INFORMATICA, 2019, Vol. 30, No. 1, 33–52

DOI: http://dx.doi.org/10.15388/Informatica.2019.196

A Heading Maintaining Oriented Compression Algorithm for GPS Trajectory Data

Pengfei HAO, Chunlong YAO*, Qingbin MENG, Xiaoqiang YU, Xu LI
School of Information Science and Engineering, Dalian Polytechnic University, Dalian, China
e-mail: yaocl@dlpu.edu.cn

Received: October 2017; accepted: November 2018

Abstract. The raw trajectories contain large amounts of redundant data that bring challenges to storage, transmission and processing. Trajectory compression algorithms can reduce the number of positioning points while minimizing the loss of information. This paper proposes a heading maintaining oriented trajectory compression algorithm, which takes into account both position information and direction information. By setting an angle threshold, the algorithm can achieve a more accurate approximation of trajectories than traditional position-preserving trajectory compression algorithms. The experimental results show that the algorithm can ensure certain effect on the direction information and is more flexible.

Key words: trajectory compression, heading maintaining, flexible.

1. Introduction

In recent years, with the popularization and application of GPS-enabled devices, massive volumes of GPS trajectory data are recorded. These data hide interesting and valuable knowledge patterns describing the behaviour of moving objects (Morzy and Rosikiewicz, 2007). Individual trajectories can reflect professional characteristics of the user, trajectories with high degree of spatio-temporal regularities on weekdays imply the user has a fixed full-time job, trajectories with irregular spatio-temporal characteristics in a week imply work unit of the user is not fixed (Li *et al.*, 2014). Group trajectories contain a lot of valuable knowledge patterns. According to these patterns, people can find restaurants and travel routes that interest them. Urban function units (e.g. residential area, commercial area and industrial area) also can be distinguished by using these patterns (Li *et al.*, 2014).

Typically, the size of raw trajectory data recorded by GPS-enabled devices is very large. Consider Beijing with 67,000 taxis, suppose we collect trajectory data in every 2–3 seconds, the size of the trajectories generated by these taxis for just a single day is as high as 60TB (Yuan, 2012). Such massive volumes of data will bring some problems for location-based applications. (1) It will take up a lot of storage spaces. (2) Sending a large amount of GPS trajectory data will cause network overhead. (3) It will bring a heavy

*Corresponding author.

burden for web browsers when rendering these trajectories on the client side. It takes approximately 1s for rendering 1000 points on the map (Chen *et al.*, 2012). (4) When the size of trajectory data gets larger, it becomes more difficult to find valuable patterns from the trajectory data. Reducing the size of trajectory data has the potential to make it more easily (Giannotti *et al.*, 2007).

1.1. *Existing Trajectory Compression Algorithms*

To solve the trajectory compression problem various algorithms are proposed, each offers a different method to compress trajectory data. In this section, we will briefly introduce the related algorithms of trajectory compression.

Douglas-Peucker (DP) algorithm (Douglas and Peucker, 1973) is widely used in computer aided design (CAD) area, it can be employed to compress trajectory data. Given a trajectory T and a parameter of spatial error μ, DP algorithm constructs the approximate trajectory T' by repeatedly adding points from T until the maximum spatial error of T' becomes smaller than μ. DP algorithm can effectively compress trajectory data, but it only focuses on spatial information. TD–TR (Meratnia and de By, 2004) is an improvement algorithm of DP, which uses SED error instead of spatial error. Compared to DP algorithm, TD–TR algorithm has the benefit of taking temporal information into account. Optimal algorithms (Perez and Vidal, 1994; Salotti, 2000; Salotti, 2001) aim at minimizing spatial error, by removing positioning points in searching process they can achieve a minimum spatial error compression. Due to the computational overhead of the optimal algorithms, near-optimal algorithms are proposed to reduce the time complexity. Near-optimal algorithms proposed in papers (Kolesnikov and Franti, 2002; Kolesnikov and Franti, 2003) can achieve a faster search by reducing the search space and using heuristic search. Paper (Tian and Xu, 2011) proposes a trajectory compression method based on turning point judgment method, this algorithm uses turning points to delineate a trajectory, in which the advantages and disadvantages of point-by-point judgment method and combined judgment method are analysed. Trajectory simplification (TS) algorithm is proposed in paper (Chen *et al.*, 2009), which considers both the shape skeleton and the semantic meanings of a GPS trajectory. TS algorithm uses heading change degree of a point and distance between this point and its most adjacent neighbours to weight importance of the point, the points with high weight will be retained in final compressed trajectory. Open Window Time Ratio (OPW-TR) (Meratnia and de By, 2004) is a compression algorithm for trajectory data, given a trajectory T and a parameter of SED error μ, OPW-TR algorithm starts by defining a segment between the first point P1 (the anchor point) and the third point P3 (the float point). Then the algorithm calculates all SED errors of the points between the anchor point and the float point. As long as all SED values are below μ, an attempt is made to move the float point one point forward in trajectory T. When the SED threshold μ is going to be exceeded, a new anchor point will be chosen out. The process will repeat until the entire trajectory has been transformed into a piecewise linear approximation. Based on the different anchor point select strategies, OPW-TR has two modes called Before-OPW-TR and Normal-OPW-TR.

Threshold-guided algorithm (Potamias *et al.*, 2006) compresses trajectory by constructing a safe area using moving object's speed and direction, if an incoming positioning point is in the safe area, then this point contributes little information and hence can be discarded without significant loss in accuracy. Spatial QUalIty Simplification Heuristic – Extended (SQUISH-E) algorithm (Muckell *et al.*, 2014) is an extended version of SQUISH algorithm (Muckell *et al.*, 2011). Compared to SQUISH algorithm, SQUISH-E algorithm has the capability of compressed trajectories while ensuring that SED error is within a user-specified bound. The main idea of SQUISH-E algorithm is to construct a priority queue of positioning points, this algorithm compresses trajectory by repeatedly removing the positioning point of the lowest priority until the termination condition is met. And based on parameters setting, SQUISH-E can be divided into SQUISH-E(μ) and SQUISH-E(λ). A hybrid trajectory compression algorithm based on multiple spatiotemporal characteristics is proposed in paper (Jiagao *et al.*, 2015), this algorithm uses characteristic points to delineate a trajectory, and the characteristic points are chosen by using both distance, direction and speed information. Articles (Qian and Lu, 2017; Sun *et al.*, 2016) are a review of related algorithms and have certain reference value. Document (Cao and Li, 2017) proposes DOTS (Directed acyclic graph based Online Trajectory Simplification) algorithm and constructs a directed acyclic graph on-line simplification.

1.2. *The Proposed Algorithm*

The positioning points where an object changed moving direction contain rich semantic meanings, these points imply user's behaviour, such as, staying, taking photos or watching something attractive, etc. Without these positioning points, we will know little about user's behaviour. In this paper, we propose a heading maintaining oriented trajectory compression algorithm called HMOTC.

The HMOTC algorithm has the benefit of taking into account both the heading change degree of positioning points and the direction change degree between positioning points and trajectory segment. So, this algorithm can accurately capture the direction information of the trajectory. As shown in Fig. 1, a multi transportation mode (walk $\rightarrow$ bus $\rightarrow$ walk $\rightarrow$ bus $\rightarrow$ walk $\rightarrow$ train) GPS trajectory from north to south is compressed by DP, TD-TR and our HMOTC. For traditional position-preserving compression algorithms DP and TD-TR, we could know little about how the trajectory's user walked on walk segment from the compressed trajectories, because they lose too much of the moving object's heading information. For HMOTC algorithms, because the direction information of the trajectory is taken into account, we can know more details about user's walking direction and walking route from the compressed trajectories.

Another benefit of HMOTC algorithm is that the algorithm has the ability of controlling compression with respect to spatial error. Without spatial information control it may lead to an inaccuracy compression. To illustrate that, let us take DPTS-SP-Prac (Long *et al.*, 2013) algorithm as an example. DPTS-SP-Prac is a direction-preserving trajectory compression algorithm which focuses on capturing the direction change degree between positioning points and trajectory segment, the algorithm has a theoretical bounded posi-

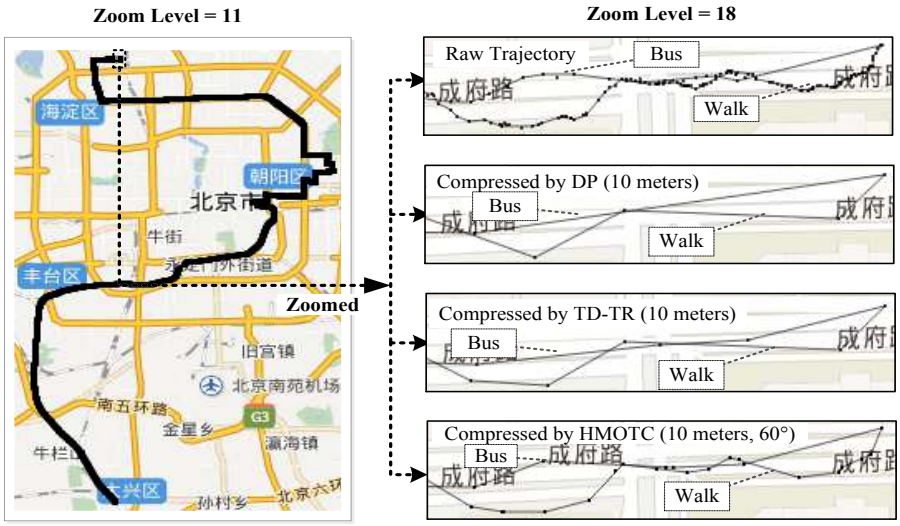


Fig. 1. A multi transportation mode GPS trajectory and it's compressed representations.

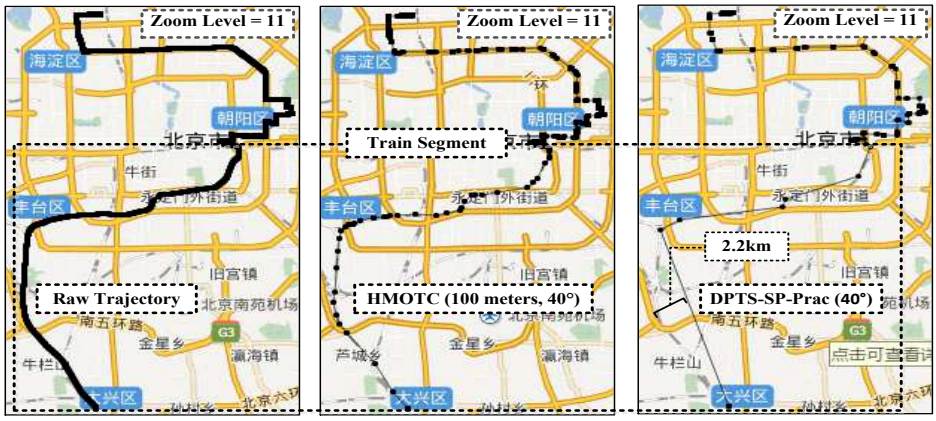


Fig. 2. The multi transportation mode trajectory after being compressed by DPTS-SP-Prac and HMOTC.

tion error, but it is uncontrollable. As shown in Fig. 2, the multi transportation mode trajectory is compressed under a 40° angle threshold. The DPTS-SP-Prac algorithm maintains a good trajectory contour in the walk segment and the bus segment. However, as the train segment has the characteristics of high speed and small direction change, a lot of spatial information is lost, the maximum perpendicular distance error of DPTS-SP-Prac in train segment is as high as 2.2 km (refer to Fig. 2, dashed box area). For the proposed algorithm, because it has the ability of spatial information control, the algorithm can maintain a good trajectory contour in walk, bus and train segments.

The HMOTC has the flexibility of controlling compression with respect to both spatial error and direction error, so the compression process can be precisely controlled accord-

ing to users' needs. If only the direction information of the trajectory is taken into account, the SED threshold of the algorithm can be set to positive infinity. If only the spatial information of the trajectory is taken into account, the angle threshold of the algorithm can be set to 180°. Because the algorithm can compress trajectory data while retrieving points from trajectory, it has the advantages of supporting both online and offline applications.

The remainder of this paper is organized as follows. In Section 2, some related definitions about trajectory compression are reported. After that our HMOTC algorithm is described in Section 3. An empirical evaluation of trajectory compression algorithms with different error measurements is given in Section 4. Finally, paper conclusions are discussed in Section 5.

2. Related Definitions

According to the loss of information, trajectory compression algorithms can be classified into two categories: lossless and lossy compression. Lossless compression algorithms enable reconstruction of the original trajectory data without information loss. Lossy compression will cause information loss, the trajectory data will be changed and can not be reconstructed. Usually, the compression efficiency of lossless compression algorithms is not obvious. For example, the compression efficiency of a lossless compression algorithm proposed in Zheng *et al.* (2005) is 25%. In contrast, lossy compression can significantly reduce the data size while maintaining an acceptable error tolerance. Due to the advantage of lossy compression, this paper focuses on lossy compression of trajectory data.

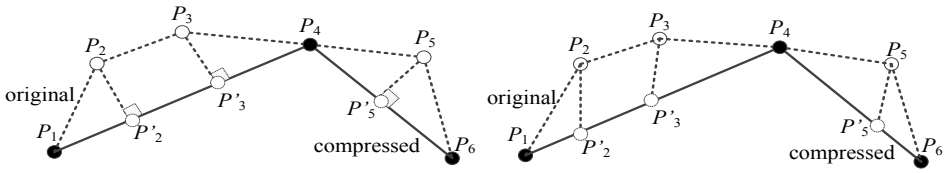
Generally, a GPS trajectory T is a temporally ordered sequence of positioning points $T = P_1, P_2, P_3, \ldots, P_{n-1}, P_n$, each point P_i consists of three tuples $\langle X_i, Y_i, t_i \rangle$, where X_i, Y_i represent the coordinate at time stamp t_i. The problem of lossy trajectory compression is to seek a set of temporally ordered points $T' = P_{i1}, P_{i2}, P_{i3}, \ldots, P_{i[m-1]}, P_{im}$ (a subset of T) as an approximation of T, where $1 = i_1 < \cdots < i_m = n$. And the definition of compression ratio CR in this paper is defined as

$$CR = \frac{n}{m}, \quad n \geqslant m. \tag{1}$$

2.1. *Error Measures*

There are many error measures to evaluate the accuracy of trajectory compression algorithms. In this section, we will introduce four error measures called spatial error, Synchronous Euclidean Distance (SED) (Potamias *et al.*, 2006) error, speed error and heading error.

Definition 1. Spatial error: Given a trajectory T and its compressed representation T', the spatial error of T' with respect to a point P_i in T is defined as the distance between P_i (x_i, y_i, t_i) and its estimation P_i' (x_i', y_i', t_i). If T' contains P_i, then $P_i' = P_i$. Otherwise, the closest point to P_i is defined as P_i' which is along the line between predT' (P_i) and succT$'$



(a) Spatial error (b) Synchronous Euclidean distance

Fig. 3. Spatial and SED errors are illustrated by using $T = P_1, P_2, P_3, P_4, P_5, P_6$ and $T' = P_1, P_4, P_6$.

(P_i), where predT′ (P_i) and succT′ (P_i) denote P_i's closest predecessor and successor among the points in T' (Muckell *et al.*, 2014).

Definition 2. SED error: Synchronous euclidean distance is an error measure which considers both spatial and temporal information. The definition of SED error is similar to spatial error, the SED error of T' with respect to a point P_i in T is defined as the distance between P_i (x_i, y_i, t_i) and its estimation P_i' (x_i', y_i', t_i). The difference is the coordinate x_i', y_i' of P_i' (x_i', y_i', t_i) are defined using formulas (4) and (5), where P_s $(x_s, y_s, t_s) =$ predT′ (P_i) and P_e $(x_e, y_e, t_e) =$ succ T′ (P_i).

$$\Delta t_1 = t_e - t_s, \tag{2}$$

$$\Delta t_2 = t_i - t_s, \tag{3}$$

$$\Delta x\prime_i = x_s + \left(\frac{\Delta t_2}{\Delta t_1}\right) \cdot (x_e - x_s), \tag{4}$$

$$\Delta y\prime_i = y_s + \left(\frac{\Delta t_2}{\Delta t_1}\right) \cdot (y_e - y_s). \tag{5}$$

For instance, in Fig. 3(a), spatial errors of P_1, P_4, P_6 are zero and spatial error of P_2 is the perpendicular distance from P_2 to line P_1P_4. In Fig. 3(b), SED errors of P_1, P_4, P_6 are zero and SED error of P_2 is the distance between P_2 and P_2'.

Definition 3. Speed error: Given a trajectory T and its compressed representation T', the speed error of T' with respect to a point P_i (x_i, y_i, t_i) in T is defined as the absolute difference value between Speed (P_i) and AverageSpeed (P_sP_e), where $P_s(x_s, y_s, t_s) =$ pred$T'(P_{i+1})$, $P_e(x_e, y_e, t_e) =$ succ$T'(P_i)$. P_i's speed and average speed of segment P_sP_e are defined as follows:

$$Speed(P_i) = Distance(P_i, P_{i+1})/(t_{i+1} - t_i),\ P_i \ddagger P_n, \tag{6}$$

$$AverageSpeed(P_sP_e) = Distance(P_s, P_e)/(t_e - t_s). \tag{7}$$

Definition 4. Heading error: given a trajectory T and its compressed representation T', the heading error of T' with respect to a point P_i in T is defined as heading change between Heading (P_i) and Heading (P_sP_e), where $P_s =$ pred$T'(P_i + 1)$, $P_e = succT'(P_i)$. P_i's

heading, heading of segment $P_s P_e$ and HeadingChange (h_1, h_2) are defined as follows:

$$Heading(P_i) = \overrightarrow{P_i P_{i+1}},\ P_i \ddagger P_n, \tag{8}$$

$$Heading(P_s P_e) = \overrightarrow{P_s P_e}, \tag{9}$$

$$HeadingChange(h_1, h_2) = \begin{cases} 360^\circ - |h_1 - h_2|, & |h_1 - h_2| > 180^\circ, \\ |h_1 - h_2|, & |h_1 - h_2| \leqslant 180^\circ. \end{cases} \tag{10}$$

3. HMOTC Algorithm

A key feature of our HMOTC algorithm is that it takes into account the direction information of GPS trajectories, the algorithm controls both the heading change degree of positioning points and the direction change degree between positioning points and trajectory segment.

3.1. *Algorithm Description*

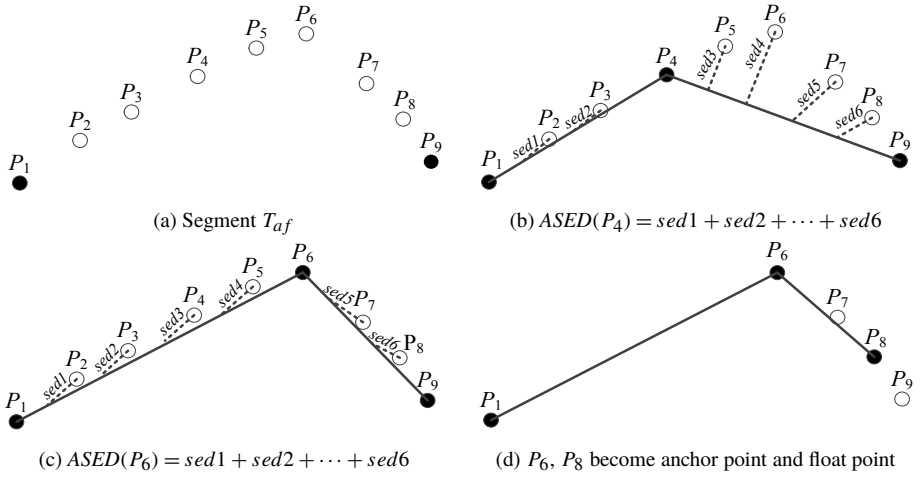
Given a trajectory T, a SED threshold μ and an angle threshold θ, HMOTC algorithm starts by defining a segment between the first point P_1 and the third point P_3 in T, where P_1 is called anchor point P_a and P_3 is called float point P_f. Then the following steps are applied in segment (window) $P_a P_f$:

(1) Checking whether predT(P_f) is a turning point by calculating heading change α between points predT(P_f) and predT(predT(P_f)). If $\alpha > \theta$ (i.e. predT(P_f) is a turning point), then predT(P_f) becomes the new anchor point and the second point behind predT(P_f) becomes the new float point.

(2) It is not enough to capture the heading change of a moving object by calculating heading change of a single point, this policy is vulnerable to error propagation, as a moving object could change its heading gradually. In order to address this problem, the heading change of any two points in segment $P_a P_f$ (except point P_f) and the heading change between segment $P_a P_f$ and each point in segment $P_a P_f$ (except point P_f) are calculated. If there is a heading change value greater than the given angle threshold θ, then the point (between P_a and P_f) of the minimum Accumulated Synchronous Euclidean Distance (ASED) error becomes the new anchor point and the second point behind it becomes the new float point.

(3) For maintaining shape skeleton of the trajectory and minimizing the loss of spatiotemporal information, SED errors of the points between P_a and P_f are calculated. If there is a SED error value greater than the given SED threshold μ, then the point (between P_a and P_f) of the minimum ASED error becomes the new anchor point and the second point behind it becomes the new float point.

As long as the anchor point P_a is not changed, an attempt is made to move the float point P_f one point forward in trajectory T, and then apply the above steps in segment $P_a P_f$. When there is a new anchor point, the above steps can be applied in segment $P_a P_{a+2}$ (i.e. $P_a P_f$), where P_a is the new anchor point and P_{a+2} is the new float point.



(a) Segment T_{af}

(b) $ASED(P_4) = sed1 + sed2 + \cdots + sed6$

(c) $ASED(P_6) = sed1 + sed2 + \cdots + sed6$

(d) P_6, P_8 become anchor point and float point

Fig. 4. An example of anchor point selecting.

The above process will be repeated until the entire trajectory T has been transformed into a piecewise linear approximation T'.

For compressing trajectories more accurate, ASED error measure is used for anchor point selection. Given anchor point P_a and float point P_f, the ASED of P_i $(f > i > a)$ is defined as

$$ASED(P_i) = \sum_{i=a}^{f} SEDError\big(T_a f, T'_a f, P_i\big), \tag{11}$$

where $T_a f = P_a, P_{a+1}, \ldots, P_f$ and $T'_a f = P_a, P_i, P_f$. By using ASED the most appropriate anchor point can be selected. For example, as shown in Fig. 4, $T_a f = P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9$, where P_1 is the anchor point and P_9 is the float point. For selecting a new anchor point between P_a and P_f, the ASED errors of $P_2, P_3, P_4, P_5, P_6, P_7, P_8$ are calculated. Due to P_6 has the minimum ASED error, P_6 and P_8 become new anchor point and new float point. And as seen in Fig. 4, the P_6 is the appropriate anchor point. Given a trajectory T of length n, HMOTC algorithm has $O(n^2)$ worst-case running time.

3.2. *Pseudo-Code of HMOTC Algorithm*

Figure 5 describes the details of HMOTC algorithm. First P_1 and P_3 are selected as anchor point P_a and float point P_f (lines 2 and 3). Then the algorithm will do some checks in segment $P_a P_f$ (line 7), as long as the anchor point P_a is not changed (i.e. index $= -1$), then move the float point P_f one point forward (line 13). Otherwise, P_{index} becomes the new anchor point and $P_{index} + 2$ becomes the new float point (lines 10 and 11). When the entire trajectory T has been transformed into a piecewise linear approximation T', the compressed trajectory T' will be returned (line 20).

Algorithm 1: HMOTC(T, μ, θ)

INPUT :
$T = \{P_1, P_2, \ldots, P_n\}$ // original trajectory
μ // SED threshold
θ //angel threshold
OUTPUT : $T' = \{P_{i1}, P_{i2}, \ldots, P_{im}\}$ //compressed trajectory, $P_{i1}=P_1$, $P_{im}=P_n$,m≤n

```
1.  T' = []
2.  anchorIndex = 1
3.  floatIndex = 3
4.  T' = T' Append P1
5.  while (floatIndex < = n)
6.  {
7.      index = findNewAnchorIndex(anchorIndex, floatIndex)
8.      if (index != -1) {          //new anchor point is founded
9.          T' = T' Append P_index
10.         anchorIndex = index
11.         floatIndex = anchorIndex + 2
12.     } else {            //anchor point not changed, move the float point
one point forward
13.         floatIndex += 1
14.     }
15. }
16. if (T' not contain Pn)
17. {
18.     T' = T' Append Pn
19. }
20. return T'
```

Fig. 5. HMOTC algorithm.

Figure 6 provides a detailed description of findNewAnchorIndex algorithm. This algorithm is used to find the new anchor point according to SED error and heading error, if there is no new anchor point found, then the value -1 will be returned (value -1 means that the segment $P_a P_f$ can represent the sub-trajectory $T_{sub} = P_a, P_a + 1, \ldots, P_f$ without too much loss of information). First, this algorithm calculates the heading change of predT(P_f) (line 3). Then, the algorithm checks whether predT(P_f) is a turning point (line 4), if predT(P_f) is a turning point then predT(P_f) becomes the new anchor point (line 5). After that, the algorithm will calculate three errors. (1) calculate each point's SED error (line 8). (2) calculate heading changes between segment $P_a P_f$ and each point (line 9). (3) calculate heading changes of any two points (line 12). If any errors are greater than the given parameters (line 13), then findNewAnchorIndex algorithm seeking the point which has the minimum ASED error (line 15) and return the point's index (line 16). When all points in segment $P_a P_f$ (expect P_f) are processed, value -1 will be returned (line 19).

Algorithm 2: findNewAnchorIndex(anchorIndex, floatIndex**)**

INPUT :
anchorIndex // anchor point's index
floatIndex // float point's index
OUTPUT : newAnchorIndex // index of new anchor point

1. newAnchorIndex = -1
2. segmentHeading = **GetHeading**($P_{anchorIndex}P_{floatIndex}$) //calculate heading of the segment
3. headingChange1 = **HeadingChange**(**GetHeading**($P_{floatIndex\text{-}1}$), **GetHeading**($P_{floatIndex\text{-}2}$))
4. **if**(headingChange1 > θ){ //Checking whether $pred_T(P_{floatIndex})$ is a turning point
5. **return** floatIndex - 1
6. }
7. **for** (i = anchorIndex **until** floatIndex) {
8. sed = **GetSED**(P_i, $P_{anchorIndex}$, $P_{floatIndex}$) //calculate P_i's SED
9. headingChange2 = **HeadingChange**(**GetHeading**(P_i), segmentHeading)
10. /* this is for calaulate the heading change of any two points in segment
11. $P_{anchorIndex}P_{floatIndex}$ (except point $P_{floatIndex,}$), as float point moving forward */
12. headingChange3 = **HeadingChange**(**GetHeading**($P_{floatIndex\ -\ 1}$), **GetHeading**(P_i))
13. **if** (headingChange2 > θ || headingChange3 > θ || sed > μ){
14. //find the point with the minimum ASED use formula 8, and return the point's index
15. newAnchorIndex = **findPointIndexWithMinimumASED(anchorIndex, floatIndex)**
16. **return** newAnchorIndex
17. }
18. }
19. **return** newAnchorIndex

Fig. 6. findNewAnchorIndex algorithm.

4. Evaluations

In order to verify the performance of the proposed algorithm, we implemented HMOTC algorithm and other algorithms by using Scala language. And Geolife dataset (Zheng *et al.*, 2009; Zheng *et al.*, 2008; Zheng *et al.*, 2010) was used for algorithms evaluating. The Microsoft Geolife dataset was obtained from 182 users over a period of five years (from April 2007 to August 2012), it contains 17,621 trajectories with a total distance of 1,292,951 kilometers and a total duration of 50,176 hours. 91.5 percent of the trajecto-

ries are logged in a dense representation, e.g. every 1–5 seconds or every 5–10 meters per point. This dataset can support transportation mode learning, 73 users have labelled their trajectories with transportation mode, such as driving, taking a bus, riding a bike and walking. In our evaluations, three labelled trajectories were used. Trajectory one is a multi-modal trajectory, it contains three transportation modes (walk, bus, train), 5911 positioning points, and a total duration of 3 hours 49 minutes (from 2008-06-18, 12:10:33 to 2008-06-18, 15:59:59). Trajectory two is a bus track, it contains 2045 positioning points, and a total duration of 50 minutes (from 2008-06-18, 12:33:34 to 2008-06-18, 13:23:02). Trajectory three is a taxi track in motorway, it contains 2167 positioning points, and a total duration of 37 minutes (from 2008-04-04, 07:16:50 to 2008-04-04, 07:53:00).

Our evaluation did not include algorithms which do not consider temporal information, because these algorithms will introduce larger temporal errors. And the direction-preserving algorithm DPTS-SP-Prac is also not included, because the algorithm lacks the control ability of the spatial error, the algorithm will introduce a larger spatial error. And the algorithms which use speed as a parameter are also not included, because it is hard to find an appropriate speed value as the parameter for a trajectory which contains multiple transportation modes. Various error metrics including average SED error, average spatial error, average speed error and average heading error were used in our evaluation. Given a trajectory $T = P_1, P_2, P_3, \ldots, P_{n-1}, P_n$ and its compressed representation $T' = P_{i1}, P_{i2}, P_{i3}, \ldots, P_{im-1}, P_{im}$, these error metrics are defined as follows:

$$AverageSpatialError\big(T, T'\big) = \frac{1}{n}\sum_{i=1}^{n} SpatialError\big(T, T', P_i\big), \tag{12}$$

$$AverageSEDError\big(T, T'\big) = \frac{1}{n}\sum_{i=1}^{n} SEDError\big(T, T', P_i\big), \tag{13}$$

$$AverageSpeedError\big(T, T'\big) = \frac{1}{n}\sum_{i=1}^{n} SpeedError\big(T, T', P_i\big), \tag{14}$$

$$AverageHeadingError\big(T, T'\big) = \frac{1}{n}\sum_{i=1}^{n} HeadingError\big(T, T', P_i\big). \tag{15}$$

4.1. *The Effect of Parameters on Compression Ratio*

Since the proposed algorithm is a heading oriented algorithm, this algorithm can accurately capture the direction information of the trajectory, and all turning points whose moving direction change degree are greater than the given angle threshold will be retained in compressed trajectory. So the compression rate of this algorithm is greatly influenced by trajectory's heading. The places where a user stayed, walked or took photos always cause a large heading change (because of the GPS positioning error, even if the user is stationary, it will produce a larger direction change). As shown in Fig. 7, due to the bus stopping at each bus station, the trajectory two has a lot of points whose direction change

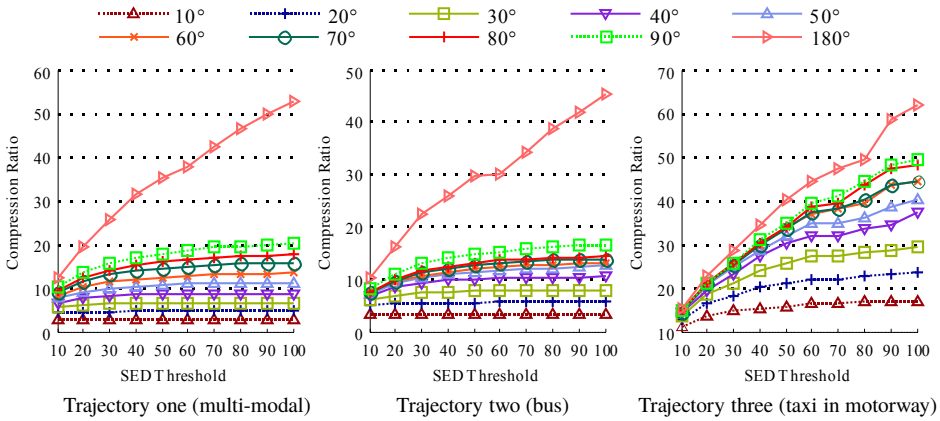


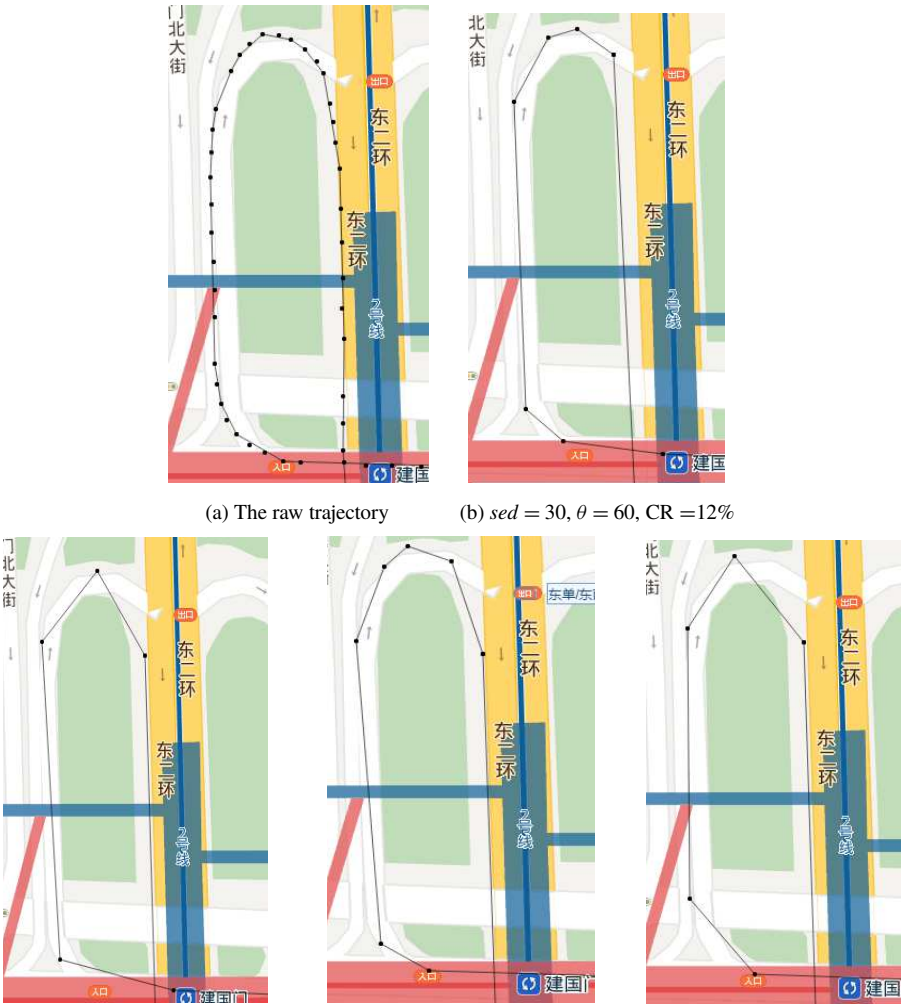
Fig. 7. The values of compression ratio under different parameters.

degree is greater than the given angle threshold at bus stations, thus the compression rate of trajectory two is mainly influenced by angle threshold. Similarly to trajectory two, the compression rate of trajectory one is also mainly influenced by angle threshold, because trajectory one contains both walk and bus transportation modes. Trajectory three is a taxi track in motorway, it contains few points whose direction change degree is greater than the given angle threshold, so the compression rate of trajectory three is influenced by both angle threshold and SED threshold.

When the zoom = 19 is shown in Fig. 8, the image effects under different conditions of the algorithm are used for compression. In order to facilitate the comparison, this paper intercepts a section of track with obvious effect. Fig. 8(a) is the original trajectory. In Fig. 8(b) the SED threshold value of 30 is taken and the angle threshold is 60°. It can be seen that keeping the contour works well and approaches the original trajectory. Figure 8(c) is a case where no angle threshold is set, and the trajectory compression rate is improved, but the profile is distorted, and there is a large difference to the original trajectory. In Fig. 8(d), the SED threshold is set to the maximum, which is the case considering only the angle threshold, and the effect is good. However, if the SED threshold is too large, trajectory distortion will be caused in other trajectory segments. Figure 8(e) is used to show that if you want to mention the compression ratio, you must increase the angle threshold and the effect of keeping the contour decreases.

4.2. *The Effect of Parameters on Compression Ratio*

The proposed algorithm has the benefit of taking into account heading information of the trajectory, it can have direction error under control. In the following evaluations, the angle threshold of the proposed algorithm was set to 60° for trajectory one, trajectory two and trajectory three. And compared to the traditional position-preserving compression algorithms which cannot control the direction error, from the experimental results, it can be seen that the proposed algorithms have smaller loss of heading information, while other algorithms will lose a lot of direction information.



(a) The raw trajectory (b) $sed = 30$, $\theta = 60$, CR =12%

(c) $sed = 30$, $\theta = 180$, CR =26% (d) $sed = 100$, $\theta = 60$, CR = 14% (e) $sed = 70$, $\theta = 90$, CR = 20%

Fig. 8. Comparison of trajectory compression in various situations.

Figures 9 and 10 contrast the trajectory compression algorithms in terms of average spatial error and average SED error. The experimental results of the two figures are basically the same, and they are introduced together. For trajectory one, the proposed algorithm outperforms other algorithms in terms of both average spatial error and average SED error. For trajectory two and trajectory three, the error of HMOTC is smaller than Before-OPW-TR and TD-TR, and slightly greater than SQUISH-E (μ).

Figure 11 provides a comparison in terms of average heading error. The result shows that the proposed algorithm is more accurate than other position-preserving compression algorithms. This is because HMOTC algorithm can compress trajectories while ensuring that heading error is within a user-specified bound. In contrast, other algorithms lack the

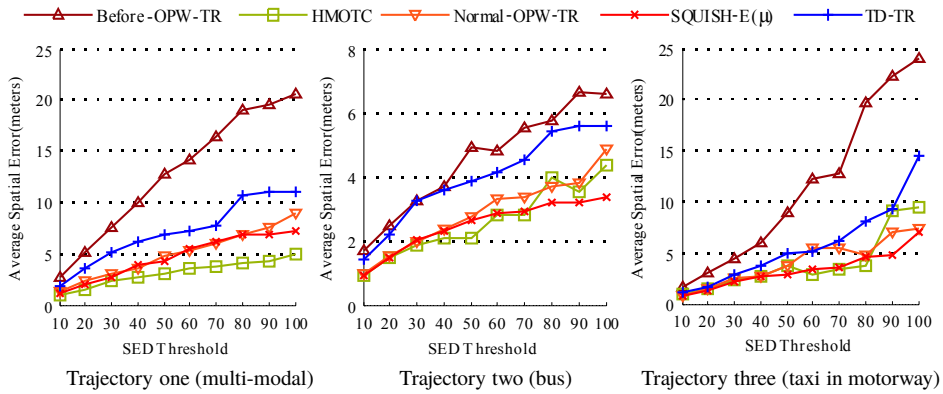


Fig. 9. Average spatial errors under different SED thresholds.

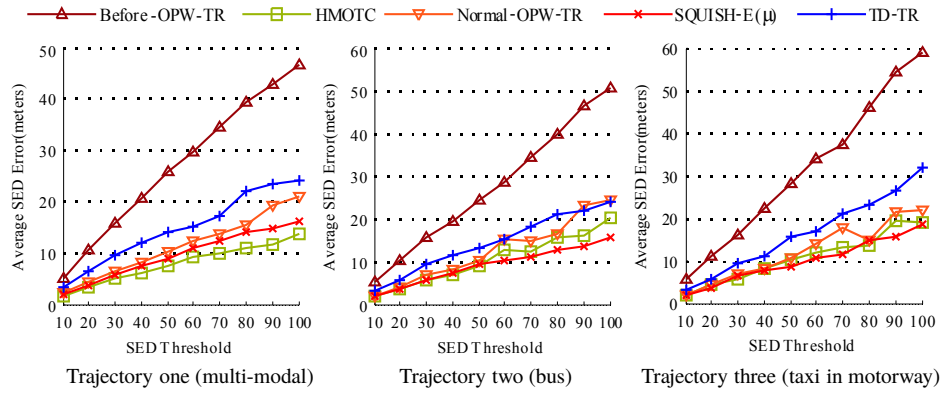


Fig. 10. Average SED errors under different SED thresholds.

capability of heading error control, so these algorithms will lose a lot of heading information.

Due to the proposed algorithm has the capability of heading error control. So, a lot of turning points are retained in compressed trajectory. Although these points are important in describing semantic meanings of a trajectory, they will cause a low compression ratio. As seen in Fig. 12, the compression ratio of HMOTC algorithm is lower than other algorithms. A high compression ratio can be achieved by setting a 180° angle threshold, if these turning points are not taken into account. In the next section, evaluations are given under the condition of same compression ratio.

4.3. *Evaluations Under the Condition of Same Compression Ratio*

In this section, we evaluated algorithms under the condition of same compression ratio. In order to achieve the given compression ratio, the value of HMOTC algorithm's angle threshold parameter may be set to obtuse angle or straight angle. In this case, HMOTC algorithm will lose the advantage of heading maintaining. From the experimental results,

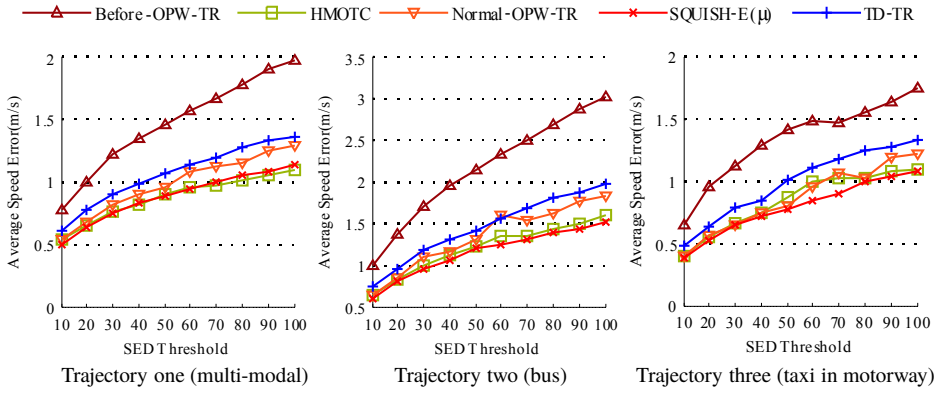


Fig. 11. Average speed errors under different SED thresholds.

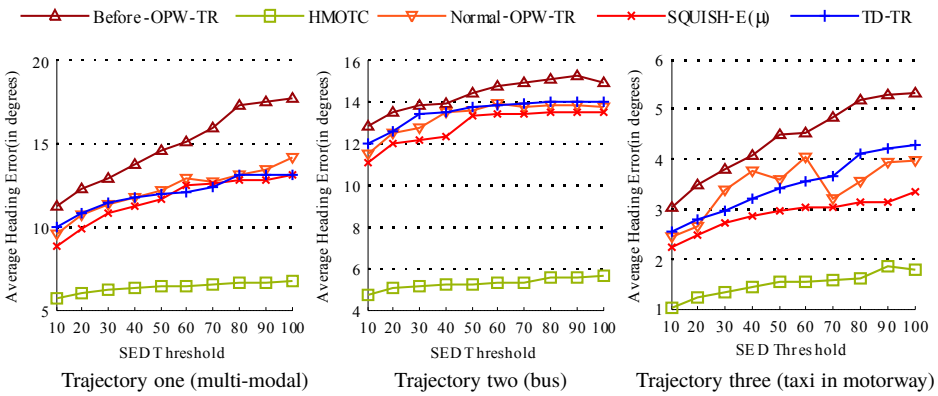


Fig. 12. Average heading errors under different SED thresholds.

it can be seen that even under the condition of the same compression ratio, information loss of the proposed algorithm is not too large. And in terms of average SED error, the proposed algorithm has a better performance than other algorithms.

In Fig. 13, algorithms are evaluated in terms of average spatial error. The average spatial error of the Normal-OPW-TR algorithm in the three trajectory modes is the largest, and the HMOTC algorithm is the smallest in the trajectory 1, but is equivalent to other algorithms in the trajectory 2 and the trajectory 3. The results show that Normal-OPW-TR algorithm introduced larger spatial error than other algorithms, and the spatial error of HMOTC algorithm is smaller than most algorithms.

In Fig. 14, average SED errors are shown for each algorithm over various compression ratios. The track 1 and track 2 can clearly show that the average SED error of the HMOTC algorithm is smaller than that of other algorithms, and the track 3 is not obvious, but it is equivalent to other algorithms. Experiments show that HMOTC and TD-TR are the most accurate algorithms in terms of SED error. Compared to TD-TR algorithm, HMOTC algorithm has the advantage of supporting both offline and online applications.

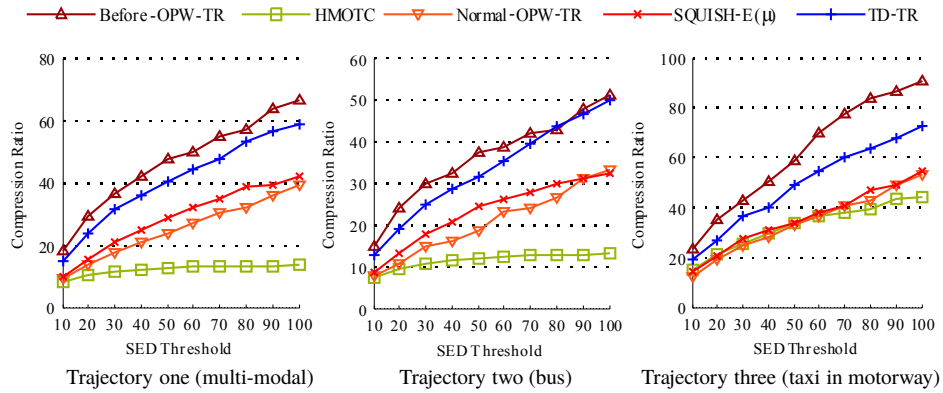


Fig. 13. Compression ratio under different SED thresholds.

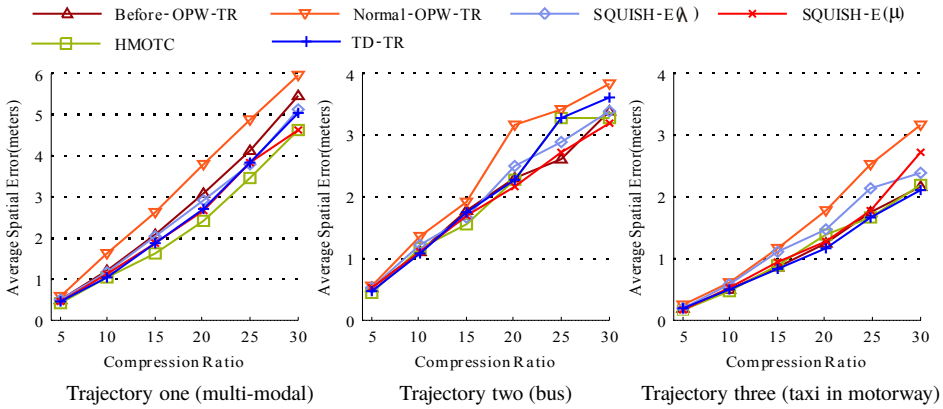


Fig. 14. Average spatial errors.

Figure 15 provides a comparison in terms of average speed error. The three models in the figure show that the average speed error of this algorithm is not very obvious, but the effect is basically the same, no uncontrollable or greater error loss occurs. As shown in the figure, SQUISH-E(μ), HMOTC and TD-TR are the most accurate algorithms in terms of speed error. Compared to HMOTC, SQUISH-E(μ) and TD-TR have the limitation that they only support offline applications.

Figure 16 shows the average heading errors for each algorithm over various compression ratios. The results show that with the increase of compression ratio, HMOTC algorithm gradually lost its advantage of heading maintaining. The reason behind this is that in order to achieve the given compression ratio, the value of HMOTC algorithm's angle threshold parameter is set to obtuse angle or straight angle. As seen in Fig. 16, the proposed algorithm has better performance in case of low compression ratio (compression ratio $CR \geqslant 15$).

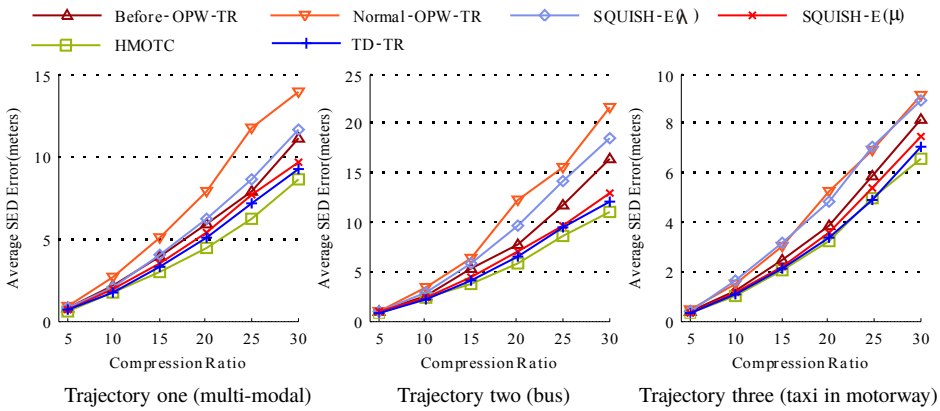


Fig. 15. Average SED errors.

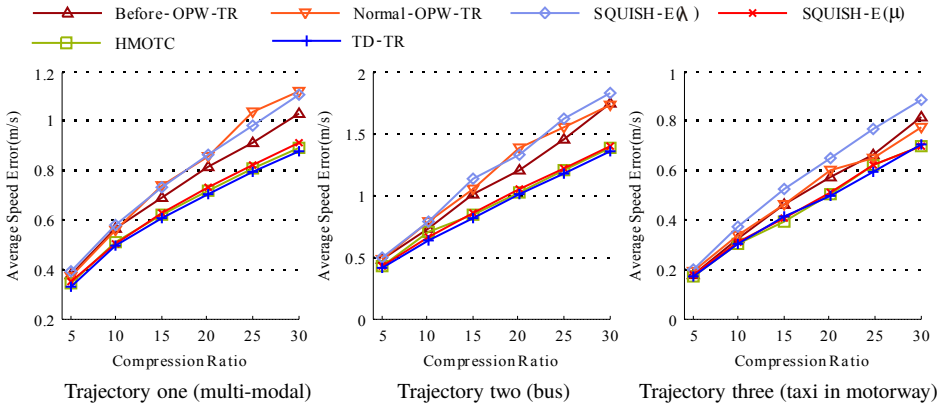


Fig. 16. Average speed errors.

5. Conclusions

In this paper, we proposed a heading maintaining oriented trajectory compression algorithm, called HMOTC. The algorithm can maintain both shape skeleton and semantic meanings of the trajectory, all turning points whose moving direction change degree is greater than the given angle threshold will be retained in compressed trajectory. Compared to traditional position-preserving compression algorithms, the HMOTC algorithm has the benefit of taking into account heading information of the trajectory, so this algorithm can support wider range of applications. The HMOTC has the flexibility of controlling compression with respect to both spatial error and direction error, so the compression process can be precisely controlled according to users' needs. If only the direction information of the trajectory is taken into account, the SED threshold of the algorithm can be set to positive infinity. If only the spatial information of the trajectory is taken into account, the angle threshold of the algorithm can be set to 180. Because the algorithm can compress

trajectory data while retrieving points from trajectory, it has the advantage of supporting both online and offline applications. The results show that the proposed algorithm can achieve more accurate approximation than other algorithms under the condition of same SED threshold, especially in the heading maintaining aspect. And it also has the good performance under the condition of same compression ratio.

Acknowledgement. This work is partially supported by Scientific Research Fund of Liaoning Provincial Education Department (No. 2017J049). The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

References

Cao, W., Li, Y. (2017). Dots: an online and near-optimal trajectory simplification algorithm. *Journal of Systems Software*, 126, 34–44.

Chen, Y., Jiang, K., Zheng, Y., Li, C., Yu, N. (2009). Trajectory simplification method for location-based social networking services. In: *Proceedings of the 2009 International Workshop on Location Based Social Networks*, Seattle, USA, pp. 33–40.

Chen, M., Xu, M., Franti, P. (2012). A fast $O(N)$ multiresolution polygonal approximation algorithm for GPS trajectory simplification. *IEEE Transactions on Image Processing*, 21(5), 2770–2785.

Douglas, D.H., Peucker, T.K. (1973). Algorithms for the reduction of the number of points required to represent a line or its caricature. *The International Journal for Geographic Information and Geovisualization*, 10(2), 112–122.

Giannotti, F., Nanni, M., Pinelli, F., Pedreschi, D. (2007). Trajectory pattern mining. In: *Proceedings of the 13th ACM SIGKDD international conference on Knowledge Discovery and Data Mining*, San Jose, USA, pp. 330–339.

Jiagao, W.U., Qian, K., Liu, M., Liu, L. (2015). Improvement of perez and vidal algorithm for the decomposition of digitized curves into line segments. *Journal of Computer Applications*, 35(5), 1209–1212.

Kolesnikov, A., Franti, P. (2002). A fast near-optimal min-# polygonal approximation of digitized curves. In: *IASTED Internatioal. Conference on Automation, Control and Information Technology (ACIT 2002)*, Novosibirsk, Russia, pp. 418–422.

Kolesnikov, A., Franti, P. (2003). Reduced-search dynamic programming for approximation of polygonal curves. *Pattern Recognition Letters*, 24(14), 2243–2254.

Li, T., Pei, T., Yuan, Y., Song C., Wang, W., Yang, G., (2014). A review on the classification, patterns and applied research of human mobility trajectory. *Progress in Geography*, 33(7), 938–948.

Long, C., Wong, R.C.-W., Jagadish, H.V. (2013). Direction-preserving trajectory simplification. *Proceedings of the VLDB Endowment*, 6(10), 949–960.

Meratnia, N., de By, R. (2004). Spatiotemporal compression techniques for moving point objects. In: *Advances in Database Technology – EDBT 2004, 9th International Conference on Extending Database Technology*, Heraklion, Crete, Greece, pp. 765–782.

Morzy, M., Rosikiewicz, L. (2007). Mining frequent trajectories of moving objects for location prediction. In: *Machine Learning and Data Mining in Pattern Recognition, 5th International Conference, MLDM 2007*, Leipzig, Germany, pp. 667–680.

Muckell, J., Hwang, J.-H., Patil,V., Lawson, C.T., Ping, F., Ravi, S.S. (2011). SQUISH: an online approach for GPS trajectory compression. In: *Proceedings of the 2nd International Conference and Exhibition on Computing for Geospatial Research & Application, COM.Geo 2011*, Washington, DC, USA, May 23–25.

Muckell, J., Olsen, P.W., Hwang, J.H., Lawson, C.T., Ravi, S.S. (2014). Compression of trajectory data: a comprehensive evaluation and new approach. *GeoInformatica*, 18(3), 435–460.

Perez, J.C., Vidal, E. (1994). Optimum polygonal approximation of digitized curves. *Pattem Recognition Letters*, 15(8), 743–750.

Potamias, M., Patroumpas, K., Sellis, T. (2006). Sampling trajectory streams with spatiotemporal criteria. In: *Proceedings of the 18th International Conference on Scientific and Statistical Database Management (SSDBM '06)*, Vienna, Austria, 3–5 July 2006, pp. 275–284.

Qian, H., Lu, Y. (2017). Simplifying gps trajectory data with enhanced spatial-temporal constraints. *International Journal of Geo-Information*, 6(11), 329.

Salotti, M. (2000). Improvement of Perez and Vidal algorithm for the decomposition of digitized curves into line segments. In: *Proceedings International Conference on Pattern Recognition ICPR'00*, Barcelona, Spain, pp. 882–886.

Salotti, M. (2001). An efficient algorithm for the optimal polygonal approximation of digitized curves. *Pattern Recognition Letters*, 22(2), 215–221.

Sun, P., Xia, S., Yuan, G., Li, D. (2016). An overview of moving object trajectory compression algorithms. *Mathematical Problems in Engineering*, 2016(3), 1–13.

Tian, S.L., Xu, J. (2011). GPS location data reducing based on turning point judgment method. In: *Second International Conference on Mechanic Automation and Control Engineering (MACE)*, Hohhot, China, pp. 7395–7398.

Yuan, J. (2012). *Querying, Mining with Applications on Large-Scale Trajectory Data*. HeFei University of Science and Technology of China.

Zheng, Y., Da-Ke, H.E., Zhang, W.F., et al. (2005). An efficient scheme for GPS data compression. *China Railway Science*, 26(3), 134–138.

Zheng, Y., Li, Q., Chen, Y., Xie, X., Ma, W.-Y. (2008). Understanding mobility based on GPS data. In: *Proceedings of the 10th ACM Conference on Ubiquitous Computing (Ubicomp 2008)*, Seoul, Korea, pp. 312–321.

Zheng, Y., Zhang, L., Xie, X., Ma, W.-Y. (2009). Mining interesting locations and travel sequences from GPS trajectories. In: *Proceedings of International Conference on World Wild Web (WWW 2009)*, Madrid, Spain. ACM Press, pp. 791–800.

Zheng, Y., Xie, X., Ma, W.-Y. (2010). GeoLife: a collaborative social networking service among user, location and trajectory. *Bulletin of the Technical Committee on Data Engineering*, 33(2), 32–40.

P. Hao, born in 1991, MS candidate. His research interests include data mining.

C. Yao, born in 1971, PhD, professor. His research interests include database theory and application, data mining, intelligent transportation.

Q. Meng, born in 1991, MS, professor. His research interests include data mining.

X. Yu, born in 1974, PhD, associate professor. His research interests include computer application.

X. LI, born in 1981, PhD, associate professor. Her research interests include machine learning.