Trajectory Simplification Algorithm based on Structure Features

Mingjun Zhu

China University of Mining and Technology

Abstract: With the extensive use of location based devices, trajectories of various kind of moving objects can be collected. As time going on, the amount of trajectory data increases exponentially, which brings a series of problems in storage, transmission and analysis. Current trajectory compression algorithms mainly focus on position preserving, compress ratio and run efficiency, but neglect the movement features in trajectories. In this paper, we propose a novel three-stage trajectory compression algorithm based on moving direction of objects, internal fluctuation in trajectories and trajectory velocity, which takes full account of movement pattern and structure features in trajectories. Firstly, the raw trajectory is compressed based on moving direction and the velocity of the object. Then, the trajectory is further simplified according to internal fluctuation in raw trajectory. Comprehensive experiments on real dataset show that: not only the efficiency and effectiveness of the proposed work is better, but also the reservation of local movement features of moving objects and internal characteristic information in trajectories is more detailed.

Keywords: GPS trajectory, data compression, velocity corner, velocity value, movement feature

1. Introduction

In recent years, with the rapid growth of GPS-equipped mobile devices, sensor network and wireless communication technologies, various kinds of moving objects can be traced all over the world. These data are the foundation for us to analyze activities and patterns for moving objects. However, the popularity of these devices and technologies has leading to an exponential growth in the amount of trajectory data as time going on. For instance, there are 5000 taxis in a city and we track the trajectory of each taxi by sampling its position once every 5 seconds, so we will overwhelm 2 GB of storage capacity for a single day to store these data. Such enormous volume of data has brought several problems [1] in transmission, computation, storage and so on.

To overcome these difficulties, it is necessary to find some efficient and effective compression methods to solve the following several tasks [1, 2]. Firstly, compression methods should reduce the data space and make trajectory easy to store and transfer. Secondly, compression methods may remove redundant information and reserve useful information of trajectory data which will make it easy for deeply analyzing trajectory data. Therefore, the aim of data compression is to minimize the volume of data while ensuring not losing important features by some strategies. For moving objects trajectories, it is essential to preserve as much features, including position, direction, corner and velocity as possible while removing redundant and trivial sampling points.

Currently, a number of trajectory compression algorithms have been studied. In many researches, the main idea of line simplification is widely used to reduce the number of trajectory points by introducing a bounded error, which loses some information after compression [2, 3]. This kind of line simplification is mainly derived from the well-known Douglas-Peucker (DP) algorithm [4], which makes use of the divide-and-conquer approach to keep the most important points of a polyline. In order to take both spatial and temporal dimension into account, Meratnia et al. [3] replace the perpendicular Euclidean distance with Synchronous Euclidean Distance (SED) in DP

algorithm, with which, compressed data is confirmed be superiority than the former ones. Besides DP algorithm, there are also various trajectory compression algorithm exists in the literature. Each offers a different trade off among compression time, compression ratio, and accuracy. Uniform sampling, which is fast and can archive the specified compression ratio by sampling trajectory at fixed time interval, but introduce large spatial and SED errors. To-Down Time Ratio (TD-TR) algorithm [3], is a variant of DP algorithm with SED instead of spatial error. It's running time is $O(n^2)$. Opening Window (OW) algorithm [5] is an online approximate line simplification algorithm by introducing a slide window. OW algorithm runs with the window anchored at the first point, and gradually checks the forthcoming points until the spatial error is greater than the given threshold. The spatial error is the distance of the point to the line segment between the first point and the last point in the window. Then it executes iteratively until the last point of trajectory is included. The running time of OW algorithm is $O(n^2)$. Opening Window Time Ratio (OW-TR) algorithm [3] is an extension to OW algorithm which takes temporal data into account and uses SED to represent the error. Like OW algorithm, the worst running time of OW-TR is $O(n^2)$. Dead Reckoning (DR) algorithm [6] is an efficient compression algorithm that considers not only spatial dimension but also velocity information. DR algorithm firstly marks the start point p_0 as the key point, and stores p_0 and its velocity in the compressed representation. Then the next point p_i is estimated whether it's location within the SED threshold from p_0. If true then continue the next point of p_i, else p_i is marked as the key point and stored to the compressed representation with its velocity. The DR algorithm will execute iteratively to the end of trajectory. The computation complexity of DR algorithm is $O(n)$.

All these algorithms take the spatial and temporal information as the basis to reduce points in trajectory, and do not take trajectory movement patterns and internal features into consideration. Due to the trivialness and redundancy of trajectory data, almost all trajectory compress algorithms are lossy. When we query trajectories from database or discover the hidden knowledge from trajectories, we hope the compressed trajectory can represent their raw ones well. However, if the movement pattern and internal features are neglected, applications, such as trajectory clustering [7, 8], outlier detection [9] and activity discovery [10] may be not so accuracy as we expected.

In literature [7], Lee pointed out the important attributes for trajectory clustering, and in our previous work [8], we gave the formal definitions on trajectory structure. In general, trajectory structure feature can be derived mainly by the corner and velocity at sampling points.

In order to solve the problem mentioned above, we present a novel trajectory compression algorithm based on corner and velocity, with which, trajectory movement patterns and internal features can be retained when compressing trajectories. Therefore, a two-phase compression algorithm is proposed in this paper, which called Trajectory Simplification Algorithm based on Structure Features (SF).

To summarize, the main contributions of this paper are as follows:

1) This paper firstly compresses trajectories based on moving direction of objects, which can better keep the outline geometrical characters of trajectories.

2) Then, the algorithm proposed in this paper simplifies trajectories according to internal fluctuations in trajectories, which will better keep the movement pattern and structure features in trajectories.

3) Finally, this algorithm compresses trajectories by trajectory velocity, which can better keep the movement pattern in trajectories.

4) To verify the performance of SF, we carry out a comprehensive comparison with other algorithms such as DP and TD-SB.

The rest of this paper is organized as follows. Section 2 introduces the related work. Section 3 describes our compression motivation and related definitions. In section 4, the compression algorithm SF is introduced in detail. An evaluation of SF and other algorithms is provided in section 5. Finally, Section 6 draws conclusions and points out some possible research opportunities.

2. Related works

The rapid development of various subjects and the wide usage of Internet provide a great deal of technical supports and a powerful motivation for the rapid development of trajectory data compression technologies. The existing compression methods can be classified into 3 categories, according to their compression ideas.

1) Distance-based trajectory compression

Many researchers have devoted their talent to compress trajectories by deciding whether the sampling point is reserved based on distance (such as perpendicular distance, Synchronized Euclidean distance and so on) since 1973. In literature [5], Douglas and Peucker proposed an algorithm called Douglas-Peucker algorithm, which recursively selects the point whose perpendicular distance is greater than given threshold until all points reserved meet the condition. Its advantage is the translation and rotation invariance, namely, when the trajectory and threshold have been given, the compression result is certain. However, there is an apparent drawback about Douglas-Peucker (DP) algorithm, which only considered spatial information, that is because temporal information also contain in trajectory data. In order to overcome this shortcoming, Meratnia et al. put forward a top-down time-ratio algorithm (TD-TR) which is a transformation of DP algorithm taking a full consideration of spatiotemporal characteristics by replacing perpendicular distance with SED distance [3,6]. This method has a higher accuracy than DP algorithm and also has the advantage of translation and rotation invariance. Both DP and TD-TR are not suitable for real-time applications, so Jonathan Muckell proposed the Spatial QUalIty Simplification Heuristic (SQUISH) method based on the priority queue data structure, which prioritizes the most important points in a trajectory stream [24]. Three years later, Muckell proposed a new version of SQUISH, called SQUISH-E (Spatial QUalIty Simplification Heuristic - Extended), which has the flexibility of tuning compression with respect to compression ratio and error [25].

2) Velocity-based trajectory compression

The researches on compressing trajectory data based on velocity are not perfect by now. A famous velocity-based trajectory compression is top-down speed-based algorithm proposed by Meratnia[3]. The algorithm improved the existing compression techniques by exploiting the spatiotemporal information hiding in the time series, which can be made by analyzing the derived speeds at subsequent of the trajectory [3]. It is trivial to implement, but the accuracy is lower than DP and TD-TR algorithm. An online algorithm called Dead Reckoning algorithm proposed by Trajcevski[26] compressed trajectory by estimating the successor point through the current point and its velocity. It has a high execution efficiency for the computational complexity $O(n)$. And the primary disadvantages are that it tends to achieve lower compression ratios than other techniques introduced in this section and it does not allow users to set the target compression ratio.

3) Semantic-based trajectory compression

Considering the different environment where objects move, compressing trajectory in road network has attracted many attentions [18-22]. Schmid and Richter proposed a new and novel representation for trajectories that replaces trajectory data by the form of semantic information in road network [8]. Zheng proposed a new framework, namely paralleled road-network-based

trajectory compression, to effectively compress trajectory data under road network constraints [7]. PRESS proposed a novel representation for trajectories to separate the spatial representation of a trajectory from the temporal representation and proposed a Hybrid Spatial Compression (HSC) algorithm and error Bounded Temporal Compression (BTC) algorithm to compress the spatial and temporal information of trajectories respectively.

Although these methods in literatures [9-14, 23] have a high computing performance, they only considered the outline geometrical characteristics of trajectories and ignore the movement pattern and structure features in trajectories, such as moving direction of objects, internal fluctuation in trajectories as well as trajectory velocity or acceleration and so on. However, in reality, it is essential for trajectory compression to reserve moving objects' movement pattern and structure features in trajectories, which is useful in data mining field, e.g. studying animals' migratory traces, behavior and living situations, as well as animal migration research and hurricanes, tornados and ocean currents prediction.

To tackle the above disadvantages, this paper proposes a new compression algorithm based on the structure features of trajectories (SF algorithm). At the end of this paper, we analyze the influence of various parameters on SF algorithm proposed in this paper and compared SF algorithm with other algorithms. The experimental results show that it can effectively compress trajectory data and keep movement pattern and structure features in trajectories, especially when the motion of a moving object is frequently changing in a certain area. The compression results of SF algorithm reserve more information (e.g. corner information, movement pattern and other structure features) compared with other algorithms.

3. Motivation and related definitions

3.1 Motivation

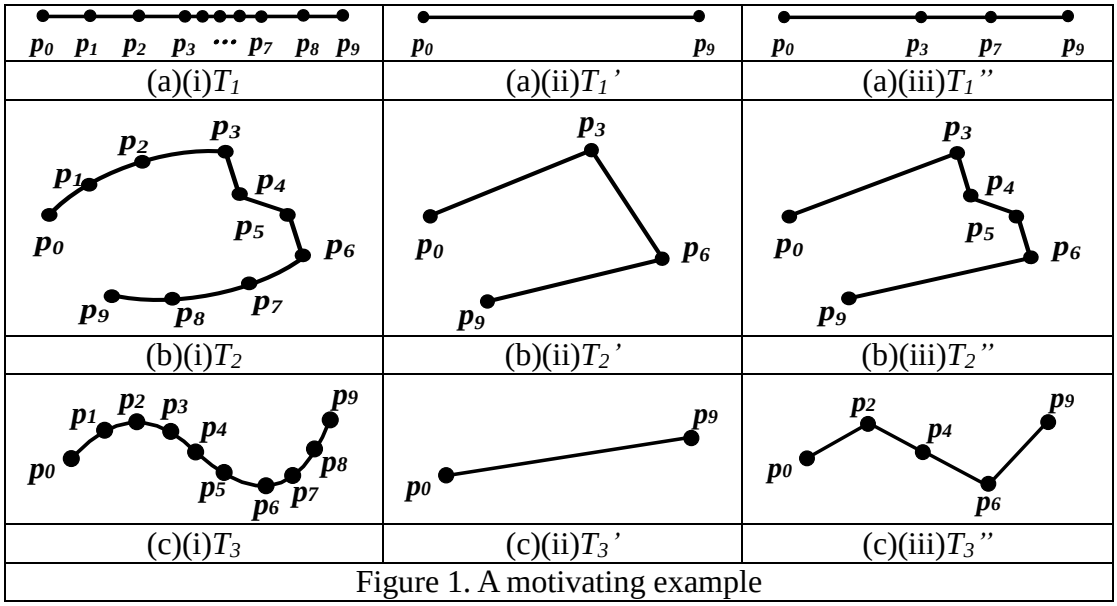
In many location-based studies and applications, trajectories are viewed as the sequence of sampling points, and the movement pattern as well as internal features are often neglected. Therefore, traditional compression algorithms always extensively pursuit the compression ratio by trading off compression time and accuracy. In literature [11], this kind of algorithms is called position-preserving trajectory simplification (PPTS) algorithms. However, these position-preserving algorithms may not suitable for many situations.

In order to preserve the movement pattern and internal features, some encoding-based algorithms are mentioned in the Master Degree Thesis of Xiaoying Liu [12], such as Huffman Coding, ZIP, LMZA2, can compress trajectories without any loss and preserve detailed trajectory information. However, encoding and decoding processes themselves are also time consuming and memory increasing. Moreover, when we query trajectory data from moving object database, it is quite difficult to find useful information from coded data. And after decoding, we still have to face the large amount of redundant trajectory data. Therefore, encoding-based algorithms are not suitable for trajectory compression.

To illustrate this motivation, we give an example to discuss the movement pattern and internal features in trajectories.

Given three raw trajectories T_1, T_2 and T_3 as shown in Figure 1 (a)(i), (b)(i) and (c)(i) respectively. Each trajectory has 10 points (p_0, p_1, …, p_9), and all points are sampled with fixed interval. T_1 is a straight trajectory, and if an existing PPTS algorithm with SED as error bound is used to compress T_1, we can get a simplified version of T_1 as T_1' shown in Figure 1 (a)(ii). However, as we can see that the velocity of p_0 to p_3 and p_7 to p_9 is quite different from that of p_3 to p_7. In a city transport system, the part p_3 to p_7 may be more important than others in data analysis. Therefore, we should

present a new technique to preserve this part as T_1'' shown in Figure 1 (a)(iii). T_2 is a circular trajectory with two smooth parts p_0 to p_3, p_6 to p_9 and a fluctuant part p_3 to p_6. We can get a simplified trajectory as T_2' shown in Figure 1 (b)(ii) with existing PPTS algorithms. However, the fluctuant part is lost in T_2', and this part may be crucial in moving object activity discovery [10]. Therefore, this part should be preserved in compression process as T_2'' in Figure 1 (b)(iii) for deep analysis. T_3 is a curve trajectory with two semicircular parts p_0 to p_4 and p_5 to p_9. Based on existing PPTS algorithms, we can get a simplified trajectory as T_3' shown in Figure 1 (c)(ii). However, the semicircular part is lost in T_3', and this part may be also crucial in moving object activity discovery. Therefore, this part should be compressed as T_3'' in Figure 1 (c)(iii) for not losing useful information.



(a)(i)T_1 (a)(ii)T_1' (a)(iii)T_1''

(b)(i)T_2 (b)(ii)T_2' (b)(iii)T_2''

(c)(i)T_3 (c)(ii)T_3' (c)(iii)T_3''

Figure 1. A motivating example

Therefore, with this idea, we propose a novel trajectory compression algorithm based on structure features to preserve as much movement pattern and internal features as possible.

3.2 Related definitions

In order to formally describe the proposed compression algorithm, we firstly give some definitions on related concepts such as trajectory corner and velocity. The definition of trajectory given in our previous work [8] is also available in this paper. *TD* (*T*rajectory *D*atabase) denotes trajectory set $TD=\{TR_1, TR_2, \ldots, TR_n\}$, and TR_i is the *i*-th trajectory. A *Tr*ajectory is a chronological sequence consisted of multi-dimensional locations, which is denoted by $TR_i= \{P_1, P_2, \ldots, P_m\}(1\leq i\leq n)$. $P_j(1\leq j\leq m)$, a sampling point in TR_i, is represented as $<Location_j, T_j>$, which means that the position of the moving object is $Location_j$ at time T_j. $Location_j$ is a multi-dimensional location point, for instance, (x_j, y_j) is a 2-dimensional location point.

As aforementioned, corner and velocity are two important structure attributes of trajectories, and they are used as the compression metrics for deciding whether the points should be persevered or not. For convenience, we introduce symbols V_A as the arriving speed and V_L as the leaving speed at each point. The definitions on trajectory corner and velocity as follows.

Definition 1. Trajectory Corner [8]: the turn angle θ of two adjacent trajectory segments reflects the moving tendency at the sampling point *p*.

As shown in Figure 2, the included angle at sampling points between adjacent segments is denoted as α, and the turn angle in the moving direction marked with θ_i and θ_{n-1} are two trajectory

corners. In order to simplify the calculation, the clockwise corner (θ_i) is marked as positive and the anticlockwise corner (θ_{n-1}) is marked as negative.

In Figure 2, trajectory segment *a* and *b* are two adjacent edges with angle α represented as vector $\vec{a}$ and $\vec{b}$ respectively, and *c* is the virtual opposite edge of angle α represented as vector $\vec{c}$. Then angle α_i at p_i can be calculated by formula (1):

$$\alpha_i = \arccos\left(\frac{|\vec{a}|^2+|\vec{b}|^2-|\vec{c}|^2}{2\cdot|\vec{a}|\cdot|\vec{b}|}\right) \quad (1)$$

Here, $|\vec{a}|$, $|\vec{b}|$ and $|\vec{c}|$ are the length of trajectory segments. Trajectory segment vectors $\vec{a}$ and $\vec{b}$ are two directed adjacent edges with angle α and can be denoted as $\vec{a} = (x_i - x_{i-1}, y_i - y_{i-1})$ and $\vec{b} = (x_{i+1} - x_i, y_{i+1} - y_i)$. Segment vector $\vec{c}$ is the virtual opposite edge of angle α, denoted as $\vec{c} = (x_{i+1} - x_{i-1}, y_{i+1} - y_{i-1})$.The trajectory corner θ_i at p_i can be calculated by formula (2):

$$\theta_i = \begin{cases} \pi - \alpha_i, if(\vec{a} \times \vec{b} \leq 0) \\ \alpha_i - \pi, if(\vec{a} \times \vec{b} > 0) \end{cases} \quad (2)$$

Based on trajectory corner, we give the definition on movement direction and internal fluctuation to decompose the trajectory structure features.

Definition 2. Movement Direction: For a given trajectory, moving direction is represented by the accumulation of turn angles at each sampling points which can reflects the moving tendency of an object.

We use D_θ to denote the movement pattern of a trajectory. According to Definition 2, D_θ is an accumulation value of turn angles. In the trajectory simplification algorithms of this paper, turn angles at each sampling points are calculated and accumulated to D_θ sequentially. Once D_θ is greater than the given threshold, and we can say that movement direction changes great and the point where direction changes should be marked as a candidate point to be preserved, then the D_θ should be reset and newly accumulated from the next point. The formula of D_θ is given as: $\boldsymbol{D_\theta = \sum \theta_i}$.

Definition 3. Internal Fluctuation: For a given trajectory *T*, the internal fluctuation is that there exists several continuous points, at which trajectory corners change sharply, denoted by $F_{\varepsilon,k}$ (ε is the threshold of corner, and *k* is the threshold of continuous points). $F_{\varepsilon,k}$ means that there are at least *k* continuous points, where trajectory corners greater than ε.

In T_2 of Figure 2(b), corners at sampling points from p_3 to p_6 change greater than others. Therefore, we can infer that there are something happened at trajectory segment p_3 to p_6. Traditional simplification algorithms may neglect this part and remove these points making internal features lost. In order to avoid noise distortion, *k* is often set greater than 2.

Trajectories are the discretized sampling points with location and timestamp. Therefore, the velocity information associated with trajectories are often denoted by their average value. In order to describe the instantaneous moving tendency approximately, we derive two speed vector from trajectories. One is called arriving speed and another is called leaving speed.

Definition 4. Arriving Speed: For a given point p_i ($1<i\leq n$, *n* is the length of the trajectory), the arriving speed is the average speed that arrives to p_i, denoted by V_A. Its computational representation is the mean speed of the closet segment before p_i, shown as formula (3).

$$V_{Ai} = \frac{\sqrt{(x_i-x_{i-1})^2+(y_i-y_{i-1})^2}}{t_i-t_{i-1}} \quad (3)$$

Definition 5. Leaving Speed: For a given point p_i ($1\leq i<n$, *n* is the length of the trajectory), the leaving speed is the average speed that leaves from p_i, denoted by V_L. Its computational representation is the mean speed of the closet segment after p_i, shown as formula (4).

$$V_{Li} = \frac{\sqrt{(x_{i+1}-x_i)^2+(y_{i+1}-y_i)^2}}{t_{i+1}-t_i} \quad (4)$$

The arriving speed (V_A) and leaving speed (V_L) at p_i are shown in Figure 2. Note that, the arriving speed at p_1 and the leaving speed at p_n are 0, for there are no succeed point of p_n and no precursor point of p_1.

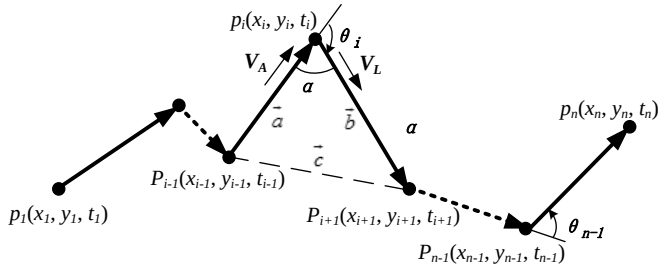


Figure 3. An example of trajectory corner and velocity

With the combination of V_A and V_L, we can easily analysis the motion characteristics of moving objects in a certain area at certain time. For example, if at some points, V_A is smaller than V_L, then we can know that the moving object speeds up at these points, otherwise, we can say the moving object slows down. Therefore, we give the definition on the velocity.

Definition 6. Trajectory Velocity: The speed deviation of two adjacent trajectory segments reflects the motion characteristics at the sampling point p.

We use V_i as speed deviation at sampling point p_i ($1 \le i \le n$), and $V_i = \text{abs}(V_{Ai} - V_{Li})$. The function abs() is used to calculate the absolute value of speed deviation, because we just need to identify at which trajectory point the speed changes great. In Figure 1 (a)(i), we can see at points p_0, p_3, p_7 and p_9, speed values change greater than those of others, so these points should be preserved in (a)(iii) according to the idea of our algorithm.

Definition 7. Trajectory Deviation Angle: For a given trajectory T and compressed trajectory T_c, the trajectory deviation angle between raw trajectory segments and compressed trajectory segments reflects the offset errors of compressed trajectory.

We use γ_i to denote trajectory deviation angle at sampling point p_i ($1 \le i \le n$). As shown in Figure 3, p_j, p_{j+1} and p_n are the points in compressed trajectory, as well as p_i is the ith point in raw trajectory. Therefore, γ_i can be calculated by formula (5).

$$\gamma_i = \arccos\left(\frac{|\overrightarrow{p_j p_{j+1}}|^2 + |\overrightarrow{p_j p_i}|^2 - |\overrightarrow{p_i p_{j+1}}|^2}{2 \cdot |\overrightarrow{p_j p_{j+1}}| \cdot |\overrightarrow{p_j p_i}|}\right) \quad (5)$$

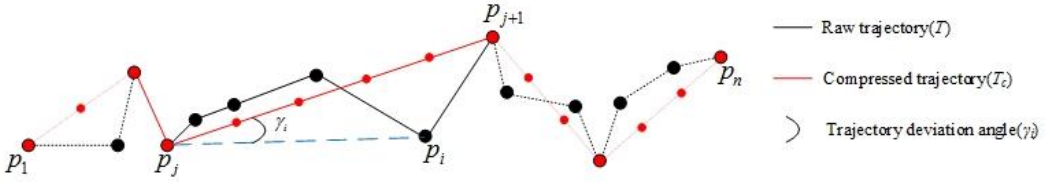


Figure 4. An example of trajectory deviation angle

4. Trajectory Simplification Algorithm based on Structure Features

Existing trajectory compression methods pay too much attention on efficiency, compression ratio as well as run time, and ignore movement pattern and internal features of trajectories. Therefore, in order to pursue high efficiency, it is easy to lose the internal information. The SF algorithm proposed in this paper is a new breakthrough to traditional ones, and pays more attention on preserving trajectory movement pattern and internal features while removing some trivial and redundant points. SF algorithm which includes two phases. Firstly, the raw trajectory is

compressed based on moving direction and the velocity of the object. Then, the trajectory is further simplified according to internal fluctuation in raw trajectory.

In the process of simplifying the trajectory, we refer to the OW [5] algorithm and do appropriate modification to make it more applicable for our SF algorithm.

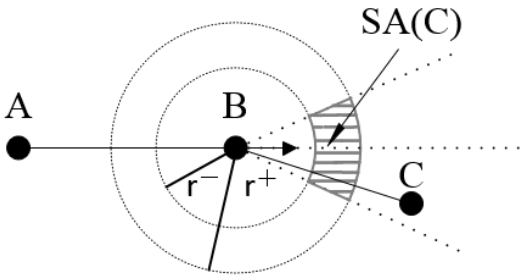


Figure 4 An example of trajectory simplification

As shown in figure 4, $\overrightarrow{AB}$ denotes the moving direction of trajectory. We definite the minimum allowed speed $v^{-}=V\times(1-v)$ and the maximum allowed speed $v^{+}=V\times(1+v)$. Where, V is the trajectory velocity and v is the threshold of trajectory velocity. We sample a point every t interval in the trajectory points sampling. According to the simple physical equation, we can get distance using velocity and time. Therefore, we can get the radius of inner circle $r^{-}=t\times v^{-}$ and the radius of outer circle $r^{+}=t\times v^{+}$. We can draw a concentric circle at point B with r^{-} and r^{+}. Later, we draw a sector based on the direction threshold in the direction of $\overrightarrow{AB}$. We call this sector *safe area*, *SA* in short. If the next point in the *SA*, we will reserve it. Otherwise, we delete it. Now, we only consider the elements of velocity and direction but not take internal fluctuation into consideration. In definition 3, we set ε as the threshold of corner. $F_{\varepsilon,k}$ means that there are at least k continuous points, where trajectory corners greater than ε. In order to avoid noise distortion, k is often set greater than 2. So, if $k>2$, even this point in the *SA*, we still reserve it. In figure 4, point *C* is not in the *SA(C)* and is not the kth continuous point to be greater than the threshold. So, we delete it.

In order to better introduce the algorithm proposed in this paper, it is necessary for a formal description of the symbols used in section 4. The symbols used in SF algorithm and their meaning are summarized in Table 1.

Table 1 Parameters and their meaning

Parameter	Meaning
T	Raw trajectory.
T_c	Trajectory compressed by SF algorithm.
$\|\bullet\|$	The absolute value of •.
k	The counter of internal fluctuation.
p_i	The i-th point in a trajectory, $p_i=<x_i, y_i, t_i>$.
β	moving direction threshold.
δ	internal fluctuation threshold.
v	trajectory velocity threshold.

4.1 Algorithm description

In order to keep the moving characteristics and the internal characteristics information in trajectories, SF algorithm removes the redundant trajectory points based on the structure features of trajectories, such as moving direction of moving objects, internal fluctuation in trajectories and trajectory velocity. SF algorithm is a two-phase algorithm, firstly, it reserves the points according

to the appropriate modified OW algorithm (Lines 01-05 and 08-11); then, it reserves the internal fluctuations in raw trajectory, while meeting the condition of internal fluctuation (Lines 06-07). This algorithm will be end until all of the points in raw trajectory have been processed.

Algorithm: Trajectory Simplification Algorithm based on Structure Features (SF)

Input: Raw trajectory (T), moving direction threshold (β), internal fluctuation threshold (δ), trajectory velocity threshold (v)
Output: Compressed trajectory (T_c)
01) $k \leftarrow 0$; // Set the counter of internal fluctuation k is 0
02) for each $p_i \in T$ do
03) calculate $SA(p_i)$; //Calculate the safe area with moving direction threshold (β) and trajectory velocity threshold (v)
04) if p_i in $SA(p_i)$ then
05) k++;
06) if $k > 2$ then
07) $T_c \leftarrow$ the coordinate information of p_i; //Save coordinate information of p_i in T_c, when the value of k is greater than 2
08) else
09) $T_c \leftarrow$ the coordinate information of p_i;
10) $k \leftarrow 0$; // Set the counter of internal fluctuation k is 0
11) end for
end

4.2 Discussion

The moving direction threshold (β), internal fluctuation threshold (δ) and trajectory velocity threshold (v) in SF algorithm are main parameters which affect computational cost, compression ratio and matching effect of algorithms. The setting of compression threshold β, δ and v in SF algorithm should combine statistical learning theories and specific application fields. The higher β is set, the more important features in trajectories will be lost and the worse the holistically fitting effect of trajectories will be, while the lower β is set, the more trajectory mutations or exceptions caused by sampling frequency and equipment error will be kept as well as the lower the compression ratio will be. Similar to β, the higher δ is set, the more movement pattern and structure features will be lost, while the lower δ is set, the more redundant points will be reserved. And, the higher v is set, the more important features and local motion characteristics in trajectories will be lost, while the lower v is set, the more trajectory mutations or exceptions caused by sampling frequency and equipment error will be kept. The computational complexity of SF algorithm is $O(n)$, where n is the number of points in the trajectory.

5. Experiments and analysis

In order to validate the algorithm proposed in this paper, a trajectory data analysis system (TrajMiner) is developed, using Microsoft Visual Studio .Net 2008. The environment of experiments includes: Windows 7, Intel(R) Core(TM) i5-3470 3.20GHz CPU with 4G Ram. The data set stored in Microsoft SQL Server 2008 R2 is GeoLife which includes 8890 trajectories consist of 23860589 sampling points. Longitude, Latitude, and sampling time are extracted from the GeoLife data set to facilitate a clean comparison of the different algorithms.

5.1 Parameter estimation

For the algorithm given in this paper, only 3 parameters are required, including moving direction

threshold (β), internal fluctuation threshold (δ) and trajectory velocity threshold (v). The guidence of β_{rms}, δ_{rms} and v_{rms} can be calculated by formula (6), (7) and (8), for the root mean square (RMS) is a kind of numerical indicators measuring accuracy of measurement. The guideline value of β_{rms}, δ_{rms} and v_{rms} are not the absolute thresholds and the actual thresholds need to adjust the guideline values by combining statistical learning theories and specific application fields as well as the practical experiences of experts in the certain field.

$$\beta_{rms} = \sqrt{\frac{1}{n-2} \times \sum_{i=2}^{n-1} \gamma_i^2} \tag{6}$$

$$\delta_{rms} = \sqrt{\frac{1}{n-2} \times \sum_{i=2}^{n-1} \theta_i^2} \tag{7}$$

$$v_{rms} = \sqrt{\frac{1}{n-2} \times \sum_{i=2}^{n-1} V_i^2} \tag{8}$$

This paper verifies the performance of SF algorithm by compressing 10 trajectories with different parameters. As shown in Table 2, the compression time of SF only depends on the size of trajectory data. The setting of β, δ and v don't have a significant impact on the running speed of SF algorithm, but they have a significant impact on the compression ratio and fitting effect.

Table 2 Comparison and performance analysis of SF algorithm with different parameters

Trajectory ID	β	δ	v	Trajectory size	Points of compressed trajectory	Time cost (ms)	Compression ratio (%)
#1	π/6	2π/3	2	2359	331	2.5884	85.97
	π/3	2π/3	2		285	2.4708	87.92
	π/6	3π/4	2		331	2.5884	85.97
	π/6	2π/3	1		638	2.4362	72.95
	π/12	2π/3	2		381	2.4375	83.85
	π/6	5π/6	2		331	2.5884	85.97
	π/6	2π/3	4		235	2.4503	90.04
#2	π/6	2π/3	2	11225	4529	12.3894	59.65
	π/3	2π/3	2		3169	12.2380	71.77
	π/6	3π/4	2		4512	11.7907	59.80
	π/6	2π/3	1		5511	12.1033	50.90
	π/12	2π/3	2		5998	12.4000	46.57
	π/6	5π/6	2		4502	11.8193	59.89
	π/6	2π/3	4		4044	11.7552	63.97
#3	π/6	2π/3	2	20429	5169	19.7831	74.70
	π/3	2π/3	2		3192	18.4062	84.38
	π/6	3π/4	2		5155	18.2745	74.77
	π/6	2π/3	1		6228	19.1605	69.51
	π/12	2π/3	2		7962	19.9240	61.03
	π/6	5π/6	2		5155	18.2745	74.77
	π/6	2π/3	4		4859	18.2193	76.22
#4	π/6	2π/3	2	30045	15786	33.7668	47.46
	π/3	2π/3	2		10864	32.6536	63.84

	π/6	3π/4	2		15733	32.2145	47.64
	π/6	2π/3	1		17325	32.5335	42.34
	π/12	2π/3	2		20292	33.8998	32.46
	π/6	5π/6	2		15682	31.0957	47.80
	π/6	2π/3	4		15033	31.0235	49.97
#5	π/6	2π/3	2	40570	13606	39.0404	66.46
	π/3	2π/3	2		8290	38.3921	79.57
	π/6	3π/4	2		13590	38.2782	66.50
	π/6	2π/3	1		14883	38.3669	63.32
	π/12	2π/3	2		20462	40.4066	49.56
	π/6	5π/6	2		13572	38.1535	66.55
	π/6	2π/3	4		13281	38.0535	67.26
#6	π/6	2π/3	2	51303	14432	50.5754	71.87
	π/3	2π/3	2		8149	49.1586	84.12
	π/6	3π/4	2		14059	49.0569	72.60
	π/6	2π/3	1		17253	50.5724	66.37
	π/12	2π/3	2		23738	51.2369	53.73
	π/6	5π/6	2		14059	49.0569	72.60
	π/6	2π/3	4		13764	48.8781	73.17
#7	π/6	2π/3	2	65119	20061	67.7166	69.19
	π/3	2π/3	2		12438	66.7228	80.90
	π/6	3π/4	2		19407	66.6007	70.20
	π/6	2π/3	1		24306	66.6765	62.67
	π/12	2π/3	2		31256	68.0664	52.00
	π/6	5π/6	2		19407	66.6007	70.20
	π/6	2π/3	4		19519	66.3790	70.03
#8	π/6	2π/3	2	69338	21667	71.1693	68.75
	π/3	2π/3	2		12966	70.9684	81.30
	π/6	3π/4	2		21058	70.4685	69.63
	π/6	2π/3	1		27591	70.5840	60.21
	π/12	2π/3	2		33377	73.5876	51.86
	π/6	5π/6	2		21058	70.4685	69.63
	π/6	2π/3	4		20253	70.4120	70.79
#9	π/6	2π/3	2	80704	27723	83.4300	63.65
	π/3	2π/3	2		16673	80.9934	79.34
	π/6	3π/4	2		27691	80.4775	65.69
	π/6	2π/3	1		32261	80.5784	60.03
	π/12	2π/3	2		41666	85.3437	48.37
	π/6	5π/6	2		27617	80.2627	63.78
	π/6	2π/3	4		26225	80.1145	67.50
#10	π/6	2π/3	2	91554	29132	96.9351	68.18

	π/3	2π/3	2		15938	91.2527	82.59
	π/6	3π/4	2		29056	90.9111	68.26
	π/6	2π/3	1		32853	91.1335	64.12
	π/12	2π/3	2		47821	98.2886	47.77
	π/6	5π/6	2		29056	90.9111	68.26
	π/6	2π/3	4		28598	90.3195	68.76

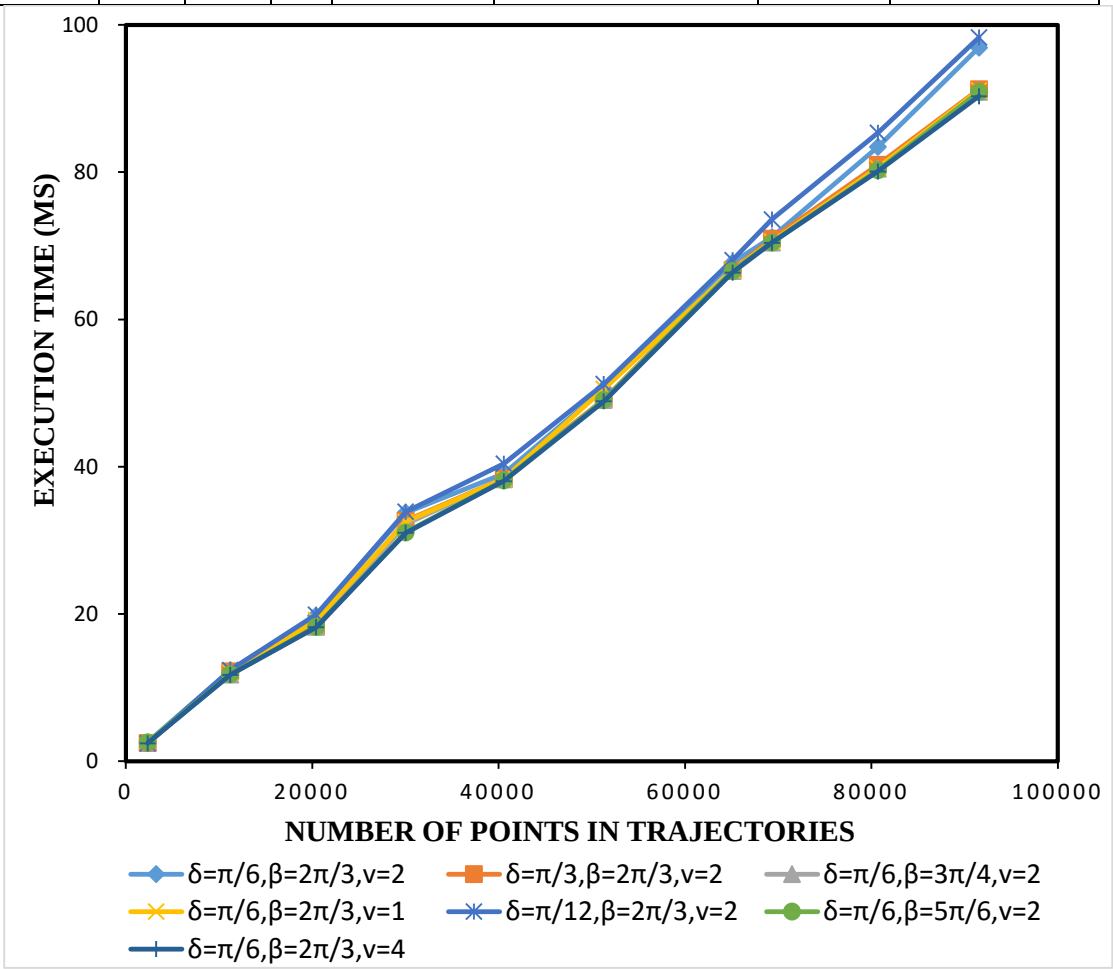


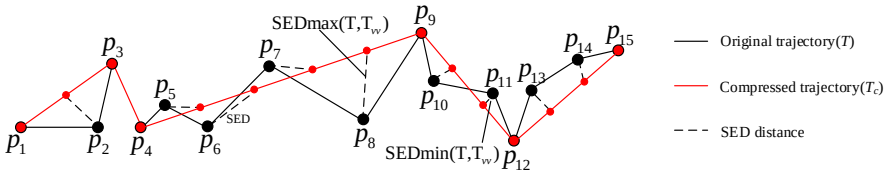
Figure 5. Compression time of SF algorithm with different parameters

This paper analyzes movement trends and internal characteristics of trajectories starting from the motion characteristics and trajectory structures, which takes a full consideration of moving objects' characteristic attributes, such as velocity value and velocity corner etc. Therefore, SF algorithm has a higher reliability and credibility, when it reserves the motion characteristics and local characteristic information of trajectories. Figure 5 shows the influence of trajectory size on compression time with different parameters. As it shows, the trajectory size will have an obvious influence on compression speed of SF algorithm, when its size is very small. The influence of trajectory size on compression speed will lower, as trajectory size increases; while, the influence of trajectory size on compression speed will enhance, when trajectory size reaches some order of magnitude. Hence, the computational complexity of SF algorithm is $O(n\log n)$, where n is the number of points in the trajectory. In addition, Figure 5 shows that the setting of velocity corner threshold has an obvious influence on compression speed and the setting of velocity value threshold doesn't have an obvious influence on compression speed. So, compression speed of SF

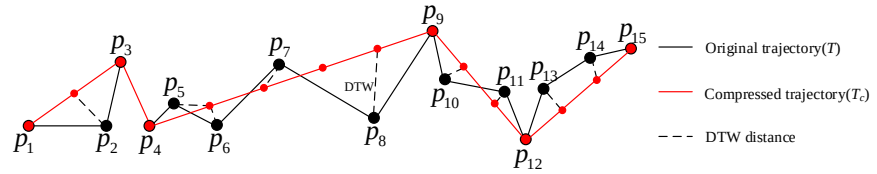
algorithm mainly relies on the size of trajectories and the setting of velocity corner threshold.

5.2 Performance analysis

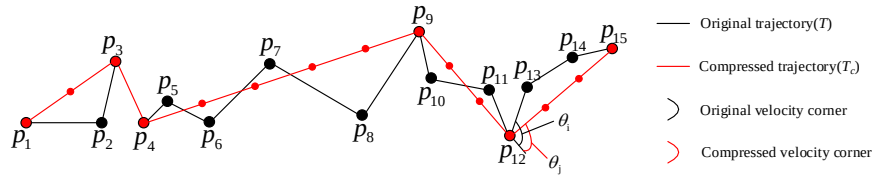
In order to verify the performance of SF as well as advantages and disadvantages between SF and existing algorithms, 3 methods are introduced in this paper to measure the information loss degree which are respectively denoted as SED comparison, DTW comparison and Corner comparison.



(a) A schematic of SED comparison



(b) A schematic of DTW comparison



(c) A schematic of Corner comparison

Figure 6. A schematic of information loss degree

1) SED comparison: SEDc(T, T_c) reflects the degree of position deviation between the compressed trajectory and the raw one, and the description of SED comparison is given in Figure 6(a) which can be calculated by formula (9), (10), (11) and (12).

$$\mathrm{SEDc}(T,T_c)=\frac{\mathrm{SEDmax}(T,T_c)+\mathrm{SEDavg}(T,T_c)+\mathrm{SEDmin}(T,T_c)}{3} \quad (9)$$

$$\mathrm{SEDmax}(T,T_c)=\max_{i=1}^{n}(\mathrm{SED}(op_i,cp_i)) \quad (10)$$

$$\mathrm{SEDavg}(T,T_c)=\frac{\sum_{i=1}^{n}\mathrm{SED}(op_i,cp_i)}{n} \quad (11)$$

$$\mathrm{SEDmin}(T,T_c)=\min_{i=1}^{n}(\mathrm{SED}(op_i,cp_i)) \quad (12)$$

Here, op_i and cp_i respectively are the i-th point of T and T_c whose length are both n. max() and min() respectively are the maximum value and minimum value among the SED distance between the compressed trajectory and the raw trajectory. SEDmax(T, T_c) is the maximum SED distance between the compressed trajectory and the raw trajectory. Similarly, SEDavg(T, T_c) and SEDmin(T, T_c) respectively are the average and minimum SED distance between the compressed trajectory and the raw trajectory. SED comparison calculates information loss degree by the mean of maximum, average and minimum SED distance.

2) DTW comparison: DTWc(T, T_c) reflects the degree of position deviation between the compressed trajectory and the raw trajectory. DTW distance is specifically defined as that in the case of ensuring the order of trajectory points, it completes the local scaling of time dimension by repeating the previous points, and makes the minimum distance between trajectories as DTW distance. The DTW distance between the compressed trajectory and the raw trajectory which can be calculated by formula (13) can be shown as Figure 6(b).

$$\mathrm{DTW}(T,T_c)=\begin{cases}0 & m=n=0\\ \infty & m=0||n=0\\ \mathrm{SED}(op_1,cp_1)+\min\begin{cases}\mathrm{DTW}(\mathrm{Rest}(T),\mathrm{Rest}(T_c))\\ \mathrm{DTW}(\mathrm{Rest}(T),T_c)\\ \mathrm{DTW}(T,\mathrm{Rest}(T_c))\end{cases} & others\end{cases} \quad (13)$$

Here, the length of T and T_c respectively are m and n. SED(op_1, cp_1) is the SED distance between two points op_1 and cp_1, which respectively are the first point of T and T_c. Rest(T) and Rest(T_c) are the remaining trajectory after removing the first sampling point. min is a function that calculates the minimum value among three parameters.

The DTWc(T, T_c) can be calculated by formula (14) according to formula (13).

$$\mathrm{DTWc}(T,T_c)=|\mathrm{DTW}(T,T_c)| \quad (14)$$

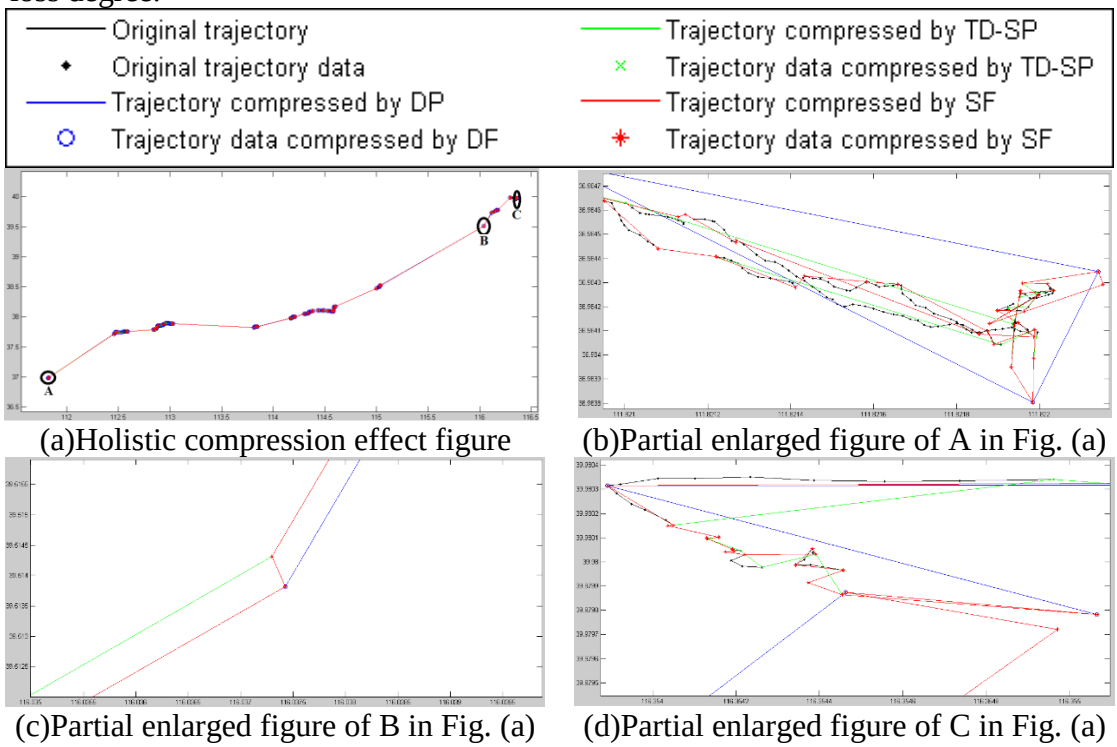
DTW comparison calculates the information loss degree by DTW distance which can measure the similarity between trajectories after the local scaling of time dimension by the scaling operation of time dimension.

3) Corner comparison: Cornerc(T, T_c) reflects the degree of motion direction deviation between the final moving object and the raw moving object shown as Figure 6(c). Corner comparison can be calculated by formula (15).

$$\mathrm{Cornerc}(T,T_c)=\frac{\sum_1^{\min(m,n)}((|\theta_i-\theta_j|)/(|\theta_i|+|\theta_j|))}{m+n} \quad (15)$$

Here, the length of T and T_c respectively are m and n. Corner comparison calculates the information loss degree by the speed corner of moving objects.

To verify the performance of SF algorithm, it is compared with DP and TD-SB in this paper by compressing 5 trajectories. This section compares the algorithms across multiple performance metrics including matching effect, compression speed and compression ratio as well as information loss degree.



(a)Holistic compression effect figure (b)Partial enlarged figure of A in Fig. (a)

(c)Partial enlarged figure of B in Fig. (a) (d)Partial enlarged figure of C in Fig. (a)

Figure 7. Effect figure of DP (ε=0.0005), TD-SB (v=2) and SF (β=π/3, δ=2π/3 and

v=4) compressing trajectory 1

As shown in Figure 7, SF not only keeps the holistic external shape of origin trajectory (Fig. 7(a)), but also keeps the local motion characteristics of moving objects in detail (Fig. 7(b), Fig. 7(c) and Fig. 7(d)). In Table 2, compared with DP and TD-SB, SF has a faster compression speed, but a lower compression ratio than DP. There are two main reasons as follow: (1) SF algorithm takes full consideration of the motion characteristics which is beneficial to the reservation of holistic motion characteristics. (2) SF algorithm emphatically analyzes the characteristic information contained in trajectories which is beneficial to keep the local motion characteristics of moving objects.

Table 2 Performance comparison between different algorithms

Trajectory ID	Algorithm name	Correlation parameters	Trajectory size	Compression time (ms)	Compression ratio (%)
#1	DP	ε=0.0005	2359	7.4857	95.13
	TD-SB	v=2		19.1639	89.87
	SF	β=π/3, δ=2π/3, v=4		3.2859	90.04
#2	DP	ε=0.0001	11225	37.3477	89.94
	TD-SB	v=2.5		144.4079	90.32
	SF	β=2π/3, δ=5π/6, v=5		17.8605	87.55
#3	DP	ε=0.0001	4201	13.4007	90.43
	TD-SB	v=3.5		31.1312	90.26
	SF	β=2π/3, δ=5π/6, v=6		10.0121	90.53
#4	DP	ε=0.0001	545	1.5155	82.75
	TD-SB	v=2.5		3.0669	84.4
	SF	β=2π/3, δ=5π/6, v=6		0.8073	83.85
#5	DP	ε=0.0001	212	0.3874	88.68
	TD-SB	v=2.5		0.9613	88.68
	SF	β=π/3, δ=5π/6, v=6		0.2331	90.1

Table 3 Comparison of information loss degree

Trajectory ID	Algorithm name	Correlation parameters	Compression ratio (%)	Information Loss		
				SED comparison	DTW comparison	Corner comparison
#1	DP	ε=0.0005	95.13	0.003716	0.561906	0.344291
	TD-SB	v=2	89.87	0.004675	2.558741	0.340755
	SF	β=π/3, δ=2π/3, v=4	90.04	0.004438	2.035024	0.340490
#2	DP	ε=0.0001	89.94	0.013970	1.409007	0.325382
	TD-SB	v=2.5	90.32	0.003484	4.241242	0.339533

	SF	$\beta=2\pi/3$, $\delta=5\pi/6$, $v=5$	87.55	0.009492	6.090871	0.324233
#3	DP	$\varepsilon=0.0001$	90.43	0.001881	0.187654	0.323714
	TD-SB	$v=3.5$	90.26	0.002610	1.670170	0.342875
	SF	$\beta=2\pi/3$, $\delta=5\pi/6$, $v=6$	90.53	0.001898	0.455062	0.301833
#4	DP	$\varepsilon=0.0001$	82.75	0.000343	0.026520	0.288413
	TD-SB	$v=2.5$	84.4	0.001005	0.233393	0.300421
	SF	$\beta=2\pi/3$, $\delta=5\pi/6$, $v=6$	83.85	0.000957	0.124727	0.281325
#5	DP	$\varepsilon=0.0001$	88.68	0.000841	0.047030	0.301000
	TD-SB	$v=2.5$	88.68	0.001423	0.163142	0.318713
	SF	$\beta=\pi/3$, $\delta=5\pi/6$, $v=6$	90.1	0.000938	0.024417	0.273268

To describe the experimental results more intuitionistic, this paper converts matching effect into numeric by the calculation methods of information loss degree given in this section. Information loss degrees caused by DP, TD-SB and SF with same or similar compression ratio are recorded in Table 3 which includes SED comparison and DTW comparison which reflect holistic matching effect of trajectories, as well as Corner comparison which reflects internal matching effect in trajectories. Generally, the smaller information loss degree is, the better matching effect of an algorithm is, namely, the higher reliability of an algorithm is. As shown in Table 4, when the compression ratio of DP, TD-SB and SF is same or similar, the Corner comparison of SF is smallest which indicates that SF has a better effect in keeping internal characteristics and motion characteristics of moving objects. Most of the SED comparison of SF is lower than TD-SB and higher than DP, so SF is superior to TD-SB and slightly inferior to DP while keeping the holistic characteristic of trajectories. Most of the DTW comparison of SF is between DP and TD-SB in Table 3, which indicates SF is superior to TD-SB and slightly inferior to DP in keeping the holistic characteristic of trajectories. Comprehensive analyzing the experimental results above, SF can not only keep the motion characteristics and internal characteristic information of trajectories, but also keep the holistic characteristics of trajectories. As a consequence, SF is more suitable for the compression of moving objects whose motion characteristics are required to be kept in detail.

6. Conclusions

This paper, which starts from motion characteristics of moving objects, introduces moving objects' velocity corner and velocity value at sampling points, as well as takes a full consideration of motion characteristics and characteristic information contained in trajectories. First, this paper proposes SF that determines retained points by velocity corner of moving objects to compress trajectory data. After that, SF smooths the compressed trajectory according to velocity value of moving objects, and finally finishes the compression. The experimental results show that: the algorithm proposed in this paper not only has high efficiency, but also can preferably keep local motion characteristics of moving objects. Thus, SF is a highly efficient trajectory data compression algorithm whose compression results are more significance in practice and very suitable for the

compression of moving objects whose motion characteristics are required to be kept in detail.

Acknowledgement This work was supported by the Fundamental Research Funds for the Central Universities, China (with grant of 2015XKMS085).

Disclosure statement The authors declare that there is no conflict of interests regarding the publication of this manuscript.

References

[1] Minjie Chen, Mantao Xu, Pasi Franti. Compression of GPS trajectories. In: Proceedings of the 2012 Data Compression Conference (DCC 2012), April 10-12, 2012, Snowbird, Utah, USA, 62-71.
[2] Nirvana Meratnia, Rolf A. de By. Spatiotemporal compression techniques for moving point objects. In: Proceedings of the 9th International Conference on Extending Database Technology (EDBT 2004), March 14-18, 2004, Crete, Greece, 765-782.
[3] Xiaoying Liu. Trajectory data compression via spatial-temporal properties. Master Degree Thesis, Hong Kong University of Science and Technology, 2014, Hong Kong, China.
[4] Douglas H David, Peucker K Thomas. 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, 1973, 10(2): 112-122.
[5] Michalis Potamias, Kostas Patroumpas, Timos Sellis. Sampling trajectory streams with spatiotemporal criteria. In: Proceedings of the 18th International Conference on Scientific and Statistical Database Management (SSDBM 2006). July 3-5, 2006, Vienna, Austria, 275-284.
[6] Jianjun Liu, Kun Zhao, Philipp Sommer, et al. Bounded quadrant system: error-bounded trajectory compression on the go. In: Proceedings of the 31th IEEE International Conference on Data Engineering (ICDE 2015), March 31-April 4, 2015, Seoul, Korea, 987-998.
[7] Birnbaum J, Meng H C, Hwang J H, et al. Similarity-based compression of GPS trajectory data. In: Proceedings of the 4nd International Conference on Computing for Geospatial Research & Applications (COM. Geo 2013), July 22-24, 2013, San Jose, CA, USA, 92-95.
[8] Cheng Long, Raymond Chi-Wing Wong, H. V. Jagadish. Direction-preserving trajectory simplification. Proceedings of the VLDB Endowment, 2013, 6(10): 949-960.
[9] Jae-Gil Lee, Jiawei Han, Xiaolei Li. Trajectory outlier detection: A partition-and-detect framework. In: Proceedings of the 24th IEEE International Conference on Data Engineering (ICDE 2008). April 7-12, 2008, Cancún, México, 140-149.
[10] Jae-Gil Lee, Jiawei Han, Kyu-Young Whang. Trajectory clustering: a partition-and-group framework. In: Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data (SIGMOD 2007). June 11-14, 2007, Beijing, China, 593-604.
[11] Guan Yuan, Shixiong Xia, Lei Zhang, Yong Zhou and Cheng Ji. An efficient trajectory-clustering algorithm based on an index tree. Transaction of the Institute of Measurement and Control 2012, 34(7):850-861.
[12] Aiden Nibali, Zhen He. Trajic: An effective compression system for trajectory data. IEEE Transaction on Knowledge and Data Engineering, 2015, 27(11):3138-3151.
[13] Rajib Rana, Mingrui Yang, Tim Wark, et al. SimpleTrack: Adaptive trajectory compression with deterministic projection matrix for mobile sensor networks. IEEE Sensors Journal, 2015, 15(1): 365-373.
[14] Kuien Liu, Yaguang Li, Jian Dai, Shuo Shang, Kai Zheng. Compressing large scale urban

trajectory data. In: Proceedings of the Fourth International Workshop on Cloud Data and Platforms (CloudDP 2014). April 13, 2014, Amsterdam, Netherlands, 3: 1-6.
[15] Renchu Song, Weiwei Sun, Baihua Zheng, Yu Zheng. PRESS: A novel framework of trajectory compression in road networks. Proceedings of the VLDB Endowment, 2014, 7(9): 661-672.
[16] Georgios Kellaris, Nikos Pelekis, Yannis Theodoridis. Map-matched trajectory compression. Journal of Systems and Software, 2013, 86(6): 1566-1579.
[17] Iulian Sandu Popa, Karine Zeitouni, Vincent Oria, et al. Spatio-temporal compression of trajectories in road networks. GeoInformatica, 2014, 19(1): 117-145.
[18] Falko Schmid, Kai-Florian Richter, Patrick Laube. Semantic trajectory compression. In: Proceedings of the 11the International Symposium on Spatial and Temporal Databases (SSTD 2009). July 8-10, 2009, Aalborg, Denmark, 411-416.
[19] Richter K F, Schmid F, Laube P. Semantic trajectory compression: Representing urban movement in a nutshell. Journal of Spatial Information Science, 2015 (4): 3-30.
[20] Shenzhu Feng, Jian Xu Ming, Xu Ning Zheng, et al. EHSTC: an enhanced method for semantic trajectory compression. In: Proceedings of the 4th ACM SIGSPATIAL International Workshop on GeoStreaming (IWGS 2013). November 5, 2013, Orlando, Florida, USA, 43-49.
[21] Jonathan Muckell, Jeong-Hyon Hwang, Vikram Patil, et al. SQUISH: an online approach for GPS trajectory compression. In: Proceedings of the 2nd International Conference on Computing for Geospatial Research & Applications (COM. Geo 2011). Article number: 13, 1-8.
[22] Jonathan Muckell, Paul W, Olsen Jr., Jeong-Hyon Hwang, et al. Compression of trajectory data: a comprehensive evaluation and new approach. GeoInformatica, 2014, 18(3): 435-460.
[23] Guangwen Liu, Masayuki Iwai, Kaoru Sezaki. A method for online trajectory simplification by enclosed area metric. In: Proceedings of the 6th International Conference on Mobile Computing and Ubiquitous Networking, May 23 - 24, 2012, Okinawa Japan, 40-47.
[24] Guangwen Liu, Masayuki Iwai, Kaoru Sezaki. An online method for trajectory simplification under uncertainty of gps. IPSJ Transactions on Databases, 2013, 6(3): 40-49.
[25] Wei Pan, Chunlong Yao, Xu Li, Lan Shen. An online compression algorithm for positioning data acquisition. Informatica, 2014, 38(4):339-346.
[26] Trajcevski G, Cao H, Scheuermanny P, et al. On-line data reduction and the quality of history in moving objects databases[C]//Proceedings of the 5th ACM international workshop on Data engineering for wireless and mobile access. ACM, 2006: 19-26.