this situation. Our choice is Fréchet distance (Alt and Godau 1995), which is a measure of similarity between curves that takes into account not only the location but also the ordering of the points along the curves.

2.2 Trajectory clustering

Many algorithms can be applied to trajectory clustering, such as *k*-means (Lloyd 1982), BIRCH (Zhang et al. 1996), DBSCAN (Ester et al. 1996), OPTICS (Ankerst et al. 1999) and STING (Wang et al. 1997). TRACLUS (Lee et al. 2007) is proposed based on a partition-and-group framework.

Different clusters are always encoded with different colors on the map or STC (Space Time Cube) Kraak (2003) to show the result. Aggregate flows (Andrienko and Andrienko 2011) and density map (Scheepens et al. 2016) can be used for spatial generalization and aggregation of movement data. Another way to aggregate movement data is to extract a representative trajectory to denote the whole trajectories in that cluster (Lee et al. 2007). While most visualization methods are meant to simplify the cluster and reduce clutter, few of them pay attention to the relationships of trajectories and the topological structure of the clusters.

2.3 Trajectory outlier detection

Many outlier detection algorithms have been developed for multi-dimensional points. These algorithms can be classified into four categories: distribution-based, distance-based, density-based, and deviation-based (Lee et al. 2008).

Knorr et al. (2000) have applied the distance-based algorithm to detect trajectory outliers. They defined the distance between two group of trajectories using their summary information. It can successfully detect outliers if their directions, starting (or ending) points and so on are completely different from those of other trajectories, which is not appropriate in any case.

Li et al. (2007) use classification to detect trajectory outliers. In this algorithm, the classification model is built using the training set, and trajectories are classified into normal or abnormal sets. Good training set is critical for this algorithm to be successful.

TRAOD (Lee et al. 2008) is a trajectory outlier detection algorithm based on partition-and-detect framework using a hybrid of the distance-based and density-based approaches. iBAT (Zhang et al. 2011) uses Isolation-Based Anomalous Trajectory detection method, and it achieves remarkable performance with large-scale taxi data.

2.4 Visualization of movement

Chang et al. (2012) designed a system called TraceViz to filter trajectories based on the similarity of the brush shapes. Guo et al. (2011) provided pertinent interactions such as directional brush and ring sliders in the traffic view of TripVista. Hurter et al. (2008) used the brush-pick-drop flowchart to select a group of trajectories into a new view, and then repeat the flowchart for deeper filtering. Krüger et al. (2013) used lens metaphor to support complex filter expressions given a series of areas of interest. Wang et al. (2014) designed a visual analysis system to explore sparse traffic trajectory data recorded by transportation cells. They ignore the micro-behaviors of individual vehicles, and focus on inter-cell flow pattern. Wu et al. (2016) designed an interactive visual analytics system called TelcoVis to explore co-occurrence in urban

human mobility based on telco data. They combined well-established visualization techniques, such as heat map and parallel coordinates, with user interaction to provide a new way to explore the data from multiple levels and perspectives. Andrienko et al. (2013) used group space to analyze movement behaviors of individuals in a group. In their system, they applied space transformation to convert trajectories of individuals from geographical space to an abstract "group space", taking account of both position and direction. Based on the positions mapped onto the group space, they compare the behaviors of different individuals and determine their roles within the groups.

3 M3 framework

As shown in Fig. 1, the framework consists of three main views: Map view, MST view, and FSDMatrix view. The pipeline of M3 is displayed in Fig. 2. It includes four parts, that is, preprocessing, pairwise calculation, construction and interaction.

Simplification of trajectories is necessary to reduce the total number of data points in the trajectories, in the order to decrease the computation cost of following analysis. Aviation airplane usually fly by a straight path during most of the time. By removing unnecessary points in trajectories without losing precision, the overall data points can be dramatically reduced. We use Ramer–Douglas–Peucker algorithm to simplify trajectories. Map view is rendered based on the result of simplification. Map view shows the lines of trajectories on the map, in which the colors of lines reflect the communities of MST. And in order to distinguish the course of trajectories, we use blue dot and red dot to represent origin and destination of each trajectory.

MST can be viewed as an interface between trajectories and users. First, each trajectory is presented as a node in it. Then, every node is connected according to the distance between trajectories. We use Fréchet distance (Alt and Godau 1995) since it is suitable in this situation. But a fully connected network is not necessary for trajectory analysis, which brings much more computation burden on following steps. Therefore, extracting the skeleton of the network is indispensible. MST, the tree that connects all nodes with minimum total edge length, is a novel way to do this. The best characteristic of MST is that similar nodes usually connect together along a branch or form a community. In order to enable fine-grained clustering of trajectories, Newman–Girvan algorithm (Girvan and Newman 2002), one community detection algorithm based on edge betweenness, used on MST, and different communities encoded with different colors. To reduce the crossover between edges of MST, we use Fruchterman–Reingold algorithm (Fruchterman and Reingold 1991), which assigns forces among the set of edges and the set of nodes to simulate the motion of the edges and nodes to minimize the energy. One interesting phenomenon about MST is that potential outliers have been discovered directly in map view that are mostly located on the leaves or the outer parts of MST.

FSDMatrix is the extension of FSD (Free Space Diagram) (Alt and Godau 1995), and it is used for trajectory outlier detection and trajectory clustering just by arranging the order of rows and columns based on outlier level which will be discussed in detail in Sect. 4.4. Users can apply color scheme to the outliers and clusters in FSDMatrix, and this change can be reflected on MST at the same time. Furthermore, by observing the FSD in FSDMatrix, users can discern which specific parts are different between two trajectories.

When analyzing large amount of data, it will be helpful to automatically detect the boundaries between clusters for rough classification. The boundary detection algorithm used to extend FSDMatrix will be discussed in Sect. 4.6 and the related use case studies are presented in Sect. 5.2.

In summary, the procedure of M3 includes four steps:

1. Extract the spatial information from trajectories data and simplify the result.
2. Calculate pairwise Fréchet distance between trajectories.
3. Construct FSDMatrix based on the result of the second step for outlier detection and trajectory clustering, and construct MST based on the network generated from the second step and detect communities.
4. Add interactions among Map view, MST view and FSDMatrix view for better exploration of trajectories.

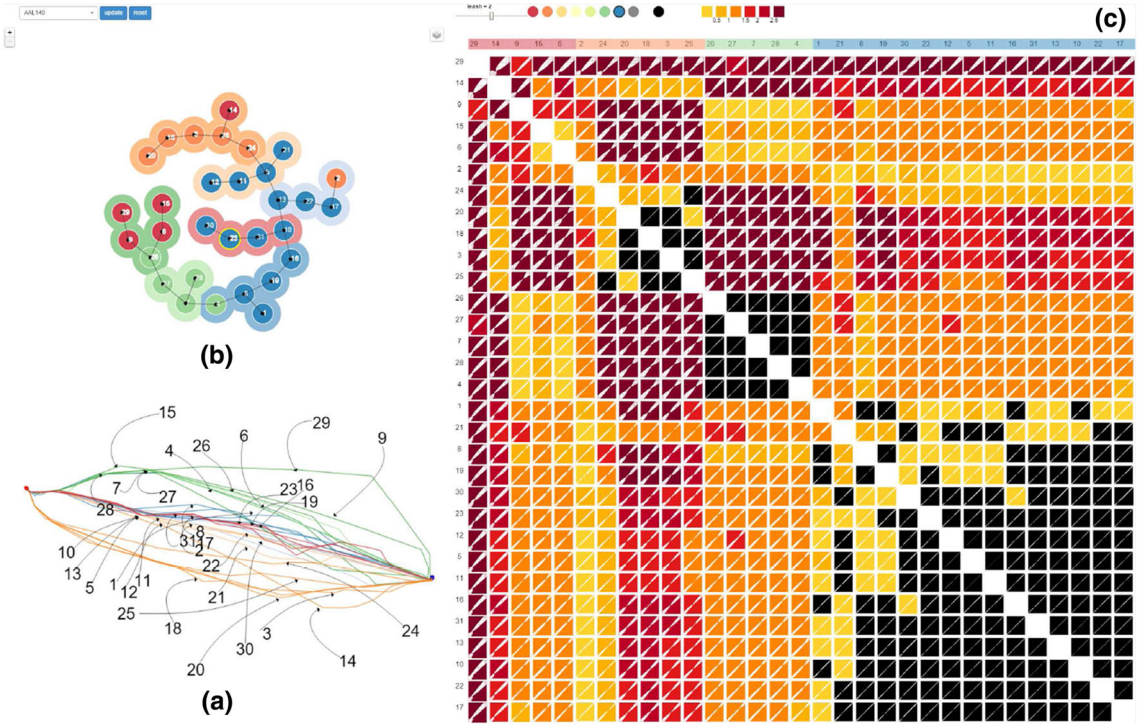

Fig. 1 Three views of M3. **a** Map view: shows trajectories on a map, in which the colors correspond to the communities in MST. **b** MST view: shows the topology of trajectories and the possible communities. It is interesting to find that outliers are always on the leaves or the outer parts of MST. **c** FSDMatrix view: extension of FSD, used to identify outlier

4 Methods

In this section, we first give the basic concepts of Fréchet distance and FSD, and show the features of FSD, and then describe the calculation of pairwise Fréchet distance between trajectories. The output of computation step is a complete graph and a matrix, which are the inputs for the construction of MST and FSDMatrix. Interactions among three views will be discussed in Sect. 4.5. Moreover, based on the characteristics of FSDMatrix, we give the boundary detection of it for massive trajectory data.

4.1 Fréchet distance and Free Space Diagram

An intuitive definition of Fréchet distance is that: suppose a man is walking along one curve f and a dog is walking along the other curve g with a leash connected. Both are allowed to control their speed and even stop but are not allowed to backtrack. Then, the Fréchet distance of f and g is the shortest leash length that allows the owner and the dog to walk on their respective curves from start to end.

Then, we give the formal definition: let f and g be two given curves in V, that is, $f : [a, a'] \to V$ and $f : [b, b'] \to V$, then the Fréchet distance $\delta_F(f, g)$ is

$$\delta_F(f,g) := \inf_{\substack{\alpha[0,1] \to [a,a'] \\ \beta[0,1] \to [b,b']}} \max_{t \in [0,1]} \left\{ d\big(f(\alpha(t)), g(\beta(t))\big) \right\}, \tag{1}$$

where α and β are continuous and non-decreasing functions with $\alpha(0) = a$, $\alpha(1) = a'$,$\beta(0) = b$ and $\beta(1) = b'$, d is the distance function of V.

Fréchet distance can also be applied to the polygonal curves which are composed of edges. For example, let $P : [0, p] \to V$ and $Q : [0, q] \to V$ be polygonal curves and p, q are the numbers of edges of P and Q. To solve the problem, consider the simplest case in which $p = 1$ and $q = 1$. Define

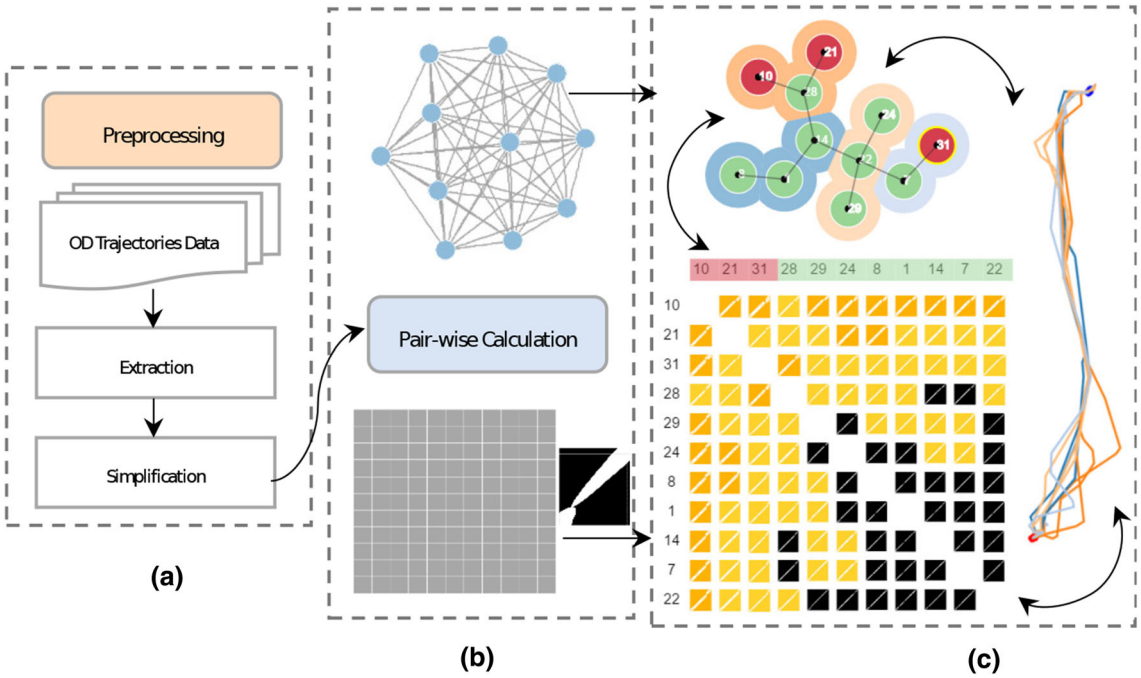

Fig. 2 Pipeline of M3. **a** Preprocessing: first, extract spatial information from trajectories data, then simplify the trajectories to reduce the number of points in each trajectory. **b** Pairwise calculation: calculate the pairwise Fréchet distance between trajectories. After that, a matrix and a complete graph will be generated. **c** Construction and interaction: construct MST from the complete graph and detect communities from MST. Highlight the inner balls of MST for distinguishing outliers and different clusters. At the same time, combine matrix with FSD to construct FSDMatrix for outlier detection and trajectory clustering. Some interactions have been added on the FSDMatrix. After construction, add interactions among the three view: Map view, MST view, and FSDMatrix, therefore, M3 for short

$$F_\epsilon := \left\{(s,t) \in [0,1]^2 | d\big(P(s), Q(t)\big) \leq \epsilon\right\} \tag{2}$$

as Free Space which is shown as white area in the cell of Free Space Diagram (see Fig. 3). Then, Free Space Diagram (FSD) is

$$D_\epsilon := \left\{(s,t) \in [0,p] \times [0,q] | d\big(P(s), Q(t)\big) \leq \epsilon\right\}. \tag{3}$$

Furthermore, free space F_ϵ in every cell of FSD D_ϵ is the intersection of the unit square with an ellipse. In this case, the Fréchet distance is at most ϵ if and only if there exists a curve from (0, 0) to (p, q) which is monotone in both coordinates (Alt and Godau 1995). In this paper, we call the monotone curve in the free space as "safe line". Along the safe line from (0, 0) to (p, q), some characteristics can be concluded:

- The thinner parts of free space are always the parts where P and Q are more far away or have much more difference, while the thicker parts are always the parts where the two are closer or more similar.
- Free space is more likely to be the intersection of two parallel lines with the unit square when the Fréchet distance is much smaller.
- With ϵ getting smaller, the free space will get thinner and thinner until the most thinnest part break apart. See Fig. 3 as an example.

Much more about the computation of Fréchet distance can refer to Eiter and Mannila (1994) and Alt and Godau (1995).

4.2 Pairwise Fréchet distance

Let the analysis object be a trajectory set $S = (f_1, f_2, ..., f_n)$, and each $f_i (i = 1, 2, ..., n)$ be a list of space positions $(p_1, p_2, ..., p_{m_i})$ with total m_i points. Different f can have different number of points. And we

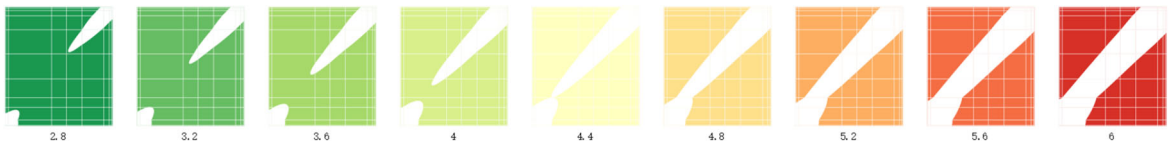

Fig. 3 FSDs of trajectory 5 and trajectory 29 of Flight AAL140. Horizontal axis stands for trajectory 5 and vertical axis stands for the other. The two trajectories are shown in Fig. 1. In this case, the free space is shaped like a spindle. In each FSD, the thinner, the more far away or more different. With ϵ' increasing, the free space is expanding. Numbers below FSDs are the values of $\epsilon'(\delta_F = 4.4)$

assume that between two consecutive space positions, the moving object moves along a straight line for each pair p_i to p_{i+1}.

The input of MST is a fully connected graph of S, and the weight of each edge is the Fréchet distance between two nodes. Similarly, the input of FSDMatrix is a matrix of S, that is $M = \{\delta_F(f_i, f_j)|i, j = 1, 2, .., n\}$.

The complexity of finding the Fréchet distance between two polygonal with p and q points is $\mathcal{O}(pq \log pq)$ (Eiter and Mannila 1994). To accelerate the computation, sometimes we need to simplify the trajectories to reduce the number of points in polygonal curves (Poiker and Douglas 1973).

4.3 Construction of MST

We borrow the idea of MST from SPADE (Qiu et al. 2011) and FlowSOM (Van Gassen et al. 2015) which are used in flow cytometry (Aghaeepour et al. 2013). M3 employs Prim's algorithm (Prim 1957; Pettie and Ramachandran 2000) to construct MST, where each node stands for one trajectory. Force directed layout was introduced to reduce the crossover of edges in MST, without changing the topology of MST at the same time.

Every node in MST is visualized as two parts: inner ball and outer ring. The default color of all inner balls of MST is the same, but can be specified by users in FSDMatrix for distinguishing outliers and clusters (see Fig. 4). The colors of outer rings correspond to the communities detected by Newman–Girvan algorithm (Girvan and Newman 2002). In order to merge the outer rings of the same community together, we use Voronoi Diagram (Senechal 1993) to split the outer rings when they overlay together. When mouse cursor moved into any outer ring, the corresponding community will be highlighted. Similarly, inner balls will be highlighted once mouse cursor moved in. In addition, every node can be fixed and locked in a specific position; at the same time, the other unlocked nodes will change their positions to minimize the energy of the system. Some examples of MST are shown in Fig. 9.

The great characteristic of MST for trajectories is that outliers are mostly located at the leaves or the outer parts of MST, while normal ones stay inside. Taking Fig. 4 as an instance, the inner balls of this MST are colored by FSDMatrix for outlier detection and trajectory clustering, in which the red and orange ones stand for outliers and green and blue ones stand for normal clusters.

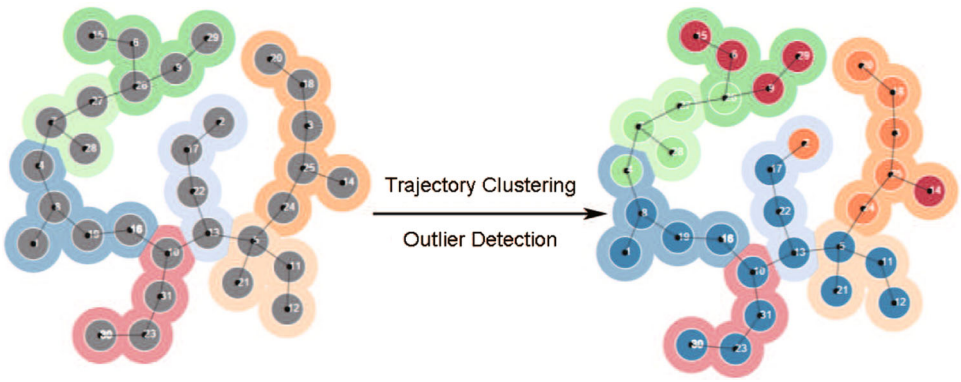

Fig. 4 Construction of MST. Every node is visualized as two parts (inner ball and outer ring). The outer ring is grouped according to the communities detected, and the default color of inner ball is the same as that in FSDMatrix

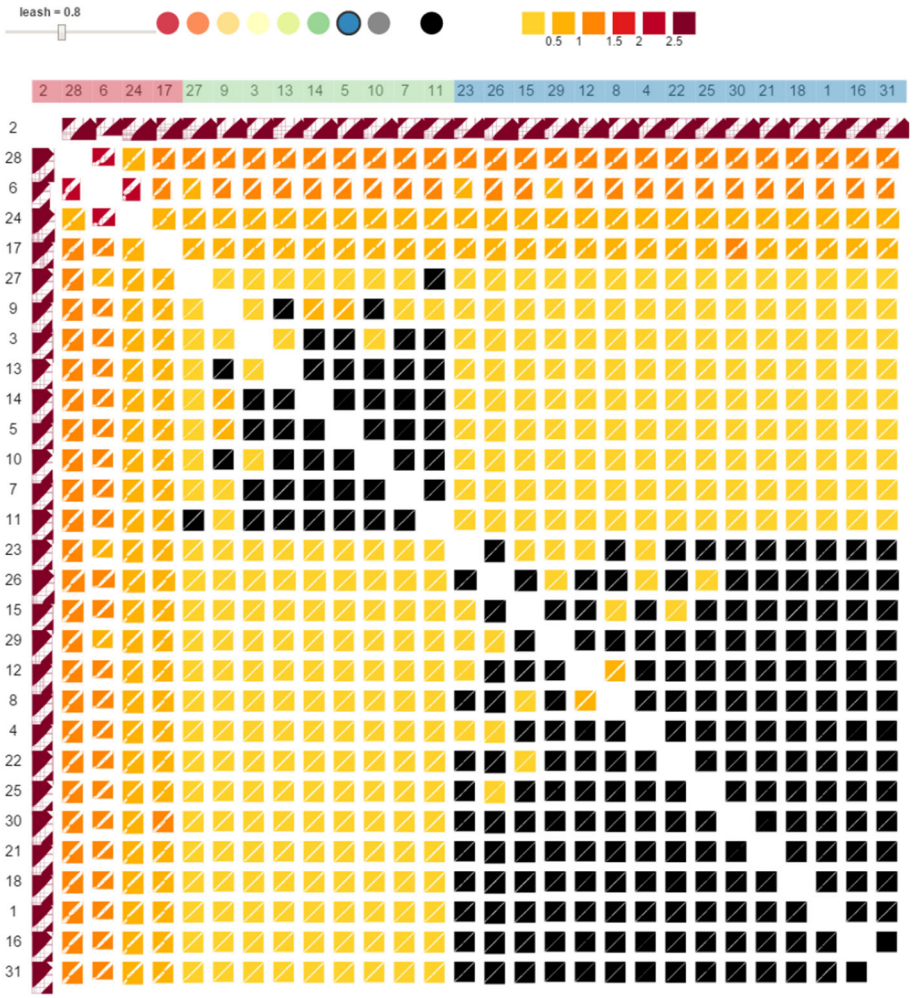

Fig. 5 FSDMatrix layout. The uppermost top is a slider for value of ϵ' and some color buttons for change mode. Below the slider is column header to denote trajectories

4.4 Construction of FSDMatrix

FSDMatrix, as shown in Fig. 5, is a matrix of FSD, where each row or column is a list of FSDs for trajectories f_i and $f_j \in (S - f_i)$. The formal definition of FSDMatrix of S is

$$\text{FSDMatrix}_S := \left\{ D_{\epsilon_{ij}} \in S \times S | i \neq j, \epsilon_{ij} = \delta_F(f_i, f_j) \right\}. \tag{4}$$

In order to measure the outlier level of f_i, we define

$$OL_i := \sum_{j=1}^{n} (\epsilon_{ij} - \epsilon') \cdot w \tag{5}$$

and

$$w = \begin{cases} 1 & \text{if } \epsilon_{ij} > \epsilon' \text{ and } i \neq j \\ 0 & \text{else,} \end{cases} \tag{6}$$

where ϵ' is the predefined threshold.

We add a slider on top of FSDMatrix which stands for value of ϵ'. When rolls the slider, $(\epsilon_{ij} - \epsilon')$ of each $D_{\epsilon_{ij}}$ will be calculated and the result will be mapped into six kinds of color, and the color legend is shown in

Fig. 1 above FSDMatrix. The redder the background of FSD, the higher the result. At the same time, the order of rows and columns of FSDMatrix$_S$ will be changed based on OL_i in descending order. In this way, f_i with higher OL_i is more likely close to the top left of FSDMatrix$_S$. Therefore, trajectories in the top left corner of FSDMatrix are more likely to be the outliers.

FSDMatrix can automatically present sensible clustering phenomenon along the diagonal line. Take Fig. 1c as an example, in this FSDMatrix, all trajectories can be divided into four groups, from top left to bottom right, which are encoded with different colors, and the top two groups could be viewed as outliers because of their deepness of color.

4.5 Interaction design

Map view can zoom in, zoom out, transform and change the background. Users can select different kinds of trajectories on the top left of M3. The lines on the map stand for trajectories users selected, and they are encoded with different colors based on the communities found in MST. Every trajectory will be highlighted and more information will be shown in the tooltip when the mouse is moved on it. Once MST has been clicked, every inner ball inside outer ring will be triggered and the corresponding trajectories in the map view will be highlighted, and the column header of FSDMatrix will be highlighted too.

In order to distinguish different kinds of trajectories, we add a color palette besides the slider of FSDMatrix. The color palette has different color buttons and one special black button which can be used to switch between "paint mode" and "click mode". M3 will be changed into "paint mode" when users click on any color button. In this mode, the color of inner trajectory nodes and the column headers of FSDMatrix can be changed at will. In this way, users can classify trajectories in light of the result FSDMatrix presented. This procedure can be seen in Fig. 4.

In the FSDMatrix view, users can roll the slider on top of FSDMatrix to change the ϵ' and reorder the rows and columns based on outlier level. At the same time, the order of column headers and row headers will also be changed. Transition animation has been added on the FSDMatrix for tracking the changes. Users can determine whether the clustering phenomenon is stable through the background color. And each column header is linked to the map view and MST view.

4.6 Boundary detection for FSDMatrix

Using dynamic FSDMatrix with just a slider to find the clustering pattern is intuitive, but could be time consuming when analyzing trajectory groups. Therefore, it would be better to automatically detect the boundaries of clusters in FSDMatrix.

Let the index of each row ($\in [1,n]$) and column ($\in [1,n]$) of FSDMatrix starts at the top left corner of FSDMatrix, FSD(i, j) be the FSD at the position of row i and column j, and $\epsilon(i,j)$ be the Fréchet distance of FSD(i, j). Specifically, let FSD(k) be the FSD at the position of row k and column k which is definitely the k FSD along the diagonal line of FSDMatrix, and $\epsilon(k)$ be the Fréchet distance of FSD(k). Then, we define the k-sub-matrix of FSDMatrix M as:

$$\mathrm{SUB}(M)_k := \left\{ \mathrm{FSD}(i,j) \in M | i \in [1,k], j \in [1,k] \right\}. \tag{7}$$

Next, for each $SUB(M)_k (k \in [1,n])$, calculate the portion of blank FSD which is defined as:

$$PB_k := \frac{\sum_{i=1}^{k} [\epsilon(i,j) \leq \epsilon']}{k * k}. \tag{8}$$

Then, the boundaries between clusters are the indexes of the peak values of $P = (PB_1, PB_2, ..., PB_n)$ if any, which can be seen in Fig. 6. If we define the set of indexes of peak values as $I = (I_1, I_2, ..., I_m | m << n, I_1 > 1, I_m < n)$, users can select random one from I to divide set S into two subsets A and $S - A$. Subset A contains outliers and some clusters which are quite different from normal ones. Choosing which I_i as a boundary index depends on users. Surely, $|O| < |A| < |S|$ if we define O as a set of outliers. And we show an application of this algorithm for massive trajectories in case study 2.

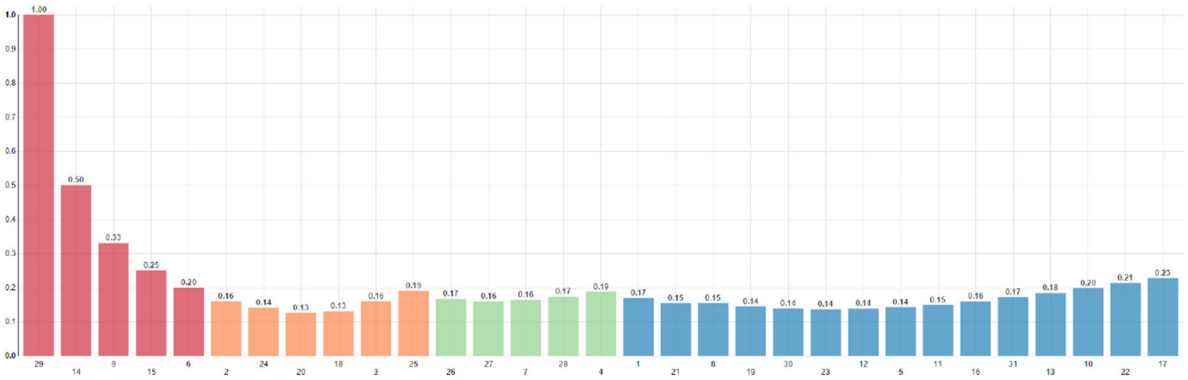

Fig. 6 Bar chart of P. The result of $(PB_1, PB_2, ..., PB_n)$ is shown as a bar chart when ϵ' equals to 2. We can see that the indexes of peak values are just boundaries of clusters in Fig. 1c

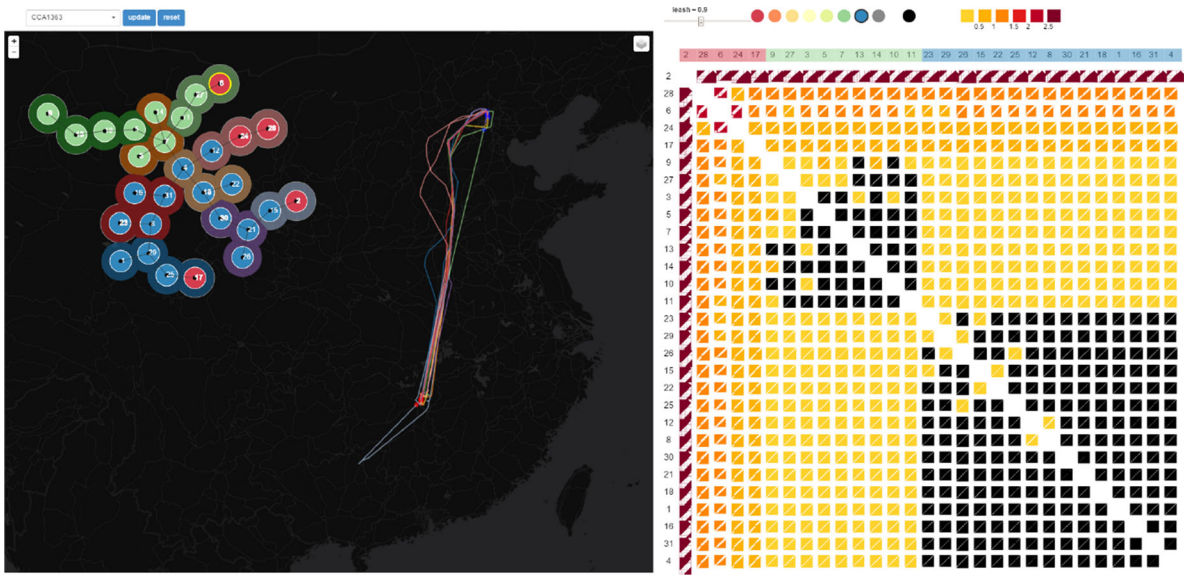

Fig. 7 M3 interface. Based on M3, we build a system to explore trajectories. Three views are integrated together. Map view is on the left, and FSDMatrix view is on the right, while MST view floats over map view

5 Case study

In this section, we show two case studies for flight trajectories. The first takes trajectories of the same flight as an example to demonstrate the pipeline. The second uses boundary detection method to classify massive trajectories and compare the results of different selected flights.

The data sources of the two cases come from FlightAware, one of the world's largest flight tracking data communities. For each point p in one single trajectory, it contains 11 properties, such as TraMark, PointId, Flight, Date, Latitude, Longitude, Direction, Speed, Height, Origin, Destination. But sometimes the value of some properties may be lost in some trajectory data. Therefore, we only extract (Latitude, Longitude) for each point. And every trajectory has the flight and date properties. Then, in order to accelerate calculation of Fréchet distance between two trajectories and rendering for massive trajectories, we use Ramer–Douglas–Peucker algorithm (Poiker and Douglas 1973) to simplify the trajectories.

In the order to explore flights trajectories more conveniently, we build a visual system based on M3 framework. The user interface is shown in Fig. 7.

5.1 Case study 1: exploring AAL140 trajectories

Without loss of generality, we choose flight AAL140, flight from Dallas–Fort Worth to San Jose, America, as an example.

As shown in Fig. 2, after preprocessing and pairwise calculation, we construct the three views. By sliding the slider above the FSDMatrix, the order of rows and columns of FSDMatrix is changed accordingly. And we set $\epsilon' = 2$ since in this case the clustering phenomenon is much more obvious. Then, we use the color palette to color the column headers of FSDMatrix, and then the inner balls of corresponding nodes in MST will be set automatically. The result is presented in Fig. 1b.

In MST, there are seven communities. Trajectories in each community are much similar except the darker orange and darker green ones, which are located at the outmost branches of MST and the trajectories in these two communities are located at the most north and south. Four clusters have been detected by FSDMatrix, colored in red, orange, green and blue ones. Red and orange clusters are located at the upper most left corner of FSDMatrix, and we can see them as outliers, and we can find that those outliers are located exactly at the outmost branches of MST. All trajectories in darker orange community are outliers which can be verified in the map view. And four outliers exist in the darker green community. Furthermore, we can find that the blue cluster is the most inner part in MST and, at the same time, it is the right bottom corner of FSDMatrix that means it is a normal cluster. Moreover, node of number 5 trajectory in the blue cluster is the core node since it has the highest degree. Therefore, we can see it as an representative trajectory of flight AAL140.

From Sect. 4.1, we can learn that each FSD involves two trajectories. We can explore more information in FSDMatrix. The black FSD in FSDMatrix means that the involved two trajectories are more closer or similar, and the free space is much thinner like a needle. We can see from FSDMatrix that all FSDs in black color are flock together and, thus, we can conclude that they all are much more closer or similar. The darker red and orange blocks next to the black blocks of FSDMatrix imply that clusters are much different among them. Furthermore, by looking at the shape of free spaces along one row or column of FSDMatrix, we can easily know the inner relationship of this trajectory with others. Taking the first row of FSDMatrix as an example, the background of each FSD in that row is red or darker red, which means that trajectory 29 is almost totally distinct with the others. Each free space in those FSDs is spindle shaped, where the thinner part stands for the most different parts of two trajectories. Therefore, the most different part between trajectory 29 and the others is near the origin of those trajectories, which is consistent with the map view.

5.2 Case study 2: exploration of flight trajectories in China

In this case, we extract all the flight trajectories whose departure airports and arrival airports are all located in China in July, 2016. This sample contains totally 72,522 trajectories. Since the boundary detection algorithm is sensitive to ϵ', we try some different flights using M3 to find a better parameter. At last, we choose 0.6 as the value of ϵ', since the clustering phenomenon is obvious for most flights at that value. Then, we choose I_1, the first peak value of P as the boundary index to divide S.

After calculation of boundary detection algorithm, we get $|A| = 19{,}923$, which accounts for one-third of total number of trajectories.

We first draw all the trajectories of A as lines in red, then add the rest in blue, and for each trajectory, using fluorescent green to identify it's origin and destination. The result is shown in Fig. 8.

As shown in Fig. 8, the network of airlines' main flight paths in the sky will show themselves in the form of blue lines. And the main airports in China can be viewed at just a glance. Using boundary detection, flight paths can be extracted roughly from the complicated net. To some extent, this method can be used as a sampling, or feature extraction of trajectories data.

Using the result of Fig. 8, we can select flights of interest based on the geographical distribution and outlier level. In Fig. 9, we choose six flights inside China to demonstrate more detail. It can be found that even though the trajectories of each flight are much different in terms of direction, scale, outlier level and so on; their topology of each flight is much similar in MSTs. Outliers of trajectory are mainly located at the outmost branches of MSTs. For such flights as CBJ5756, AMU162 and CAL928 (Fig. 9a–c), outliers are especially apparent in map view and even larger ϵ' can detect them in FSDMatrixs with dark red color. Some may not clear in map view, such as the last three flights in Fig. 9, and they need the help of FSDMatrix with different ϵ'.

Fig. 8 Boundary detection result for flight trajectories in China. Blue lines with opacity = 0.1 form the network of airplane main channels in the sky and red lines with opacity = 0.2 are crossing each other in the grids of the network

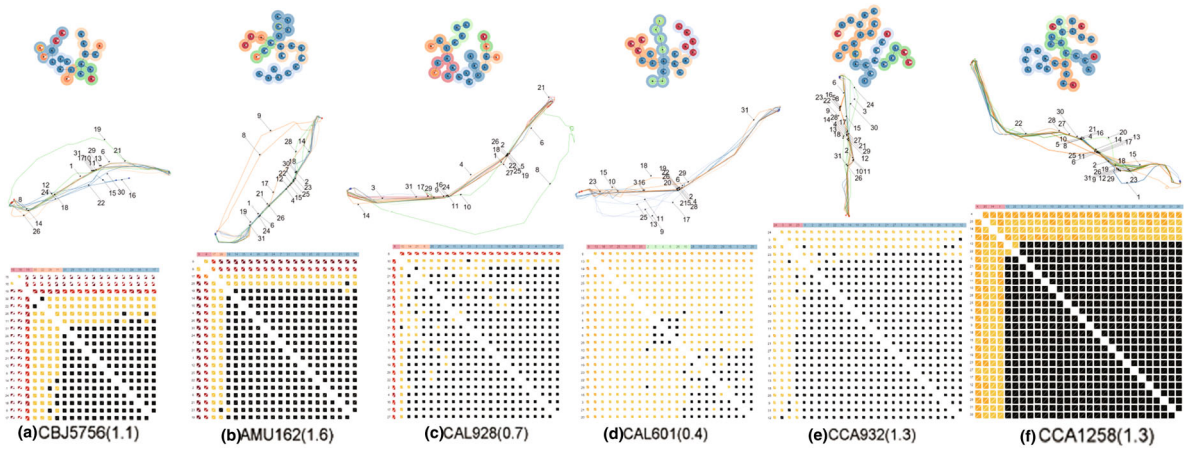

Fig. 9 M3 views of six flights. The communities of each MST are in different colors on the outer rings. The colors of inner balls are based on the outliers and clusters divided by users using FSDMatrix. The red and orange groups are outliers and the first cluster, respectively. Green and blue ones are normal clusters. Flight number and ϵ' of each M3 are written below the FSDMatrixs

6 Conclusion and future work

Outlier detection and clustering are important to analyze trajectory. While many algorithms have been developed to tackle these issues, they lack the combination with visualization to enable the involvement of human intelligence during the analyzing process. Traditional trajectory visualizations do not show the spatial relationships between trajectories. We propose a visual framework called M3, which combines data mining algorithms with visualization technique through three coordinated views: Map, MST, and FSDMatrix. The relationships between trajectories can be found in MST. With FSDMatrix, we can easily discover outliers and clusters of trajectories only through simple interactive operation.

While we showed the applicability of our visual framework, it also has limitations, such as scalability. The scalability issue is mainly caused by the computation of Fréchet distance. When computing Fréchet distance between two trajectories, the computation cost is mainly determined by a matrix, which consists of points of the two trajectories. The average number of points in flight trajectories is about four hundred in our case studies, and that may be even larger for long-time flight. The problem can get worse if there are large

quantities of trajectories to process. To reduce the Fréchet distance cost, we use a method to simplify each trajectory before computing the distance. With this method, we can remove unnecessary points from trajectories without dramatically losing accuracy and, thus, less points take part in the following computation. We believe that our visual framework is more suitable for such scenarios as detecting outliers and finding relationships between trajectories, whose departure airport is a same one and their arrival airport is another same one, such that not so many trajectories should be considered.

In the future, we will extend MST to explore more relationships of trajectories besides spatiality, and some optimizations of M3 should be done to improve scalability, incorporate more visual analytics tools, and put temporal information into consideration.

References

Aghaeepour N, Finak G, Hoos H et al (2013) Critical assessment of automated flow cytometry data analysis techniques. Nat Methods 10(3):228–38

Alt H, Godau M (1995) Computing the Fréchet distance between two polygonal curves. Int J Comput Geom Appl 5:75–91

Alt H, Behrends B, Blömer J (1995) Approximate matching of polygonal shapes (extended abstract). Ann Math Artif Intell 13(3):251–265

Andrienko G, Andrienko N, Fuchs G (2016) Understanding movement data quality. J Locat Based Serv 10(1):31–46

Andrienko N, Andrienko G (2011) Spatial generalization and aggregation of massive movement data. IEEE Trans Vis Comput Graph 17(2):205–19

Andrienko N, Andrienko G, Barrett L, Dostie M, Henzi P (2013) Space transformation for understanding group movement. IEEE Trans Vis Comput Graph 19(12):2169–2178. https://doi.org/10.1109/TVCG.2013.193

Ankerst M, Breunig MM, Kriegel HP, Sander J (1999) Optics: ordering points to identify the clustering structure. In: ACM SIGMOD international conference on management of data. pp 49–60

Buchin M (2010) Constrained Free Space Diagrams: a tool for trajectory analysis. Int J Geogr Inf Sci 24(7):1101–1125

Chang YJ, Hung PY, Newman M (2012) Traceviz: "brushing" for location based services. In: International conference on human–computer interaction with mobile devices and services ACM. pp 345–348

Eiter T, Mannila H (1994) Computing discrete Fréchet distance. Tech. Rep. CD-TR94/64, Information Systems Department, Technical University of Vienna

Ester M, Kriegel HP, Sander J, Xu X (1996) A density-based algorithm for discovering clusters in large spatial databases with noise. In: Conf on knowledge discovery and data mining. pp 226–231

Fruchterman TMJ, Reingold EM (1991) Graph drawing by force-directed placement. Softw Pract Exp 21(11):1129–1164

Girvan M, Newman MEJ (2002) Community structure in social and biological networks. Proc Natl Acad Sci USA 99(12):7821–6

Guo H, Wang Z, Yu B et al (2011) Tripvista: triple perspective visual trajectory analytics and its application on microscopic traffic data at a road intersection. IEEE Pac Vis Symp PACIFICVIS 2011:163–170

Hurter C, Tissoires B, Conversy S (2008) Fromdady: spreading aircraft trajectories across views to support iterative queries. IEEE Trans Vis Comput Graph 15(6):1017–1024

Knorr EM, Ng RT, Tucakov V (2000) Distance-based outliers: algorithms and applications. VLDB J 8(3):237–253

Kraak MJ (2003) The space-time cube revisited from a geovisualization perspective. In: Proceedings of the 21st international cartographic conference

Krüger R, Thom D, Wörner M, Bosch H, Ertl T (2013) Trajectorylenses—a set-based filtering and exploration technique for long-term trajectory data. In: Proceedings of the 15th Eurographics conference on visualization, The Eurographs Association & Wiley, Chichester, UK, EuroVis '13. pp 451–460

Lee JG, Han J, Whang KY (2007) Trajectory clustering: a partition-and-group framework. In: ACM SIGMOD international conference on management of data. pp 593–604

Lee JG, Han J, Li X (2008) Trajectory outlier detection: a partition-and-detect framework. In: Data engineering, 2008. ICDE 2008. IEEE 24th international conference on, IEEE. pp 140–149

Li X, Han J, Kim S, Gonzalez H (2007) Roam: rule and motif-based anomaly detection in massive moving object data sets. In: 7th SIAM int'l conf on data mining

Lloyd S (1982) Least squares quantization in pcm. IEEE Trans Inf Theory 28(2):129–137

Pettie S, Ramachandran V (2000) An optimal minimum spanning tree algorithm. J ACM 49(1):16–34. https://doi.org/10.1145/505241.505243

Poiker T, Douglas DH (1973) Reflection essay: algorithms for the reduction of the number of points required to represent a digitized line or its caricature. Cartogr Int J Geogr Inf Geovis 10(2):112–122

Prim RC (1957) Shortest connection networks and some generalizations. Bell Labs Tech J 36(6):1389–1401

Qiu P, Simonds EF, Bendall SC et al (2011) Extracting a cellular hierarchy from high-dimensional cytometry data with spade. Nat Biotechnol 29(10):886–91

Scheepens R, Hurter C, Van De WH (2016) Visualization, selection, and analysis of traffic flows. Vis Comput Graph IEEE Trans 22(1):379–388

Senechal M (1993) Spatial tessellations: concepts and applications of voronoi diagrams. Science 260(5111):1170–1173

Van Gassen S, Callebaut B, Van Helden MJ et al (2015) Flowsom: using self-organizing maps for visualization and interpretation of cytometry data. Cytom Part A 87(7):636–645

Wang W, Yang J, Muntz RR (1997) Sting: a statistical information grid approach to spatial data mining. In: 23rd int'l conf on very large data bases, Athens, Greece. pp 186–195

Wang Z, Ye T, Lu M, Yuan X, Qu H, Yuan J, Wu Q (2014) Visual exploration of sparse traffic trajectory data. IEEE Trans Vis Comput Graph 20(12):1813–1822. https://doi.org/10.1109/TVCG.2014.2346746

Wu W, Xu J, Zeng H, Zheng Y, Qu H, Ni B, Yuan M, Ni LM (2016) Telcovis: visual exploration of co-occurrence in urban human mobility based on telco data. IEEE Trans Vis Comput Graph 22(1):935–944. https://doi.org/10.1109/TVCG.2015.2467194

Zhang D, Li N, Zhou ZH, et al (2011) ibat: detecting anomalous taxi trajectories from gps traces. In: UBICOMP 2011: ubiquitous computing, international conference. pp 99–108

Zhang T, Ramakrishnan R, Livny M (1996) Birch: an efficient data clustering method for very large databases. In: ACM SIGMOD int'l conf on management of data. pp 103–114