J Inf Process Syst, Vol.13, No.3, pp.533~545, June 2017
https://doi.org/10.3745/JIPS.03.0073

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

Improvement of OPW-TR Algorithm for Compressing GPS Trajectory Data

Qingbin Meng*, Xiaoqiang Yu*, Chunlong Yao*, Xu Li*,
Peng Li*, and Xin Zhao*

Abstract

Massive volumes of GPS trajectory data bring challenges to storage and processing. These issues can be addressed by compression algorithm which can reduce the size of the trajectory data. A key requirement for GPS trajectory compression algorithm is to reduce the size of the trajectory data while minimizing the loss of information. Synchronized Euclidean distance (SED) as an important error measure is adopted by most of the existing algorithms. In order to further reduce the SED error, an improved algorithm for open window time ratio (OPW-TR) called local optimum open window time ratio (LO-OPW-TR) is proposed. In order to make SED error smaller, the anchor points are selected by calculating point's accumulated synchronized Euclidean distance (ASED). A variety of error metrics are used for the algorithm evaluation. The experimental results show that the errors of our algorithm are smaller than the existing algorithms in terms of SED and speed errors under the same compression ratio.

Keywords

ASED, GPS Trajectory, SED, Trajectory Compression

1. Introduction

In recent years, with the number of GPS-enabled devices growing rapidly, demand of location-based service shows a significant growth trend. Through location-based service, users can upload, visualize, browse and share those trajectories [1]. From other traveler's trajectories, people can find travel routes and hot areas that interest them. However, these devices generate massive volumes of GPS trajectory data. A calculation in [2] shows that if we collect data in every 10 second, without any compression, 100 MB of storage capacity is required to store the data for just 400 objects in a single day. This brings some problems in location-based service applications: (1) These GPS trajectory data will take up huge storage spaces. (2) These GPS trajectory data will cause network overhead. (3) It is difficult to find useful patterns from massive raw trajectory data [3,4]. (4) It will bring a heavy burden for a web browser while rendering these trajectories on the client side. It takes approximately 1 second to render just 1,000 points [5]. To solve these problems, the trajectory compression technique is given increasing concern.

According to the different categorization methods, trajectory compression can be classified as online

Manuscript received April 5, 2016; first revision December 15, 2016; accepted January 4, 2017.
Corresponding Author: Chunlong Yao (yaocl@dlpu.edu.cn)
* School of Information Science and Engineering, Dalian Polytechnic University, Dalian, China (mengqingbin@foxmail.com, {tigeryxq, yaocl}@dlpu.edu.cn, lixu102@aliyun.com, {lipeng, zhaoxin}@dlpu.edu.cn)

and offline compression or lossless and lossy compression. Online algorithm has the advantage of supporting real-time applications, which can compress trajectory data while retrieving points from trajectory. Offline algorithm begins to compress only after all points are obtained from the input trajectory. In general, offline algorithm has smaller errors than online algorithm. Lossless compression algorithm enables reconstruction the original data without information loss. Lossy compression will lose some information, and which cannot reconstruct the original data. Usually, the compression efficiency of lossless compression algorithm is not obvious. For example, the compression efficiency of lossless compression algorithm in which the local dictionary encoding and difference Huffman encoding are used, is 25% [6]. The main idea of lossy trajectory compression is to remove the redundant points. For example, a uniform linear motion trajectory can be represented by using just two points (first point and last point). The primary advantage of lossy compression is that it can significantly reduce the size of data while maintaining an acceptable error tolerance. Due to the advantage of lossy compression, this paper focuses on lossy compression of the trajectory data.

Douglas-Peuker (DP) algorithm [7] compresses trajectory data through recursive calculating the Euclidean distance from point to beeline to decide which points should be retained. Optimal algorithms [8-10] aim at minimizing Euclidean distance error, which can achieve a minimum error by removing points in searching process. Due to the computational overhead of the optimal algorithms, near-optimal algorithm is proposed. The algorithms proposed in [11,12] can achieve a faster search by reducing search space and using heuristic search. The paper [13] proposes an algorithm based on inflection point judgment method, in which the advantages and disadvantages of point by point judgment method and multi-point joint judgment method are analyzed. Trajectory simplification (TS) algorithm [14] uses heading change degree of the point and distance between this point and its most adjacent neighbors to weight importance of the point. The points with high weight will be retained in final compressed trajectory. These algorithms focus only on maintaining the shape of the trajectory, there is a lack of consideration of temporal information. However, the GPS trajectory contains both spatial and temporal information. A number of algorithms have been proposed with temporal information considered. Top-down time-ratio (TD-TR) [2] is an improvement algorithm of DP, which uses SED instead of Euclidean distance. Compared to DP, TD-TR has the benefit of taking temporal information into account. Open window time ratio (OPW-TR) [2] algorithm calculates each point's SED between the anchor point and the float point. If all the SED values are less than a given threshold, then the float point moves forward one point. Otherwise, a new anchor point will be chosen out. Based on the different anchor point selection strategies, OPW-TR has two modes called Before-OPW-TR and Normal-OPW-TR. Threshold-guided algorithm [15] compresses trajectory by constructing a safe area using moving object's speed and direction, if an incoming positioning point lies in the safe area, then this point contributes little information and can be discarded without significant loss in accuracy. Spatial QUalIty Simplification Heuristic – extended (SQUISH-E) algorithm [16] is an extended version of SQUISH [17], SQUISH-E compresses trajectory data by removing points of the lowest priority from the priority queue while ensuring that SED error is within a user-specified bound. According to parameters setting, SQUISH-E can be divided into SQUISH-E(μ) and SQUISH-E(λ).

Because online algorithm has the advantage of supporting real-time applications, an online trajectory compression algorithm is proposed in this paper. The algorithm compresses trajectory data by calculating point's accumulated synchronized Euclidean distance (ASED). In this algorithm, the redundant points will be discarded to reduce storage spaces and improve the efficiency of data processing and transmitting.

The remainder of this paper is organized as follows. In the next section, some definitions are given. In Section 3, our LO-OPW-TR (local optimum open window time ratio) algorithm is described in detail. In Section 4, some results with different error measurements are shown. Finally, paper conclusion and future work are discussed in Section 5.

2. Definitions

Definition 1. GPS trajectory: A GPS trajectory T of length n is a temporally ordered sequence of positioning points $\{P_1, P_2, P_3, ..., P_{n-1}, P_n\}$, each GPS point contains coordinates x, y and timestamp t.

Definition 2. Trajectory compression: Given a GPS trajectory $T = \{P_1, P_2, P_3, ..., P_{n-1}, P_n\}$, the intention of trajectory compression is to seek a set of temporally ordered positioning points T' (a subset of T), i.e., $T' = \{P_{i1}, P_{i2}, P_{i3}, ... , P_{i[m-1]}, P_{im}\}$, where $1= i1 < ... < im = n$.

Definition 3. Spatial error and Synchronized Euclidean Distance (SED) [16]: Given a trajectory T and its compressed representation T', the spatial error of T' with respect to 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 $\text{pred}_{T'}(P_i)$ and $\text{succ}_{T'}(P_i)$, where $\text{pred}_{T'}(P_i)$ and $\text{succ}_{T'}(P_i)$ denote P_i's closest predecessor and successor among the points in T'. The SED [15] of point P_i is also 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 that the value of x'_i and y'_i are calculated by using formulae (3) and (4), where $P_s(x_s, y_s, t_s)=\text{pred}_{T'}(P_i)$ and $P_e(x_e, y_e, t_e)=\text{succ}_{T'}(P_i)$.

For instance, in Fig. 1(a), the spatial errors of P_1, P_4, P_6 are zeros and the spatial error of P_2 is the perpendicular distance from P_2 to line $\overline{P_1P_4}$. In Fig. 1(b), the SED values of P_1, P_4, P_6 are zeros and the SED value of P_2 is the distance between P_2 and P'_2.

$$\Delta t_1 = t_e - t_s \tag{1}$$

$$\Delta t_2 = t_i - t_s \tag{2}$$

$$x'_i = x_s + (\Delta t_2 / \Delta t_1) * (x_e - x_s) \tag{3}$$

$$y'_i = y_s + (\Delta t_2 / \Delta t_1) * (y_e - y_s) \tag{4}$$

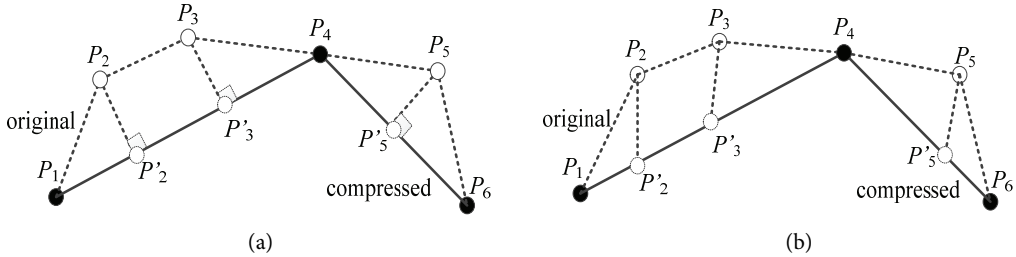


Fig. 1. Spatial error (a) and SED (b) are illustrated by using $T = \{P_1, P_2, P_3, P_4, P_5, P_6\}$ and $T' = \{P_1, P_4, P_6\}$.

Definition 4. Speed and heading: Given a trajectory $T = \{P_1, P_2, P_3, ..., P_{n-1}, P_n\}$, for any point P_i in the trajectory T (except the last point P_n), we can calculate P_i's heading and speed through formulae (5) and (6).

$$h_i = \overrightarrow{P_i P_{i+1}} \tag{5}$$

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

Definition 5. Compression ratio: Given a GPS trajectory $T = \{P_1, P_2, P_3, \ldots, P_{n-1}, P_n\}$ and its compressed representation $T' = \{P_{i1}, P_{i2}, P_{i3}, \ldots, P_{i[m-1]}, P_{im}\}$, the compression ratio CR is defined as follow.

$$CR = n/m \tag{7}$$

3. LO-OPW-TR Algorithm

Given a trajectory T and a SED threshold ε, OPW-TR algorithm starts by defining a segment between the first point P_1 (the anchor point) and the third point P_3 (the float point). Then the algorithm calculates all the SED values of the points between the anchor point and the float point. As long as all the SED values are smaller than ε, an attempt is made to move the float point one point forward in trajectory T. When the SED threshold ε is exceeded, two strategies can be applied: either the point causing the threshold violation becomes the new anchor point (Normal-OPW-TR), or the point just before it becomes the new anchor point (Before-OPW-TR).

The above process will be repeated until the entire trajectory has been transformed into a piecewise linear approximation. By modifying anchor point selection strategy, a new algorithm called LO-OPW-TR is proposed. This algorithm selects new anchor point by calculating point's ASED, the point with the minimum ASED value becomes the new anchor point. For each point P_i between the anchor point (P_a) and the float point (P_b), P_i's ASED value can be calculated by using formula (8).

$$ASED\ (P_i) = \sum_{j=a+1}^{i-1} SED\ (P_j) + \sum_{j=i+1}^{b-1} SED\ (P_j) \tag{8}$$

Fig. 2 gives an example of the anchor point selection, in which P_1 is the anchor point and P_9 is the float point. First, the SED values of P_2, P_3, P_4, P_5, P_6, P_7, P_8 are calculated, and the SED values of P_4, P_5, P_6 are exceed ε. Then, the ASED values 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 value, P_6 and P_8 become new anchor point and new float point, P_2, P_3, P_4, P_5 are removed from T. Fig. 2 shows that the most appropriate anchor point is selected by using new anchor point selection strategy.

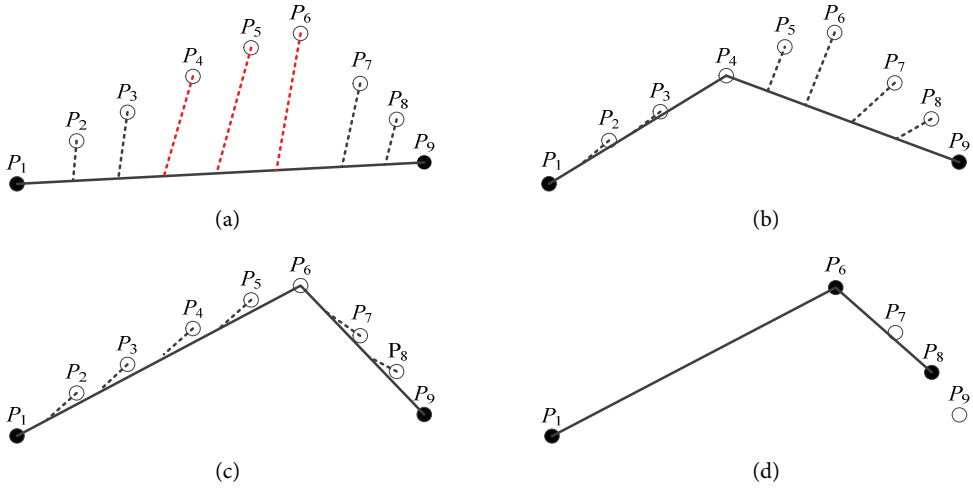


Fig. 2. An example of anchor point selection. (a) SEDs of P_4, P_5, P_6 are exceed ε, (b) calculate ASED of P_4, (c) calculate ASED of P_6, and (d) P_6, P_8 become anchor point and float point.

3.1 Pseudo-Code of LO-OPW-TR Algorithm

Algorithm 1 describes the details of LO-OPW-TR algorithm. At first P_1 and P_3 are selected as the anchor point and the float point (lines 2 and 3). Then the algorithm calculates all the SED values of the points between the anchor point and the float point (line 7). As long as all the SED values are smaller

Algorithm 1. LO-OPW-TR(T, ε)

```
INPUT :
  T = {P1, P2, ... , Pn}                    //original trajectory
  ε                                         //SED error tolerance
OUTPUT : Tout = {P'1, P'2, ... , P'm}       //compressed trajectory, P'1=P1,P'm=Pn ,m<n
1.  Tout = []
2.  anchorIndex = 1
3.  floatIndex = 3
4.  Tout = Tout Append P1
5.  while (floatIndex <= n)
6.  {
7.     index = findNewAnchorIndex(anchorIndex, floatIndex)
8.     if (index != -1) {                   //new anchor point and new float point are founded
9.        Tout = Tout Append Pindex
10.       anchorIndex = index
11.       floatIndex = anchorIndex + 2
12.    } else {                             //all SED values are smaller than ε
13.       floatIndex += 1
14.    }
15. }
16. if (Tout not contain Pn)
17. {
18.    Tout = Tout Append Pn
19. }
20. return Tout
```

than ε, an attempt is made to move the float point 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 all the points in T are processed, the compressed trajectory T_{out} will be returned (line 20).

Algorithm 2 provides the detailed description of findNewAnchorIndex algorithm. This algorithm calculates SED value of each point between the anchor point and the float point (line 3). If all the SED values are smaller than ε, -1 will be returned (line 10). Otherwise, the new anchor point's index will be returned (line 7).

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

```
INPUT :
   anchorIndex            //anchor point's index
   floatIndex             //float point's index
OUTPUT : newAnchorIndex   //index of the new anchor point
1.  newAnchorIndex = -1
2.  for (i = anchorIndex + 1 until floatIndex) {
3.     sed = GetSED(P_i, P_anchorIndex, P_floatIndex)  //calculate P_i's SED value
4.     if (sed > ε){
5.        //find the point with the minimum ASED (use formula (8), and return the point's index
6.        newAnchorIndex = findPointIndexWithMinimumASED(anchorIndex, floatIndex)
7.        return newAnchorIndex
8.     }
9.  }
10. return newAnchorIndex
```

Algorithm 3 provides a detailed description of findPointIndexWithMinimumASED algorithm. This algorithm calculates each point's ASED value between the anchor point and the float point, and returns the index of the point which has the minimum ASED value. Given the anchor point and the float point, P_i's ($anchorIndex<i<floatIndex$) ASED value can be calculated by using formula (8) (lines 4 to 10).

Algorithm 3. findPointIndexWithMinimumASED(anchorIndex, floatIndex**)**

```
INPUT :
   anchorIndex            //anchor point's index
   floatIndex             //float point's index
OUTPUT : pointIndexWithMinimumASED //index of the point which has the minimum ASED
1.   pointIndexWithMinimumASED  = -1
2.   minASEDError = Double.MaxValue
3.   for (i = anchorIndex + 1 until floatIndex) {
4.      ased_of_pi = 0
5.      for(j = anchorIndex + 1 until i){
6.         ased_of_pi = ased_of_pi + GetSED(P_j, P_anchorIndex, P_i)  //calculate P_j's SED value
7.      }
8.      for(j = i + 1 until floatIndex){
9.         ased_of_pi = ased_of_pi + GetSED(P_j, P_i, P_floatIndex)  //calculate P_j's SED value
10.     }
11.     if (ased_of_pi < minASEDError ){
12.        minASEDError  = ased_of_pi
13.        pointIndexWithMinimumASED  = i
14.     }
15.  }
16. return pointIndexWithMinimumASED
```

3.2 Time Complexity of LO-OPW-TR Algorithm

The proposed algorithm consists of three parts: LO-OPW-TR, findNewAnchorIndex and findPointIndexWithMinimumASED, where findNewAnchorIndex is a sub_function of LO-OPW-TR and findPointIndexWithMinimumASED is a sub_function of findNewAnchorIndex. First, we analyze the time complexity of findPointIndexWithMinimumASED. As shown in Algorithm 3, the findPointIndexWithMinimumASED function calculates the ASED values of all points between the anchor point $P_{anchorIndex}$ and the float point $P_{floatIndex}$, and for calculate each point's ASED value this function need to calculate SED value (*floatIndex* – *anchorIndex* – 2) times. So, the total time cost of findPointIndexWithMinimumASED is (*floatIndex* – *anchorIndex* – 1) × (*floatIndex* – *anchorIndex* – 2), where *floatIndex* is the index of the current float point and *anchorIndex* is the index of the current anchor point.

For findNewAnchorIndex function, if all SED values are smaller than the given threshold, the time cost of this function is (*floatIndex* – *anchorIndex* – 1). If only the SED value of $P_{anchorIndex-1}$ is greater than the given threshold, the time cost of this function is (*floatIndex* – *anchorIndex* – 1) + (*floatIndex* – *anchorIndex* – 1) × (*floatIndex* – *anchorIndex* – 2). So the worst-case running time of this function is (*floatIndex* – *anchorIndex* – 1) + (*floatIndex* – *anchorIndex* – 1) × (*floatIndex* – *anchorIndex* – 2), i.e. the function calculates SED values of all points between the anchor point $P_{anchorIndex}$ and the float point $P_{floatIndex}$, then calls the findPointIndexWithMinimumASED function.

Given a trajectory $T = \{P_1, P_2, P_3, ..., P_{n-1}, P_n\}$ and its compressed representation $T' = \{P_{i1}, P_{i2}, P_{i3}, ... , P_{i[m-1]}, P_{im}\}$ (1= $i1 < ... < im = n$), the number of anchor points founded by LO-OPW-TR algorithm is m-2 (except the first point P_1 and the last point P_n), and the maximum time to generate a single anchor point P_{ic} (1<c<m) is 1+2+3+ ... + (n-1-i[c-1]-1) + (n-i[c-1]-1)^2, i.e. the proposed algorithm calls the findNewAnchorIndex function in the order of findNewAnchorIndex(i[c-1], i[c-1]+2), findNew AnchorIndex(i[c-1], i[c-1]+3), findNewAnchorIndex(i[c-1], i[c-1]+4), ..., findNewAnchorIndex(i[c-1], n). The worst-case running time of proposed algorithm occurs if $P_{i1}=P_1$, $P_{i2}=P_2$, $P_{i3}=P_3$, ..., $P_{i[m-1]}=P_{m-1}$, $P_{im}=P_n$. In this situation, the maximum time costs of P_{i2}, P_{i3}, ..., $P_{i[m-1]}$ are 1+2+3+ ... + (n-1-1-1)+(n-1-1)^2, 1+2+3+ ... + (n-1-2-1)+(n-2-1)^2, ..., 1+2+3+ ... + (n-1-(m-2)-1)+(n-(m-2)-1)^2. We assume that the time cost of each point P_{ic} (1<c<m) is as long as P_{i2}, the upper bound time of proposed algorithm is thus 0.5×(m-2)(3n-7)(n-2). So the worst-case running time of proposed algorithm is $O(mn^2)$.

In practice, the float point seldom always float to the last point in T (i.e., P_n). We assume that for generate an single anchor point P_{ic} (1<c<m), the proposed algorithm limits the maximum floating times of the float point to d×L, where L is the average segment length (L=n/(m-1)) and d is a constant. Based on this assumption, the maximum time cost to generate a single anchor point is 1+2+3+ ... +d*L+(d*L+1)^2, the total time cost of proposed algorithm is 0.5×(m-2)(d×L+1)(3×d×L+2), i.e., $O(n^2/m)$. According to the paper [18], the segments generated by OPW algorithm are tightly clustered around the average length, so this limit has little effect in practice.

4. Evaluations

In order to verify the performance of the proposed algorithm, we used Scala language to implement LO-OPW-TR algorithm and other algorithms, and evaluated them by using GeoLife dataset [19-21].

The Microsoft GeoLife dataset was obtained from 182 users over a period of five years (from April 2007 to August 2012), 73 users have labeled their trajectories with transportation mode, such as driving, taking a bus, riding a bike and walking. GeoLife dataset contains 17,621 trajectories with a total distance of 1,292,951 km and a total duration of 50,176 hours. The 91.5% of the trajectories are logged in a dense representation, e.g., every 1–5 seconds or every 5–10 m per point. We selected three labeled trajectories with the different transportation modes (walk, multi-modal, train) to evaluate our algorithm. Trajectory one is labeled with walk, which contains 2,121 points, over a period of 1 hour 34 minutes (from 2011-09-03 04:42:02 to 2011-09-03 06:16:53). Trajectory two contains three transportation modes (walk, bus, train), 5,911 points, over a period of 3 hours 49 minutes (from 2008-06-18 12:10:33 to 2008-06-18 15:59:59). Trajectory three is labeled with train, which contains 2,621 points, over a period of 11 hours 24 minutes (from 2008-04-04 16:00:04 to 2008-04-05 03:24:30).

4.1 Evaluation Based on Error Metrics

Our evaluation dosen't include the most algorithms in which temporal information is not be considered. And the multiple error metrics are used in our evaluation, such as average SED error, average spatial error, average speed error and average heading error. Given a trajectory $T = \{P_1, P_2, P_3, ..., P_{n-1}, P_n\}$ and its compressed representation $T' = \{P_{i1}, P_{i2}, P_{i3}, ... , P_{i[m-1]}, P_{im}\}$, these error metrics are defined as follows:

$$AverageSpatialError(T,T') = \frac{1}{n}\sum_{i=1}^{n} SpatialError(P_i) \tag{9}$$

$$AverageSED\ Error(T,T') = \frac{1}{n}\sum_{i=1}^{n} SED(P_i) \tag{10}$$

$$AverageSpeedError(T,T') = \frac{1}{n}\sum_{i=1}^{n-1}\left| speed(P_i) - \frac{distance(\mathrm{pred}_{T'}(P_i), \mathrm{succ}_{T'}(P_i))}{\left|\Delta t(\mathrm{succ}_{T'}(P_i), \mathrm{pred}_{T'}(P_i))\right|}\right| \tag{11}$$

$$AcuteAngel\ (angel) = \textbf{if}\ (angel > 180)\ \textbf{return}\ 360 - angel\ \textbf{else return}\ angel \tag{12}$$

$$AverageHeadingError(T,T') = \frac{1}{n}\sum_{i=1}^{n-1} AcuteAngel\left(\left|heading(P_i) - \overrightarrow{\mathrm{pred}_{T'}(P_i)succ_{T'}(P_i)}\right|\right) \tag{13}$$

The trajectory compression algorithms are compared in terms of compression ratio in Fig. 3. The results show that Before-OPW-TR and TD-TR have higher compression ratio than LO-OPW-TR, but they also have larger SED errors (it is shown in Fig. 4). As seen in Fig. 3, under the same SED threshold, the compression effect of trajectory one (walk transportation mode) is the most obvious. Trajectory three (train transportation mode) has the minimum compression ratio compared with other trajectories. Thus, it can be concluded that the trajectory's sampling interval and speed have effect on compression ratio. When a trajectory (such as trajectory three) which has high speed and long sampling interval is compressed, a lower compression ratio will be received. On the contrary, when a trajectory (such as trajectory one) which has low speed and short sampling interval is compressed, a higher compression ratio will be received.

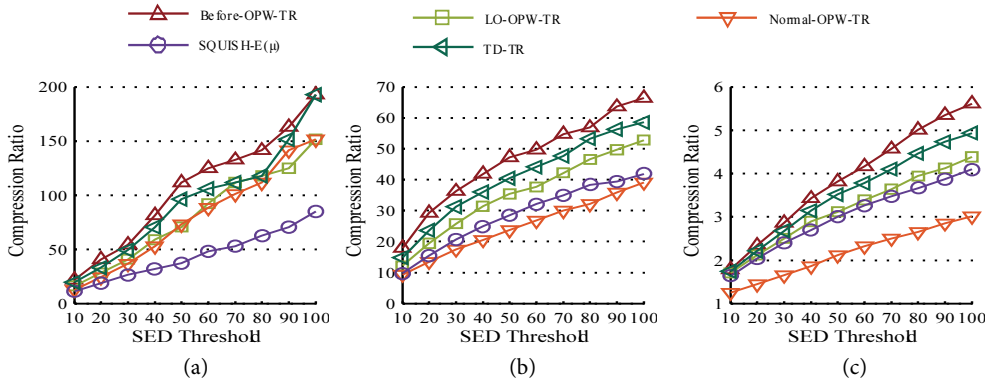


Fig. 3. The values of compression ratio under different SED thresholds. (a) Trajectory one, (b) trajectory two, and trajectory three.

Because the large errors will be introduced for trajectory three when the compression ratio is too high, the compression ratio was set from 5 to 15 for trajectory three in the following evaluation. In Fig. 4, we compared the performance of the algorithm in terms of average SED error. The results show that our algorithm outperform other algorithms with the smallest average SED error.

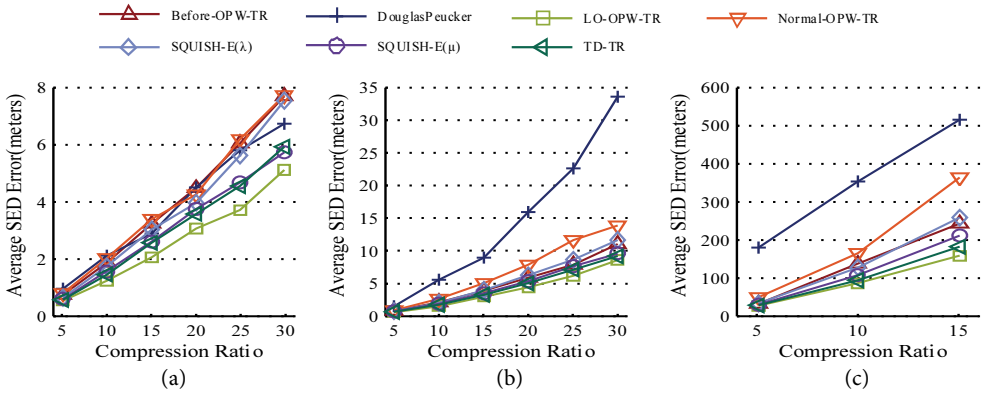


Fig. 4. Average SED errors. (a) Trajectory one, (b) trajectory two, and trajectory three.

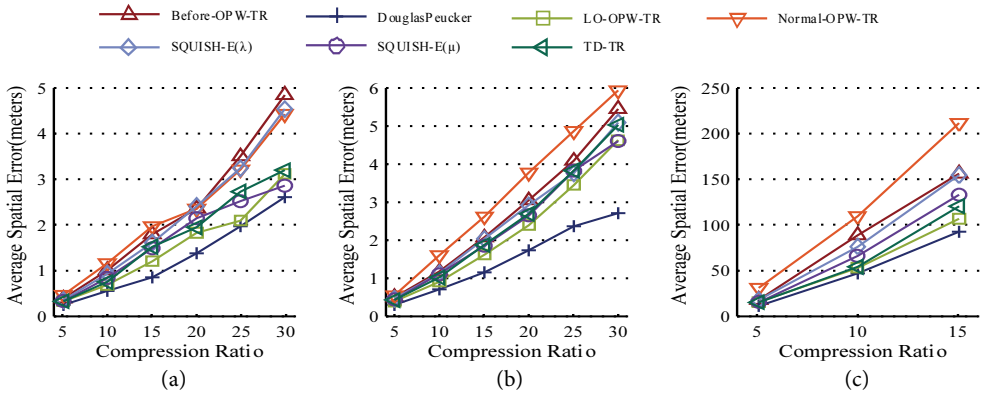


Fig. 5. Average spatial errors. (a) Trajectory one, (b) trajectory two, and trajectory three.

In Fig. 5, average spatial errors are evaluated under the different compression ratios. The results show that LO-OPW-TR and DP outperform other algorithms with smaller spatial error. DP algorithm is more accurate than LO-OPW-TR, however DP algorithm is not suitable for trajectory compression in which trajectory's temporal information is not be considered.

Fig. 6 shows the average speed errors of each algorithm over various compression ratios. TD-TR, SQUISH-E(μ) and LO-OPW-TR are most accurate algorithms in terms of speed error. LO-OPW-TR has the advantage of supporting both online and offline applications, while TD-TR and SQUISH-E(μ) only can be used in offline applications.

In Fig. 7, average heading errors of each algorithm over various compression ratios are shown. The best performance is exhibited by DP algorithm. The curve of our algorithm is in the middle.

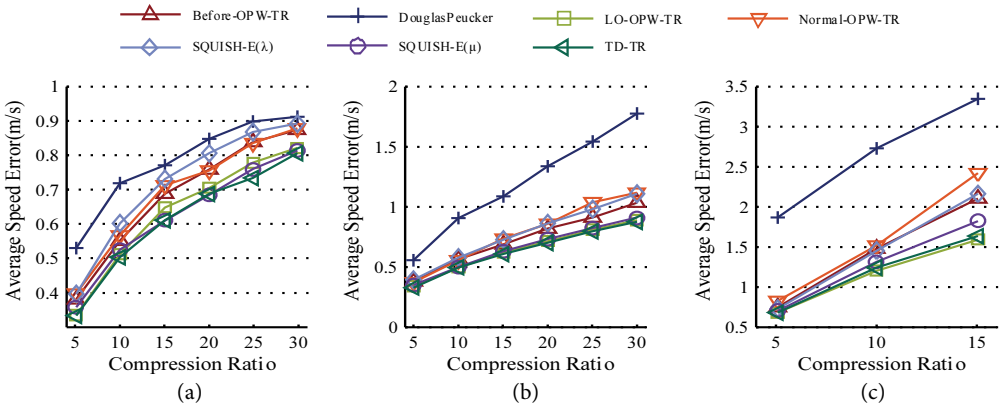


Fig. 6. Average speed errors. (a) Trajectory one, (b) trajectory two, and trajectory three.

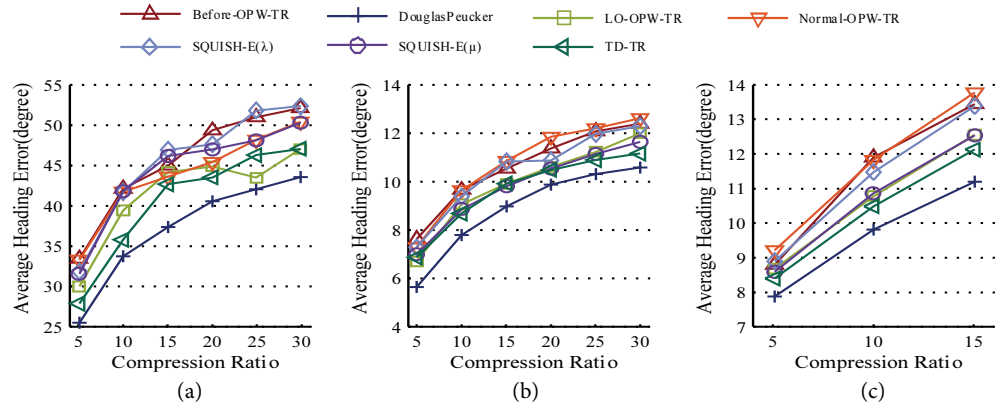


Fig. 7. Average heading errors. (a) Trajectory one, (b) trajectory two, and trajectory three.

5. Conclusion and Future Work

In this paper we have presented a new trajectory compression algorithm called LO-OPW-TR, in which a new anchor point selection strategy based on ASED is proposed. By using this anchor point

selection strategy, the most appropriate anchor point can be founded. The algorithm can compress trajectory data in process of receiving positioning points, so it has the advantage of supporting both online and offline applications. The experimental results show that the algorithm can achieve the most accurate compression in terms of SED error, and it also has the good performance in terms of speed error and spatial error. In the future, one possible approach for improving this algorithm would be use the heading information, which will further reduce the SED error and make heading error under control.

Acknowledgement

This work was supported by Dalian Municipal Science and Technology plan project of China (No. 2015A11GX011).

References

[1] Y. Zheng, Y. Chen, X. Xie, and W. Y. Ma, "GeoLife 2.0: a location-based social networking service," in *Proceedings of 10th International Conference on Mobile Data Management: Systems, Services and Middleware*, Taipei, Taiwan, 2009, pp. 357-358.

[2] N. Meratnia and A. Rolf, "Spatiotemporal compression techniques for moving point objects," in *Proceedings of the 9th International Conference on Extending Database Technology (EDBT)*, Crete, Greece, 2004, pp. 765-782.

[3] M. Morzy, "Mining frequent trajectories of moving objects for location prediction," in *Proceedings of 5th International Conference on Machine Learning and Data Mining in Pattern Recognition*, Leipzig, Germany, 2007, pp. 667-680.

[4] F. Giannotti, M. Nanni, F. Pinelli, and D. Pedreschi, "Trajectory pattern mining," in *Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, San Jose, CA, 2007, pp. 330-339.

[5] M. Chen, M. Xu, and P. Franti, "A fast $O(N)$ multiresolution polygonal approximation algorithm for GPS trajectory simplification," *IEEE Transactions on Image Processing*, vol. 21, no. 5, pp. 2770-2785, 2012.

[6] Y. Zheng, D. K. He, W. F. Zhang, and X. H. Lu, "Efficient scheme for GPS data compression," *Zhongguo Tiedao Kexue (China Railway Science)*, vol. 26, no. 3, pp. 134-138, 2005.

[7] D. H. Douglas and T. 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*, vol. 10, no. 2, pp. 112-122, 1973.

[8] J. C. Perez and E. Vidal, "Optimum polygonal approximation of digitized curves," *Pattern Recognition Letters*, vol. 15, no. 8, pp. 743-750, 1994.

[9] M. Salotti, "Improvement of Perez and Vidal algorithm for the decomposition of digitized curves into line segments," in *Proceedings of 15th International Conference on Pattern Recognition*, Barcelona, Spain, 2000, pp. 878-882.

[10] M. Salotti, "An efficient algorithm for the optimal polygonal approximation of digitized curves," *Pattern Recognition Letters*, vol. 22, no. 2, pp. 215-221, 2001.

[11] A. Kolesnikov and P. Franti, "A fast near-optimal min-# polygonal approximation of digitized curves," in *Proceedings of International Conference on Automation, Control and Information Technology (ACIT'02)*, Novosibirsk, Russia, 2002, pp. 418-422.

[12] A. Kolesnikov and P. Franti, "Reduced-search dynamic programming for approximation of polygonal curves," *Pattern Recognition Letters*, vol. 24, no. 14, pp. 2243-2254, 2003.

[13] Y. Xuan and J. Xu, "GPS location data reducing based on turning point judgment method," in *Proceedings of 2011 Second International Conference on Mechanic Automation and Control Engineering (MACE),* Hohhot, China, 2011, pp. 7395-7398.

[14] Y. Chen, K. Jiang, Y. Zheng, C. Li, and N. Yu, "Trajectory simplification method for location-based social networking services," in *Proceedings of the 2009 International Workshop on Location Based Social Networks*, Seattle, WA, 2009, pp. 33-40.

[15] M. Potamias, K. Patroumpas, and T. Sellis, "Sampling trajectory streams with spatiotemporal criteria," in *Proceedings of 18th International Conference on Scientific and Statistical Database Management,* Vienna, Austria, 2006, pp. 275-284.

[16] J. Muckell, P. W. Olsen, J. H. Hwang, C. T. Lawson, and S. S. Ravi, "Compression of trajectory data: a comprehensive evaluation and new approach," *GeoInformatica*, vol. 18, no. 3, pp. 435-460, 2014.

[17] J. Muckell, J. H. Hwang, V. Patil, C. T. Lawson, F. Ping, and S. S. Ravi, "SQUISH: an online approach for GPS trajectory compression," in *Proceedings of the 2nd International Conference on Computing for Geospatial Research & Applications*, Washington, DC, 2011.

[18] E. Keogh, S. Chu, D. Hart, and M. Pazzani, "An online algorithm for segmenting time series," in *Proceedings IEEE International Conference on Data Mining*, Washington, DC, 2001, pp. 289-296.

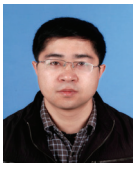
[19] Y. Zheng, L. Zhang, X. Xie, and W. Y. Ma, "Mining interesting locations and travel sequences from GPS trajectories," in *Proceedings of the 18th International Conference on World Wide Web*, Madrid, Spain, 2009, pp. 791-800.

[20] Y. Zheng, Q. Li, Y. Chen, X. Xie, and W. Y. Ma, "Understanding mobility based on GPS data," in *Proceedings of the 10th International Conference on Ubiquitous Computing*, Seoul, Korea, 2008, pp. 312-321.

[21] Y. Zheng, X. Xie, and W. Y. Ma, "GeoLife: a collaborative social networking service among user, location and trajectory," *IEEE Data Engineering Bulletin*, vol. 33, no. 2, pp. 32-39, 2010.

Qingbin Meng

He was born in Liaoning province, China, in 1991. He received his B.E. degree in computer science and technology from Dalian Polytechnic University, Dalian, China, in 2014. He is currently pursuing his M.E. degree in control science and engineering at Dalian Polytechnic University, Dalian, China. His current research interests include spatio-temporal data mining and machine learning.



Xiaoqiang Yu

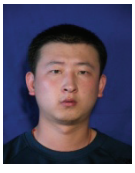
He was born in Shandong province, China, in 1974. He received his B.E. degree in computer science and technology from Northeast Normal University, China, in 1997, and his M.E. degree in computer application technology from Dalian Maritime University, China, in 2004. He is currently an associate professor and supervisor of postgraduate students at Dalian Polytechnic University, Dalian, China. His current research interests include enterprise information system, genetic algorithm and data mining, etc.

Chunlong Yao

He was born in Heilongjiang province, China, in 1971. He received his B.E. degree in computer and its application from Northeast Heavy Machinery Institute, Qiqihar, China, in 1994; his M.E. degree in computer application technology from Northeast Heavy Machinery Institute, Qiqihar, China, in 1997; and his Ph.D. degree in computer software and theory from Harbin Institute of Technology, Harbin, China, in 2005. He is currently a professor and supervisor of postgraduate students at Dalian Polytechnic University, Dalian, China. His current research interests include database and data mining, and intelligent information system.

Xu Li

She was born in Jilin province, China, in 1980. She received her B.E. degree in computer science and technology from University of science and technology Anshan, China, in 2003, and her M.E. and Ph.D. degrees in computer application technology from Yanshan University, China, in 2006 and 2010, respectively. She is currently an associate professor and supervisor of postgraduate students at Dalian Polytechnic University, Dalian, China. Her current research interests include natural language processing, machine learning and data mining.



Peng Li

He was born in Liaoning province, China, in 1979. He received his B.E. degree in communication engineering from Harbin Institute of Technology, China, in 2002; his M.E. degree in communication and information system from Harbin Institute of Technology, China, in 2004; and his Ph.D. degree in information and communication engineering from Harbin Institute of Technology, China, in 2009. He is currently an associate professor at Dalian Polytechnic University, Dalian, China. His research interests include multi-hop networks and Internet of things.

Xin Zhao

He was born in Liaoning province, China, in 1968. He received his B.E. degree in applied physics from Jilin University, China, in 1991; and his M.E. degree in condensed matter physics from Jilin University, China, in 1997. He is currently a professor at Dalian Polytechnic University, Dalian, China. His research interests include optoelectronic materials and devices, and Internet of things.