A Turning Contour Maintaining Method of Trajectory Data Compression

Shihao Feng¹, Luo Chen²*, Mengyu Ma³ and Anran Yang⁴

¹,²,³,⁴National University of Defence Technology, College of Electronic Science, Changsha, China

*Corresponding author: luochen@nudt.edu.cn

Abstract. As the dramatical improvement of Global Position System (GPS) sensors in recent years, the research of trajectory dataset has become a hotspot in the field of geographic information systems (GIS). Pretreatment is very important for extracting useful information from massive trajectory dataset. One of key problems is trajectory data compression. Aiming at preserving crucial feature of trajectories while compressing, this paper puts forward a method for compressing trajectory data that combines distance, angle, and velocity while maintaining the contour of the trajectory, especially the turning corners and U-turns in the compressed trajectory and the original trajectory. So, the compressed trajectory can be very useful for trajectory mining and road network update. The new method also shows applicability and stability through experiments on different datasets.

1. Introduction

Nowadays, more and more position information with timestamp is obtained by various sensors. The trajectory data thus formed become an important data type in application [1]. The trajectory data help researchers to study the behavior characteristics of moving objects. The obtained information also can be used to analyse and mine the travel preferences of moving objects, and promote the technological development of intelligent transportation, smart cities, and the update of road network [2].

At the present stage, a large number of sensors continuously generate a large amount of trajectory data, which has a lot of redundancy in the obtained original data. The process of transmitting and storing the original trajectory data requires too much network bandwidth and storage capacity [3]. This brings huge challenges to the management and transmission of data and is not conducive to the analysis using trajectory data. Therefore, trajectory data compression becomes an important issue. The main idea of the existing trajectory compression algorithm is to simplify the trajectory, discard data that has little or no impact on the trajectory, and then construct a new trajectory from the remaining data to replace the original trajectory. The advantages of these algorithms are that they can greatly reduce the amount of data while setting an acceptable error range, they can save valuable data in the trajectory data set to a certain extent, and facilitate the management and storage of the trajectory data [4].

However, most trajectory compression algorithms may lose critical trajectory information during the process of compression, especially the contours of the turning corners and U-turns of moving objects, which are important for the reproduction of the trajectory and the display of trajectory behaviour of moving objects [5]. Moreover, this is not conducive to using the compressed trajectory data for behaviour analysis and data mining (for example, using the trajectory data to update the road network). Therefore, it would make sense to find a trajectory compression method that maintains the
turning corners and U-turn of the trajectory data. This paper proposes a trajectory compression method that maintains contours of each trajectory. With this method, the trajectory data can be compressed on the basis of maintaining the trajectory turning corners and U-turns. Experiment indicates this method has better contour preservation effect under the same threshold condition, and can get satisfying results.

2. Related work

The research of the existing trajectory compression algorithms mainly focuses on deleting redundant and unnecessary data, and saving critical data in the trajectory [4]. It can be divided into two categories: lossless compression and lossy compression. Most lossless compression algorithms only eliminate the spatial and temporal redundancy to accurately reconstruct the original data without losing its trajectory information, but its reduction in the amount of trajectory data is relatively limited. Lossy compressions can greatly reduce the amount of data and retain only the key data in the trajectory. The basic lossy compression technology is the trajectory simplification algorithm. Trajectory simplification algorithms are mainly divided into two categories which are offline compression algorithms and online compression algorithms according to different application scenarios [6].

2.1. Offline compression algorithms

The offline compression algorithm requires all the data of the trajectory stored in the database, and makes full use of the characteristics of the trajectory for compression. Douglas and Peucker proposed an algorithm that uses a distance threshold to preserve directional trends in the approximation line using a distance threshold, which may be varied according to the amount of simplification required (DP algorithm) [7]. Nirvana Meratnia et al. improved the calculation of the distance threshold in the DP algorithm by considering the timestamp, and named it the TD-TR algorithm [8]. Minjie Chen et al. proposed a polygonal approximation algorithm (MRPA algorithm) in 2-D space for the GPS trajectory simplification under the so-called integral square synchronous distance error criterion in a linear time complexity [9]. H.V. Jagadish et al. first proposed a lossy trajectory compression algorithm (SP algorithm) for retaining direction, which aims to preserve the directional information of the trajectory while also retaining the position information [6].

2.2. Online compression algorithms

The online compression algorithm can compress the data of the track points in real time, which has the advantage of supporting online applications. Nirvana Meratnia et al. first proposed a window-based online trajectory simplification algorithm (OPW algorithm). Then they add the timestamp into consideration and improved the distance threshold to proposed a new improved algorithm (OPW-TR algorithm) [8]. Goce Trajcevski et al. proposed an online compression algorithm that uses the position and velocity of the current trajectory point to predict the next trajectory point, then compares the predicted data with the received data to determine whether the received data is retained. This method works well in batch processing mode [10]. M. Potamias et al. proposed a Threshold algorithm, which uses the intersection of the fan-shaped regions formed by the velocity and direction of the last two points in the current trajectory and the compressed trajectory as the safe region for predicting the trajectory points. Then it is judged whether the trajectory point falls in this security area, and decides whether to keep the trajectory point and achieve the purpose of simplifying the trajectory [11]. Jonathan Muckell et al. proposed the Spatial QUALity Simplification Heuristic (SQUISH) method that demonstrates improved performance when compressing up to roughly 10% of the original data size, and preserves velocity information at a much higher accuracy under aggressive compression [12].

The trajectory compression efficiency of the above algorithms is relatively high, but the contour of the trajectory is not well controlled during the compression process. The trajectory compression using the above algorithms will cause the loss of the trajectory contour, and the complete trajectory contour can be reflected in detail in the behavioural characteristics and action patterns of moving objects [13].
so the incomplete trajectory cannot reflect the detailed information of the behaviour of moving objects, which will cause the loss of trajectory semantics [14]. At the same time, the incompleteness of the trajectory contour has a certain impact on the road network update using the trajectory dataset. In order to solve this problem, this paper proposes a trajectory compression algorithm to maintain the turning corners and U-turns.

3. Proposed method
The key point of the method proposed is compressing trajectory based on three variables which are velocity, distance and angle. Velocity is set to reduce some spatial and temporal redundancy. Distance is used as the main factor for simplifying the trajectory data. Angle is used as the factor for trajectory turning points and points of U-turns, it can help us to maintain the basic contour of the turning points.

3.1. Problem definitions

3.1.1. Velocity
In this paper, we take the ratio of the distance and the time difference between the two points as the velocity of the first point. Then filter the trajectory data and delete the duplicated data with small or large velocity changes.

3.1.2. Distance
Generally, there are two methods for distance calculation of trajectory data processing, Perpendicular Euclidean Distance (PED) [7] and Synchronized Euclidean Distance (SED). they are shown in Fig.1. The PED of the point is the shortest distance between the point and the vertical point of the simplified path. The SED of the point is the distance between the currently real point and the synchronized point acquired by interpolating [8]. PED can better preserve the spatial characteristics of the trajectory data. However, SED takes the information of time into consideration. In this paper we select SED as the threshold of the distance, so we can keep the timestamp of the trajectory when we discuss the distance. The calculation method of SED is shown in (1).

\[ \Delta e = t_B - t_A \]
\[ \Delta e = t_C - t_A \]
\[ t'_B = t_C \]
\[ x'_B = x_C + \frac{\Delta e}{\Delta e} (x_B - x_A) \]
\[ y'_B = y_C + \frac{\Delta e}{\Delta e} (y_B - y_A) \]

\[ SED = \sqrt{(x_B - x'_B)^2 + (y_B - y'_B)^2} \]

In (1), the original trajectory point is \( P_C \) and its approximate point \( P'_C \) on the segment from trajectory point \( P_A \) to trajectory point \( P_B \), \((x'_B, y'_B)\) is the coordinates of point \( P'_C \) and they are calculated based on the ratio of time intervals \( \Delta e \) and \( \Delta e \). SED is calculated based on distance formula between two points.

![Fig. 1. The difference of SDE and PED](image-url)
3.1.3. Angle
The angle between the connection line of first point and second point and the connection line of second point and third point. $\theta$ is the angle of point $P_C$, it can be calculated by (2).

$$\theta = \arccos\left(\frac{d_1^2 + d_2^2 - d_3^2}{2 \times d_1 \times d_2}\right) \quad (2)$$

In (2), $d_1$, $d_2$, and $d_3$ respectively represent the distances between the trajectory points $P_A$, $P_C$ and $P_B$. It is illustrated in Fig.2.

3.2. Method description
The method used in this paper is based on a top-down manner. After sorting the trajectory data by time, we calculate the velocity, SED and angle defined above. Then we set the corresponding threshold to divide the trajectory data sequence in a top to bottom manner, then select points that meet the threshold as the split points. In this way we can get two subsequences. Next, we perform the above steps on each subsequence recursively, until all trajectory points that meet the threshold requirements are selected. The details of the whole process are as follows:

1) The trajectory data generally contains the latitude, longitude and time information of trajectory points. From the above information, the approximate velocity of each point can be calculated. Because there are some kind of errors in the trajectory data, intuitively, the points with too low velocity can be regarded as the stay points, and the points with too high velocity can be regarded as the drift points. The stay points will make the data redundant, and drift points can cause errors in trajectory. Therefore, by setting the velocity threshold, the stopping and drifting points can be filtered out.

2) First sort the trajectory dataset after velocity screening in chronological order, then select the first point and last point in the dataset as the start point and end point. Next, all trajectory points are connected into lines in order. By traversing from the second point, the SED and angle($\theta$) of each point to the connected line is calculated from the start point and end point.

3) Compare the calculated value of SED and $\theta$ with the threshold we set already. When the result is higher than the SED threshold or lower than the angle threshold, this point is selected as the index point, and the trajectory data is segmented into two parts from this point, one is from the start point to the index point, and the other is from the index point to the end point. Then re-execute the previous step in the two-part trajectory to find the index points.

4) Put the start point, end point and all filtered index points into the dataset. The dataset contains all the final compressed data, and the entire trajectory compression is completed. The whole process can be simply illustrated with Fig.3. In Fig.3, the original trajectory is $T = \{P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8\}$, and the trajectory can be compressed to $T' = \{P_1, P_4, P_8\}$. 

![Fig. 3. The compressed of the trajectory](image-url)
3.3. Pseudocode
The pseudocode of the trajectory compression method that maintains the turning contour are as follows:

```
Method: the turning contour maintaining method of trajectory compression

Input:
- The original trajectory: \( T = \{P_1, P_2, ..., P_N\} \)
- Velocity threshold: lower limit \( \bar{\theta}_1 \), upper limit \( \bar{\theta}_2 \)
- Distance threshold: SED
- Angle threshold: \( \theta \)

Output:
- The Compressed trajectory: \( T' = \{P'_1, P'_2, ..., P'_N\} \)

1: \( T' \leftarrow [ ] \)
2: \( T'' \leftarrow [ ] \)
3: \( \text{index point} \leftarrow [ ] \)
4: while (i < N-1) {
   5:   \( \{ \)
   6:   \( V_i \leftarrow \text{Get velocity}(P_i, P_{i+1}) \) // Calculate velocity of the point
   7:   if \( (V_i < \bar{\theta}_1 || V_i > \bar{\theta}_2) \)
   8:      delete the point
   9:   else
10:      \( T'' \leftarrow T'' \) append \( P_i \) // Get point with velocity filtered
11: \} \)
12: \( P_{\text{start}} \leftarrow P_i \)
13: \( P_{\text{end}} \leftarrow P_{\text{len}(T'')} \)
14: for \( 1 < j < \text{len}(T'')-1 \) {
15:   \( d = \text{Get SED}(P_{\text{start}}, P_j, P_{\text{end}}) \) // Calculate SED of the point
16:   \( a = \text{Get degree}(P_{\text{start}}, P_j, P_{\text{end}}) \) // Calculate angle of the point
17:   if \( (d > \text{SED} || a < \theta) \)
18:      \( P_{\text{index}} \leftarrow P_j \)
19:      index point \leftarrow index point append \( P_{\text{index}} \)
20:   do 13-17 in \( T''[P_{\text{start}}, P_{\text{index}}] \)
21:   do 13-17 in \( T''[P_{\text{index}}, P_{\text{end}}] \)
22: \( T' \leftarrow T' \) append \( P_{\text{start}}, P_{\text{end}}, \) index point
23: return \( T' \)
```

4. Experiments and Results
In this section, we compare the turning contour maintaining method of trajectory compression proposed in this paper with several other typical algorithms, which are: DP, OPW, OPW-TR, TD-TR and SQUISH-E. We select experiment data from three different datasets, (see Table 1). Experiments were performed with the average synchronized Euclidean distance error (ASEDE) and visualization results as the basis for data analysis.
Table 1. Trajectory Dataset in the Paper

| Trajectory number | source                        | points | sampling interval/s | type    |
|-------------------|-------------------------------|--------|---------------------|---------|
| Trajectory 1      | Didi Chuxing GAIA Initiative | 172    | 3                   | Vehicle |
| Trajectory 2      | OpenStreetMap                 | 336    | 1-2                 | Walking |
| Trajectory 3      | GeoLife                       | 1832   | 1-5                 | Mix     |

4.1. Datasets

We use three datasets in the experiments. Trajectory 1 comes from Didi Chuxing GAIA Initiative. It is a vehicle trajectory data of Xi’an in 2018 and contains 172 points with a sampling interval of 3 seconds. Trajectory 2 comes from OpenStreetMap, we selected a walking trajectory data and it contains 336 points with a sampling interval of 1 to 2 seconds. Trajectory 3 comes from GeoLife dataset which is from the Microsoft Research GeoLife project. The trajectory data we selected is a mix trajectory which contain walking and vehicle. There are 1,832 points in the trajectory, and the sampling interval is 1 to 5 seconds. The three data all contain the longitude, latitude, and time stamps of the trajectory points.

4.2. Settings and environment

The experiment was programmed in Python. PostgreSQL was used as the Database Management System (DBMS) to store the trajectory data, and QGIS was used to visualize the trajectory. In order to evaluate the compression accuracy of the trajectory compression algorithms, the ASEDE was selected to evaluate the algorithm. The calculation formula is defined in (3).

Fig.4 ASEDE of different distance thresholds under six methods
and represents the original trajectory and the compressed trajectory, and \( P_i \) represents the points in the trajectory.

\[
ASEDE = \frac{1}{n} \sum_{i=1}^{n} SED(T, T', P_i)
\]  

(3)

### 4.3. Results and analysis

There are three groups of variables in the method proposed: velocity, distance and angle. The traditional methods are all based on distance threshold. Therefore, before the comparison of all the methods, the velocity threshold and the angle threshold in the paper are set in advance. The data used in paper includes walking and vehicles trajectories, so we set the upper limit of the velocity threshold to 25m per second and the lower limit to 3m per second. It is found that when the angle threshold of trajectory 1 is 30°, the angle threshold of trajectory 2 is 150°, and the angle threshold of trajectory 3 is set to 90°, the effect of maintaining the turning contour is better. Then we choose these thresholds when we compare the methods under different distance thresholds.

In Fig.4, we can get the performance of six different algorithms at different distance thresholds. ASEDE reflects the trajectory of the original trajectory of compression on the value of the degree of similarity, it means that the compressed trajectory is closer to the original trajectory when ASEDE is smaller. Therefore, experiment results of trajectory 1 show that, our proposed method have batter performance compared with other three methods OPW-TR, SQUISH-E, TD-TR, while DP and OPW perform poorly. In trajectory 2 which is a walking trajectory, it can be concluded that within the range of 10 to 100 meters from the distance threshold, the trajectory compressed by using our method and SQUISH-E is the best, and our method is even better than SQUISH-E. In trajectory 3 which is a mixed trajectory, it can be seen that our method is just better than DP and OPW, and slightly worse than the remaining three algorithms. Therefore, in general, compared with the traditional five trajectory compression algorithms, the proposed method has less error in the single trajectory of the vehicle and walking trajectory, and the performance is centered for the mixed trajectory.

We also get the visualization results of the trajectory. Fig.5 is a visual comparison chart of trajectory 1. It can be seen that for the turning trajectory of the part 1, the turning corners contours are well preserved with our proposed method, while DP and OPW fail to do this. In the part 2, DP and OPW cannot maintain the U-turn of the trajectory, and the remaining four methods perform better. For the U-turn of the part 3, although all the six methods can maintain the contour of trajectory 1, the compressed trajectory using our method is smoother and closer to the original trajectory.

Fig.6 shows the visual comparison result of trajectory 2. This trajectory is a walking trajectory. It can be clearly seen from the results that when DP, OPW, OPW-TR, and TD-TR are executed, corner contour information is lost.
Only the method of this paper and SQUISH-E perform better, and our method can handle smoother when dealing with turning corners than SQUISH-E.

Fig.7 is the visual comparison result of trajectory 3. This trajectory is a mixed trajectory composed of walking and vehicle trajectory. As can be seen from Fig.7, in the part 1, the process of DP and OPW caused the trajectory loss, and in the part 2, the trajectory is a U-turn trajectory, and there is a lot of redundant in it. Our method removed the redundant data of this part which other five algorithms did not. This is also the reason why the ASEDE of our method is higher in the comparison of the ASEDE in the trajectory 3 shown in the Fig.4.

5. Conclusions and future work
This paper proposes a trajectory compression method based on the threshold of velocity, distance and angle, which can well maintain the trajectory contour of turning corners and U-turns after compression. However, there are still some shortcomings in this paper:

1) Although the method in this paper can maintain the contours of turning corners and U-turns, the compression rate is lower than the traditional algorithms.

2) The method in this paper uses three different thresholds for calculation, and its time complexity is higher than the traditional algorithms.

In the next work, we are considering further enhancing the compression ratio of this method, improving the efficiency of calculation by introducing other methods, and reducing the time complexity of the method to improve the efficiency and applicability of the method. At the same time, the compressed trajectory obtained by this method is used to update the road network for further data analysis.
Acknowledgments
Thanks to the data source supported by: Didi Chuxing GAIA Initiative. This work was supported by the National Natural Science Foundation of China (41971362) and The Natural Science Foundation of Hunan Province (2019JJ50718)

References
[1] Qingfu Fan, Lei Zhang, and Leijun Liu. *Online GPS trajectory data compression based on offset calculation*. Computer Engineering and Applications 53.8 (2017): 254-259.
[2] Penghui Sun, Shixiong Xia, GuanYuan, and Daxing Li. *An overview of moving object trajectory compression algorithms*. Mathematical Problems in Engineering 2016 (2016).
[3] Haozhou Wang, Kai Zheng, Jiajie Xu, Bolong Zheng, and Xiaofang Zhou. *Sharkdb: An in-memory column-oriented trajectory storage*. Proceedings of the 23rd ACM international conference on conference on information and knowledge management. (2014).
[4] MENG Qingbin, YU Xiaoqiang, LIU Bai, and SHAO Li. *A contour maintaining oriented trajectory data compression algorithm*. Journal of Dalian Polytechnic University 2 (2018): 14.
[5] Cheng Long, Raymond Chi-Wing Wong, and H. V. Jagadish. *Direction-preserving trajectory simplification*. Proceedings of the VLDB Endowment 6.10 (2013): 949-960.
[6] Douglas David H., and Thomas K. Peucker. *Algorithms for the reduction of the number of points required to represent a digitized line or its caricature*. Cartographica: the international journal for geographic information and geovisualization 10.2 (1973): 112-122.
[7] Meratnia Nirvana, and A. Rolf. *Spatiotemporal compression techniques for moving point objects*. International Conference on Extending Database Technology. Springer, Berlin, Heidelberg, (2004).
[8] Chen Minjie, Mantao Xu, and Pasi Franti. *A fast O(N) multiresolution polygonal approximation algorithm for GPS trajectory simplification*. IEEE Transactions on Image Processing 21.5 (2012): 2770-2785.
[9] Trajcevski Goce, Hu Cao, Peter Scheuermanny, Ouri Wolfsonz, and Dennis Vaccaro. *On-line data reduction and the quality of history in moving objects databases*. Proceedings of the 5th ACM international workshop on Data engineering for wireless and mobile access. (2006).
[10] Potamias Michalis, Kostas Patroumpas, and Timos Sellis. *Sampling trajectory streams with spatiotemporal criteria*. 18th International Conference on Scientific and Statistical Database Management (SSDBM'06). IEEE, (2006).
[11] Muckell Jonathan, Paul W.Olsen Jr., Jeong-Hyon Hwang, Catherine T.Lawson, and S.S.Ravi. *Compression of trajectory data: a comprehensive evaluation and new approach*. GeoInformatica 18.3 (2014): 435-460.
[12] GAO Qiang, ZHANG Feng-Li, WANG Rui-Jin, and ZHOU Fan. *Trajectory Big Data: A Review of Key Technologies in Data Processing*. 2016.
[13] Xu Xiujuan, Xiaobo Gao, Xiaowei Zhao, Zhenzhen Xu, and Huajian Chang. *A novel algorithm for urban traffic congestion detection based on GPS data compression*. 2016 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI). IEEE, (2016).
[14] Hao, Pengfei, Yao, Chunlong, Meng, Qingbin. Yu, Xiaoliang, and Li, Xu. *A Heading Maintaining Oriented Compression Algorithm for GPS Trajectory Data*. Informatica 30.1 (2019): 33-52.