**GPS Trajectory Compression and Recovery based on Compressive sensing**

Guan Yuan, Jie Zhao, Shixiong Xia, Penghui Sun

a.School of Computer Science and Technology, China University of Mining and Technology,

b.Jiangsu Key Laboratory of Mine Mechanical and Electrical Equipment, China University of Mining and Technology,

No. 1 Daxue Road, Xuzhou, Jiangsu, China, 221116

Corresponding to: Guan Yuan ([yuanguan@cumt.edu.cn](mailto:yuanguan@cumt.edu.cn))

**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 original trajectory is compressed based on moving direction of the object. Then, the original trajectory is simplified according to internal fluctuation in original trajectory. Finally, the original trajectory is compressed by trajectory velocity. 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. 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 to store a single day trajectory data. These data are the foundation for us to analyze activities and patterns for moving objects. However, the enormous volume of data has brought several problems [1]. First, it is quite expensive and time-consuming to transmit these large amounts of data. Second, it is computationally expensive operations to query and extract useful patterns from these large amounts of trajectory data. Third, GPS trajectories are often with much redundant and trivial data that waste storage and cause increased disk I/O time. These issues can be addressed by reducing the size of trajectory data. Therefore, the aim of data compression technique is to decrease the occupied memory space and improves the transmission, storage and processing by reducing data volume without obviously losing information, or by reorganizing data with certain strategies to reduce the redundancy and memory cost. For moving objects trajectories, it is essential to preserve as much features, including position, direction, corner and velocity as possible while reducing redundant 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*(*n2*). 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*(*n2*). 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*(*n2*). 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 *p0* as the key point, and stores *p0* and its velocity in the compressed representation. Then the next point *pi* is estimated whether it’s location within the SED threshold from *p0*. If true then continue the next point of *pi*, else *pi* 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 original ones well. However, if the movement pattern and internal features are neglect, applications, such as trajectory clustering [7, 8], outlier detection [9] and activity discovery [10] may be not so accuracy as we expected. Therefore, in this paper, 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.

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. Therefore, in this paper, we take the corner and velocity as our main goal to conquer in trajectory compression.

In order to solve the problem mentioned above, a three-phase compression algorithm is proposed in this paper, which is called as Trajectory Simplification Algorithm based on Structure Features (SF). Firstly, SF algorithm compresses the original trajectory based on moving direction of the object; then, it simplifies the original trajectory according to the internal fluctuations in trajectory; finally, it compresses the original trajectory by the trajectory velocity.

Therefore, we should emphasize trajectory compression should satisfy the following goals [1, 2]. Firstly, compression method should identify which parts are redundant, and which are noise or outliers. Secondly, compression method should make sure how to organize the compressed data after filtering noise and removing redundant data. Finally, compression method should make sure as much as important information be obtained.

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.

The velocity corner of moving objects is introduced to compress trajectory data, through which, the movement characteristics and the internal characteristic information in trajectories will be kept in detail.

2) Then, the algorithm proposed in this paper simplifies trajectories according to internal fluctuation 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 VTC, we carry out a comprehensive comparison with other algorithms such as DP and TD-SP.

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 (DP) 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 DP algorithm, which only considers spatial information but neglect temporal information in trajectory data. In order to overcome this shortcoming, Meratnia et al.(ref.??) 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 characteristics of moving objects and internal characteristic information in trajectories, such as the influence of velocity and acceleration on trajectories, as well as the influence of sampling errors on the trajectory calculation and so on. However, in reality, it is essential for trajectory compression to reserve moving objects’ movement characteristics and internal 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 while keeping the holistic feature of moving objects’ trajectories.

To tackle the above disadvantages, this paper proposes a new compression algorithm based on the structure features of trajectories (SF algorithm). It firstly compresses trajectories based on the moving direction of moving objects. Then, it simplifies trajectories according to the internal fluctuation in trajectories. Finally, it compresses trajectories by trajectory velocity. At the end of this paper, we analyze the influence of various parameters on the algorithm proposed in this paper and compared the algorithm proposed in this paper 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 the algorithm proposed in this paper 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-basedalgorithms are mentioned in theMaster Degree Thesis of Xiaoying Liu [12], such as Huffman Coding, ZIP, LMZA2, can compress trajectories without any loss and preserve detailed trajectory information. However, the 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 two raw trajectories *T1* and *T2* as shown in Figure 1 (a)(i) and (b)(i) respectively. Each trajectory has 10 points (*p0*, *p1*, …, *p9*), and all points are sampled with fixed interval. *T1* is a straight trajectory, and if an existing PPTS algorithm with SED as error bound is used to compress *T1*, then *T1’* can be obtained as a simplified version of *T1* shown in Figure 1 (a)(ii). However, as we can see that the velocity of *p0* to *p3* and *p7* to *p9* is quite different from that of *p3* to *p7*. In a city transport system, the part *p3* to *p7* may be more important than others in data analysis. Therefore, we should present a new technique to preserve this part as *T1”* shown in Figure 1 (a)(iii). *T2* is a circular trajectory with two smooth parts *p0* to *p3*, *p6* to *p9* and a fluctuant part *p3* to *p6*. We can get a simplified trajectory as *T2’* shown in Figure 1 (b)(ii) with existing PPTS algorithms. However, the fluctuant part is lost in *T2’*, and this part may be crucial in moving object activity discovery [10]. Therefore, this part should be preserved in compression process as *T2”* in Figure 1 (b)(iii) for deep analysis.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| (a)(i)*T1* | (a)(ii)*T1’* | (a)(iii)*T1’’* |
|  |  |  |
| (b)(i)*T2* | (b)(ii)*T2’* | (b)(iii)*T2’’* |
| 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.

|  |  |
| --- | --- |
|  |  |
| (a) Raw points of trajectory 1# | (b) Construction of trajectory 1# |
|  |  |
| (c) Direction changes in trajectory 1# | (d) Speed changes in trajectory 1# |
|  |  |
| (e) Acceleration changes in trajectory 1# | (f) Compressed version of trajectory 1# |
| Figure 2. A motivating example | |

**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 [11] is also available in this paper. *TD* (*T*rajectory *D*atabase) denotes trajectory set *TD*={*TR1*, *TR2*, …, *TRn*}, and *TRi* is the *i*th trajectory. A *TR*ajectory trajectory is a chronological sequence consisted of multi-dimensional locations, which is denoted by *TRi= {P1*, *P2*, *…*, *Pm}*(*1≤i≤n*). *Pj*(*1≤j≤m)* a sampling point in *TRi*, is represented as *<Locationj*, *Tj>*, which means that the position of the moving object is *Locationj* at time *Tj*. *Locationj* is a multi-dimensional location point, for instance, (*xj, yj*) 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 convenient, we introduce symbols *VA* as the arriving speed and *VL* as the leaving speed at each point. The definitions on trajectory corner and velocity as follows.

**Definition 1.** Trajectory Corner [11]: 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 two adjacent segments is denoted as *α*, and the turn angle in the moving direction marked with *θi* and *θn-1* are two trajectory corners. In order to simplify the calculation, the clockwise corner (*θi*) is marked as negative and the anticlockwise corner (*θn-1*) is marked as positive.

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

(1)

Here, |·| is the length of trajectory segment. Trajectory segment vectors and are two directed adjacent edges with angle *α* and can be denoted as and . Segment vector is the virtual opposite edge of angle *α*, denoted as .The trajectory corner *θi* at *pi* can be calculated by formula (1) and (2):

(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: the moving tendency of an object. For a given trajectory, moving direction is represented by the accumulation of turn angles at each sampling points.

We use *Dθ* to denote the movement pattern of a trajectory. According to Definition 2, *Dθ* 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 *Dθ* sequentially. Once *Dθ* 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 *Dθ* should be zero clearing and newly accumulated from the next point. The formula of *Dθ* is given as: ***Dθ*** =**∑*θi***.

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

(3)

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

(4)

The arriving speed (*VA*) and leaving speed (*VL*) at *pi* are shown in Figure 2. Note that, the arriving speed at *p1* and the leaving speed at *pn* are 0, for there are no succeed point of *pn* and no precursor point of *p1*.



Figure 2. An example of trajectory corner and velocity

With the combination of *VA* and *VL*, we can easily analysis the motion characteristics of moving objects in a certain area at certain time. For example, if at some points, *VA* is smaller than *VL*, 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 5.** Trajectory Instantaneous Velocity: the speed value at each sampling point, which is calculated in the form of deviation between two adjacent trajectory segments. Trajectory instantaneous velocity reflects the motion characteristics at the sampling point *p*.

We use *Vi* as the speed deviation at sampling point *pi* (1≤*i*≤*n*), and *Vi* = abs(*VAi- VLi*). 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 *p0*, *p3*, *p7* and *p9*, speed values change greater than those of others, so these points should be preserved in a(iii) according to the idea of our algorithm.

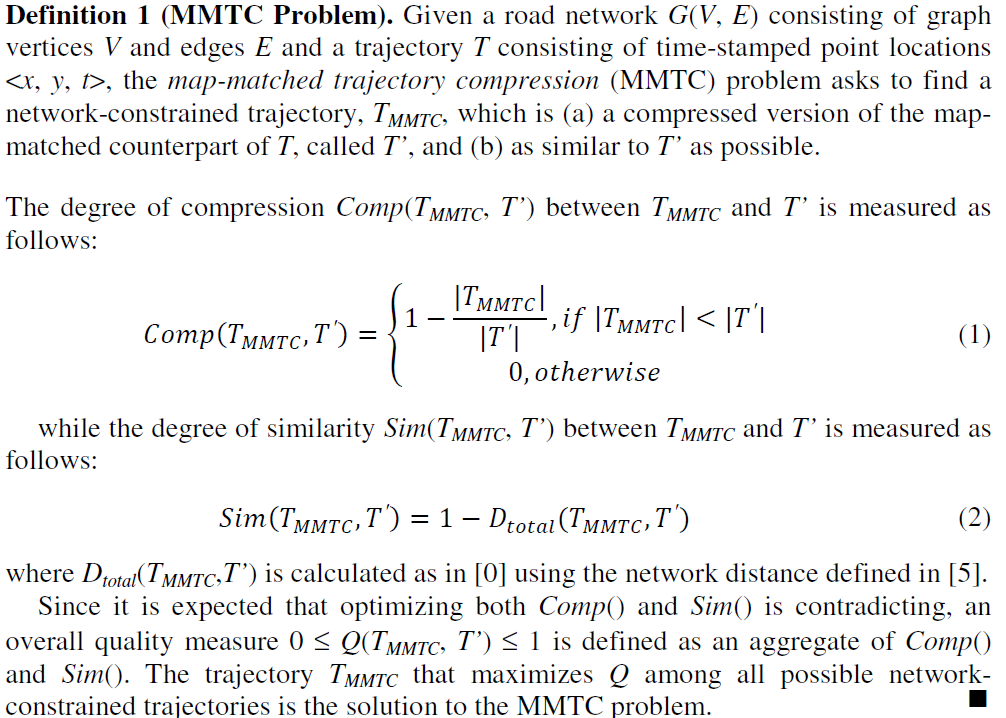
**Definition 6.** Internal Fluctuation: For a given trajectory *T*, the internal fluctuation is that there exists several continuous points, at which trajectory corners change sharply, denoted by *Fε,k*(*ε* is the threshold of corner, and *k* is the threshold of continuous points). *Fε,k* means that there are at least *k* continuous points, where trajectory corners greater than *ε*.

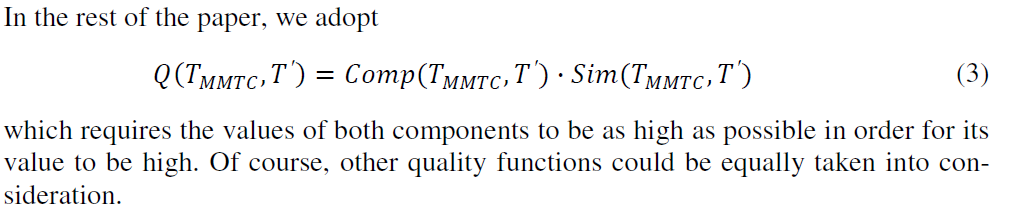
In *T2* of Figure 1(b), corners at sampling points from *p3* to *p6* change greater than others. Therefore, we can infer that there are something happened at trajectory segment *p3* to *p6*. Traditional simplification algorithms may neglect this part and remove these points making internal features lost. In order to avoid noise distortion, *k* is often set greater than 3.

The greater turn at a sampling point, the more semantic meaning it is.

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.

**4. Error Matrix**





**5. 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, 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.

最小描述长度准则（Minimum Description Length Principle， MDL）是一种信号压缩准则，没有阈值的设定与选择，综合考虑了压缩效果和重构精度两方面的要求，因而是一种数据压缩的准则方法。MDL定义如下：

一个关键的问题是，如何尽可能的压缩数据，同时又具有较好的信息保留度。这两方面是一个互斥（对应）的指标。很难找到这个指标都达到最优的方案。因此，需要寻找一种折中的思路，让压缩后的轨迹既具有较好的压缩效果，同时又具有良好的信息保留度。

Hereafter, we propose our methodology for finding the optimal tradeoff between compression and similarity, in other words, a method that would find the path on the road network that maximize both compression and similarity. In particular, we adopt the Minimum Description Length (MDL) principle. There are two components that comprise MDL, namely L(H) and L(D|H), where H denotes the hypothesis and D denotes the data. According to **Lee[27] and Hansen[28]**, “L(H) is the length, in bits, of the description of the hypothesis; and L(D|H) is the length, in bits, of the description of the data when encoded with the help of the hypothesis”. The best hypothesis H to explain D is the one that minimizes the sum of L(H) and L(D|H). Mapping the above discussion to our problem, a hypothesis corresponds to a specific compression achieved by a compressed trajectory on the network and the data are the nodes of the map-matched counterpart of the original trajectory. Therefore, finding the best compressed trajectory translates to finding the best hypothesis using the MDL principle. Thus, L(H) represents the compression achieved by a compressed trajectory and L(D|H) represents the difference between the compressed trajectory and the original one.

Lee的方法智能找到近似的轨迹划分，无法找到比较精确的轨迹划分。如：比较恰当的拟合，比较好的信息保留度、比较好的结构信息保留等。

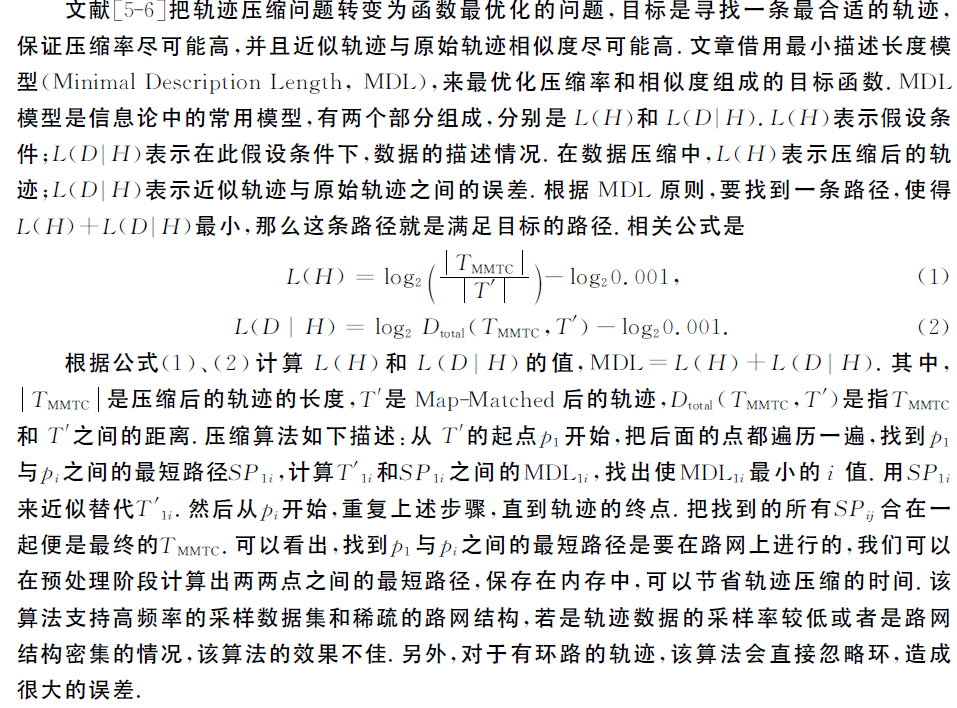
Lee的方法比较简单，压缩效率比较高，在精度preciseness和conciseness具有比较好的压缩效果和精度，不需要参数进行交互。我们对其进行了改进，通过引入参数，使得轨迹压缩变得可调、可控。通过参数提高压缩的精度。

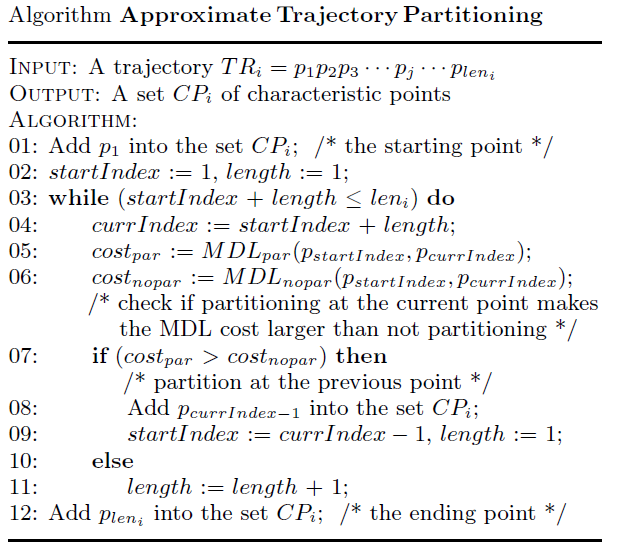
Lee [10] proposed a partition-and-group framework to cluster trajectories. In his work, trajectories are partitioned into many segments at characteristic points detected by MDL principle. Then trajectories are The is the path that minimizes the sum L(H) + L(D|H) adopting the MDL principle.

The Approximate Trajectory Partitioning algorithm is efficient for compressing trajectories and doesn’t need parameters interaction. However, the algorithm fails to find the optimal partitioning, and a great many of structure features are lost in the partitioned trajectories.

Kellaris[29]等在路网环境下利用Map-match方法和轨迹压缩方法组合，给出了路网环境下的轨迹数据压缩方法。

We formulate L(H) as the binary logarithm of the ratio of the number of points |TMMTC| of the resulted trajectory over the number of points |T| of the map-matched counterpart of the original trajectory, i.e.:





There two stages in VTC algorithm shown in Figure 3. Firstly, trajectory points are filtered based on trajectory velocity, and in this stage, trajectory points, at which their speed changed little will be removed according to the given velocity threshold. Then, compressed trajectories are smoothed based on trajectory corner that removes the trajectory points whose velocity value are less than the given velocity value threshold.

|  |
| --- |
|  |
| (a) |
|  |
| (b) |
|  |
| (c) |
| Figure 3 A compression schematic of VTC algorithm. (a) An origin trajectory of moving objects denoted as *T*. (b) A new trajectory denoted as *Tc* obtained by compressing *T* based on VC algorithm. (c) A new trajectory denoted as *Tv* acquired by smoothing *Tc* based on VV algorithm.  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* | Origin trajectory. | | *Tc* | Trajectory compressed by SF algorithm. | | |•| | The absolute value of •. | | *k* | The counter of internal fluctuation. | | *pi* | The i-th point in a trajectory, *pi=<xi, yi, ti>.* | | *md*, *β* | Moving direction, moving direction threshold. | | *tc*, *δ* | Trajectory corner, internal fluctuation threshold. | | *tv*, *v* | Trajectory velocity, 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 three-phase algorithm, firstly it reserves the points whose moving direction is greater than moving direction threshold (*β*) (Lines 01-08); then, it reserves the internal fluctuations in original trajectory, while meeting the condition of internal fluctuation (Lines 09-17); finally, it reserves the points whose trajectory velocity is greater than trajectory velocity threshold (*v*) (Lines 18-22). This algorithm will be end until all of the points in original trajectory have been processed.

|  |
| --- |
| **Algorithm:** Trajectory Simplification Algorithm based on Structure Features (SF) |
| Input: Origin trajectory (*T*), moving direction threshold (*β*), internal fluctuation threshold (*δ*), trajectory velocity threshold (*v*)  Output: Compressed trajectory (*Tc*)  01) *md* ← *0*; // Set the value of *md* is *0*  02) *k* ← *0*; // Set the counter of internal fluctuation *k* is *0*  03) for each *pi* ∈ *T* do  /\* Compress original trajectory based on moving direction \*/  04) *tc* ← *θi* ; // Calculate trajectory corner *tc* of each trajectory point *pi*  05) *md* ← *md + tc*; // Calculate moving direction *md*  06) if |*md|* > *β* then // Save coordinate information of *pi* in *Tc*, when the absolute value of moving direction *md* of *pi* is greater than moving direction threshold *β*  07) *Tc* ← the coordinate information of *pi*;  08) *md* ← *0*; // Set the value of *md* is *0*  /\* Compress original trajectory based on internal fluctuation \*/  09) if *|tc|* > *δ* then // The value of *k* grows*1* automatically, when the absolute value of trajectory corner *tc* of *pi* is greater than trajectory corner threshold *δ*  10) *k++*;  11) else  12) if *k > 2* then  13) for *j ← 0* to *k+2* do  14) *Tc* ← the coordinate information of *p*i*-j* ; //Save coordinate information of *pi-j* in *Tc*, when the value of *k* is greater than *2*  15) end for  16) *k* ← *0*; // Set the counter of internal fluctuation *k* is *0*  17) *md* ← *0*; // Set the value of *md* is *0*  /\* Compress original trajectory based on trajectory velocity \*/  19) *tv* ← *Vi* ; // Calculate trajectory velocity *tv* of each trajectory point *pi*  20) if *tv* > *v* then // Save coordinate information of *pi* in *Tc*, when the value of trajectory velocity *tv* of *pi* is greater than trajectory velocity threshold *v*  21) *Tc* ← the coordinate information of *pi*;  22) 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*log*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 algorithms given in this paper, only 2 parameters are required, corner threshold (*β*) and velocity threshold (*v*). The guidence of *βrms* and *vrms* can be calculated by formula (7) and (8), for the root mean square (RMS) is a kind of numerical indicators measuring accuracy of measurement. The guideline value of *βrms* and *vrms* 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.

(7)

(8)

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

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

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Trajectory ID | *β* | *v* | Points of origin trajectory | Points of compressed trajectory | Time cost (ms) | Compression ratio (%) |
| #1 | 0.1496 | 2 | 2359 | 168 | 4.8947 | 92.88 |
| 0.1745 | 2 | 152 | 4.395 | 93.56 |
| 0.2094 | 2 | 132 | 3.6182 | 94.40 |
| 0.1745 | 1 | 220 | 4.4922 | 90.67 |
| 0.1745 | 4 | 115 | 4.3886 | 95.13 |
| #2 | 0.1496 | 2 | 11225 | 1878 | 31.9411 | 83.27 |
| 0.1745 | 2 | 1715 | 25.4143 | 84.72 |
| 0.2094 | 2 | 1538 | 23.4502 | 86.30 |
| 0.1745 | 1 | 3281 | 26.6946 | 70.77 |
| 0.1745 | 4 | 1012 | 26.0624 | 90.98 |
| #3 | 0.1496 | 2 | 20429 | 1344 | 57.0758 | 93.42 |
| 0.1745 | 2 | 1202 | 46.5066 | 94.12 |
| 0.2094 | 2 | 1538 | 43.9987 | 92.47 |
| 0.1745 | 1 | 3474 | 48.2638 | 82.99 |
| 0.1745 | 4 | 463 | 45.1552 | 97.73 |
| #4 | 0.1496 | 2 | 30045 | 6122 | 93.8325 | 79.62 |
| 0.1745 | 2 | 1657 | 71.5765 | 94.48 |
| 0.2094 | 2 | 1494 | 67.9026 | 95.03 |
| 0.1745 | 1 | 4986 | 77.4775 | 83.40 |
| 0.1745 | 4 | 681 | 76.6432 | 97.73 |
| #5 | 0.1496 | 2 | 40570 | 1988 | 127.2203 | 95.10 |
| 0.1745 | 2 | 1813 | 92.1038 | 95.53 |
| 0.2094 | 2 | 1586 | 84.4227 | 96.09 |
| 0.1745 | 1 | 6346 | 99.2556 | 84.36 |
| 0.1745 | 4 | 608 | 96.8964 | 98.50 |
| #6 | 0.1496 | 2 | 51303 | 3349 | 153.616 | 93.47 |
| 0.1745 | 2 | 3044 | 128.555 | 94.07 |
| 0.2094 | 2 | 2563 | 114.7754 | 95.00 |
| 0.1745 | 1 | 7768 | 136.7463 | 84.86 |
| 0.1745 | 4 | 1482 | 137.5139 | 97.11 |
| #7 | 0.1496 | 2 | 65119 | 6120 | 187.7347 | 90.60 |
| 0.1745 | 2 | 5565 | 158.5361 | 91.45 |
| 0.2094 | 2 | 4842 | 146.0848 | 92.56 |
| 0.1745 | 1 | 14855 | 175.3391 | 77.19 |
| 0.1745 | 4 | 2972 | 177.6402 | 95.44 |
| #8 | 0.1496 | 2 | 69338 | 8199 | 200.499 | 88.18 |
| 0.1745 | 2 | 7228 | 173.8603 | 89.58 |
| 0.2094 | 2 | 6208 | 158.5186 | 91.05 |
| 0.1745 | 1 | 15166 | 185.0874 | 78.13 |
| 0.1745 | 4 | 4122 | 183.9311 | 94.06 |
| #9 | 0.1496 | 2 | 80704 | 7295 | 247.5391 | 90.96 |
| 0.1745 | 2 | 6559 | 206.4071 | 91.87 |
| 0.2094 | 2 | 5766 | 196.9282 | 92.86 |
| 0.1745 | 1 | 17349 | 221.1404 | 78.50 |
| 0.1745 | 4 | 2764 | 222.4883 | 96.58 |
| #10 | 0.1496 | 2 | 91554 | 4203 | 299.3554 | 95.41 |
| 0.1745 | 2 | 3842 | 249.094 | 95.80 |
| 0.2094 | 2 | 3308 | 243.6899 | 96.39 |
| 0.1745 | 1 | 11959 | 260.0325 | 86.94 |
| 0.1745 | 4 | 1862 | 259.6663 | 97.97 |

Figure 4 Compression time of VTC 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, VTC algorithm has a higher reliability and credibility, when it reserves the motion characteristics and local characteristic information of trajectories. Figure 4 shows the influence of trajectory size on compression time with different parameters. As shown in Figure 4, the trajectory size will have an obvious influence on compression speed of VTC 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 VTC algorithm is *O*(*n*log*n*), where n is the number of points in the trajectory. In addition, Figure 4 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 VTC 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 VTC as well as advantages and disadvantages between VTC 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 4. A schematic of information loss degree |

1) SED comparison: SED(*T*, *Tv*) reflects the degree of position deviation between the compressed trajectory and the origin one, and the description of SED comparison is given in Figure 5(a) which can be calculated by formula (7), (8), (9) and (10).

(7)

(8)

(9)

(10)

Here, *opi* and *rpi* respectively are the i-th point of *OTR* and *FTR* whose length are both *n*. max() and min() respectively are the maximum value or minimum value among the SED distance between the final trajectory and the origin trajectory. SEDmax(*T*, *Tv*) is the maximum SED distance between the final trajectory and the origin trajectory. Similarly, SEDavg(*T*, *Tv*) and SEDmin(*T*, *Tv*) respectively are the average and minimum SED distance between the final trajectory and the origin trajectory. SED calculates information loss degree by the mean of maximum, average and minimum SED distance.

2) DTW comparison: DTW(*T*, *Tv*) reflects the degree of position deviation between the final trajectory and the origin 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 final trajectory and the origin trajectory which can be calculated by formula (11) can be shown as Figure 5(b).

(11)

Here, the length of *T* and *Tv* respectively are *m* and *n*. SED(*op1*, *rp1*) is the SED distance between two points *op1* and *rp1*, which respectively are the first point of *OTR* and *FTR*. Rest(*T*) and Rest(*Tv*) are the remaining trajectory after removing the first sampling point. min is a function that calculates the minimum value among three parameters.

The DTW(*T*, *Tv*) can be calculated by formula (12) according to formula (11).

DTW(*T*, *Tv*)=|DTW(*T*, *Tv*)| (12)

DTW 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: CornerDist(*T*, *Tv*) reflects the degree of motion direction deviation between the final moving object and the origin moving object shown as Figure 5(c). Corner comparison can be calculated by formula (13).

(13)

Here, the length of *T* and *Tv* respectively are *m* and *n*. CornerDist calculates the information loss degree by the speed corner of moving objects.

To verify the performance of VTC algorithm, it is compared with DP and TD-SP in this paper by compressing 5 trajectories (such as trajectory 1, 2048, 4876, 5199 and 8822 etc.). This section compares the algorithms across multiple performance metrics including fitting effect, compression speed and compression ratio as well as information loss degree.

|  |  |
| --- | --- |
|  | |
| DP、TDSP和MOSCC整体轨迹压缩效果图AB | (JT@Z4TMD(UEVP]N4~96CX9 |
| (a)Holistic compression effect figure | (b)Partial enlarged figure of A in Fig. (a) |
| DP、TDSP和MOSCC图（a）中B处的局部放大图CDE | OVW(2{2AR6)C6V7G($]RNDS |
| (c)Partial enlarged figure of B in Fig. (a) | (d)Partial enlarged figure of C in Fig. (c) |
| DUR41MXRWQ1O0K%`VTF[O@5 | 7WHFQK[UE)E4H1BH98}4G_G |
| (e)Partial enlarged figure of D in Fig. (c) | (f)Partial enlarged figure of E in Fig. (c) |
| Figure 6. Effect figure of DP(*ε*=0.001), TD-SP(*v*=3) and VTC(*β*=0.1745, *v*=2) compressing trajectory | |

As shown in Figure 6, VTC keeps not only the holistic external shape of origin trajectory (Fig. 6(a)) but also the local motion characteristics of moving objects in detail (Fig. 6(b), Fig. 6(c), Fig. 6(d), Fig. 6(e) and Fig. 6(f)). In Table 3, compared with DP and TD-SP, VTC has a faster compression speed, but a lower compression ratio than DP. There are two main reasons as follow: (1) VTC algorithm takes full consideration of the motion characteristics which is beneficial to the reservation of holistic motion characteristics. (2) VTC algorithm emphatically analyzes the characteristic information contained in trajectories which is beneficial to keep the local motion characteristics of moving objects.

Table 3 Performance comparison between different algorithms

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Algorithm name | Correlation parameters | Trajectory size | Compression time (ms) | Compression ratio (%) |
| DP | ε=0.001 | 73 | 12.0969 | 96.91 |
| TD-SP | v=3 | 170 | 20.3568 | 92.79 |
| VTC | β=0.1745、v=2 | 152 | 4.395 | 93.56 |

Table 4 Comparison of information loss degree

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Trajectory ID | Algorithm name | Correlation parameters | Compression ratio (%) | Information Loss | | |
| SED comparison | DTW comparison | Corner comparison |
| #1 | DP | *ε*=0.0001 | 87.90 | 0.003156 | 0.383372 | 0.320427 |
| TD-SP | *v*=2.5 | 88.44 | 0.002550 | 0.728261 | 0.334721 |
| VTC | *β*=0.65, v=2 | 88.44 | 0.002020 | 0.491092 | 0.319264 |
| #2 | DP | *ε*=0.0001 | 90.43 | 0.001881 | 1.187654 | 0.323714 |
| TD-SP | *v*=3 | 88.72 | 0.002598 | 1.535941 | 0.335769 |
| VTC | *β*=0.75, *v*=2 | 90.34 | 0.001991 | 1.443385 | 0.307635 |
| #3 | DP | *ε*=0.0001 | 94.57 | 0.001305 | 0.592190 | 0.326281 |
| TD-SP | *v*=2.5 | 94.53 | 0.002940 | 3.598822 | 0.327195 |
| VTC | *β*=0.2, *v*=4 | 96.06 | 0.002798 | 1.596501 | 0.293749 |
| #4 | DP | *ε*=0.0002 | 94.16 | 0.030045 | 2.744029 | 0.328640 |
| TD-SP | *v*=4 | 93.64 | 0.034009 | 29.719967 | 0.335451 |
| VTC | *β*=0.6, *v*=2 | 92.60 | 0.008030 | 4.028139 | 0.320782 |

To describe the experimental results more intuitionistic, this paper converts fitting effect into numeric by the calculation methods of information loss degree given in this section. Information loss degrees caused by DP, TD-SP and VTC with same or similar compression ratio are recorded in Table 4. The information loss degrees include SED comparison and DTW comparison which reflect holistic fitting effect of trajectories, as well as Corner comparison which reflects internal fitting effect of trajectories. Generally, the smaller information loss degree is, the better fitting 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-SP and VTC is same or similar, the Corner comparison of VTC is smallest. It indicates that VTC keeping internal characteristic and motion characteristics of moving objects has a better effect. The SED comparison of VTC is lower than TD-SP, while compared with DP, the SED comparison of VTC is both higher and lower. From the above, VTC keeping holistic characteristic of trajectories is superior to TD-SP and slightly inferior to DP. The DTW comparison of VTC is between DP and TD-SP in Table 4, which indicates VTC keeping holistic characteristic of trajectories is superior to TD-SP and slightly inferior to DP. Comprehensive analyzing the experimental results above, VTC can not only keep the motion characteristics and internal characteristic information of trajectories, but also keep the holistic characteristics of trajectories. As a consequence, VTC 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 VTC that determines retained points by velocity corner of moving objects to compress trajectory data. After that, VTC 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, VTC 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] Yukun Chen, Kai Jiang, Yu Zheng, Chunping Li, Nenghai Yu. Trajectory simplification method for location-based social networking service. In: Proceedings of the 2009 International Workshop on Location Based Social Networks (LBSN 2009), November 4-6, 2009, Seattle, Washington, USA, 33-40.

异常检测

[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.

聚类相关的，轨迹内部结构

[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.

路网

[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.

语义

[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.

在线

[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] Goce Trajcevski, Hu Cao, Peter Scheuermann, et al. On-line data reduction and the quality of history in moving objects databases. In: Proceedings of the 5th ACM international workshop on Data Engineering for Wireless and Mobile Access (MobiDE 2006). June 25, 2006, Chicago, Illinois, USA, 19-26.

MDL

[27] Thomas C. M. Lee. An introduction to coding theory and the two-part minimum description length principle. International Statistical Review, 2001, 69 (2), 169-184.

[28] Mark H. Hansen, Bin Yu. Model selection and the principle of minimum description length. Journal of the American Statistical Association. 2001, 96(454): 746-774.

[29] Georgios Kellaris, Nikos Pelekisb, Yannis Theodoridisc. Map-matched trajectory compression. Journal of systems and software. 2013, 86(6), 1566-1579.