Improvement of TD-TR Algorithm for Simplifying GPS Trajectory Data

Kanasuan Hansuddhisuntorn
School of Information, Computer and Communication Technology
Sirindhorn International Institute of Technology
Pathum Thani, Thailand
boss.kanasuan@gmail.com

Teerayut Horanont
School of Information, Computer and Communication Technology
Sirindhorn International Institute of Technology
Pathum Thani, Thailand
teerayut@siit.tu.ac.th

***Abstract*—Over the past decades, massive amounts of GPS trajectories have been obtained with the development of low-cost GPS enabled devices, capturing users' spatial and temporal information. The massive increase in trajectory data generates high storage and data processing burdens. Several of trajectory simplification algorithms have been proposed to overcome these difficulties. A key requirement in trajectories simplification is to minimize information loss while preserving the quality of the information. To further reduce the compression time, an improved algorithm for top-down time-ratio (TD-TR) called top-down time-ratio Reduce (TD-TR Reduce) is proposed. The algorithms were evaluated using several parameters, such as compression times and errors arising from trajectory data simplifications in the *Geolife* trajectory data set. The results of the simulation show that TD-TR Reduce can achieve an attractive trade-off between the compression rate and the simplification error with up to 33% lower compression time.**

***Index Terms*—GPS, Trajectory Data, Trajectories Simplification**

I. Introduction

Over the past decade, the number of GPS-enabled devices has increased significantly [1]. Due to the increasing number of GPS-enabled devices, such as mobile phones or in-car navigation systems, the volume of spatial and temporal information recording the footprint of a moving device has increased dramatically. The massive amounts of trajectory data could easily exceed the existing available data storage, which leads to three major challenges: storing, transmitting and visualizing the data. For example, a calculation due to Meratnia and de By [2] shows that without any data compression, storing a trajectory data of 400 objects per day at an interval of 10 seconds requires a storage capacity of 100 Mb.

While dealing with these massive amounts of trajectory data, an effective compression mechanism is one of the most key components of the storage layer. Many trajectory compression types have been introduced in the past. This can be divided into various methods of categorization, lossless compression and lossy compression or online compression and offline compression. The advantage of online compression is that it supports real-time applications, which can compress trajectory data while picking up new trajectory points. Only after all points are obtained from the input trajectory, offline algorithms soon begin to compress. However, offline compression usually has smaller errors compare to online compression. Lossless compression enables the original data to be reconstructed without loss of information, while Lossy compression is not possible. The main advantage of lossy compression is that it can significantly reduce the trajectory size while maintaining reasonable error tolerances.

This paper introduces an offline lossy trajectory simplification algorithm, by efficiently utilizing the feature extraction points and skip threshold, which results in a shorter compression time against the current state of the art compression algorithms.

The rest of this paper is arranged according to the following: Section II summarizes related work. Section III presents the metrics for evaluating the simplification algorithm. Section IV explains the feature point extraction procedure. Section V presents the proposed trajectory algorithm in detail. Section VI discusses our evaluation results. Section VII describes our conclusion and future work.

II. Related Work

Several algorithms have been proposed to simplify trajectory data, different algorithms use different approaches to find a similar trajectory with fewer points. As for trajectory simplification, the current state of the art algorithm had been studied by zhang et al. [3] for both online and offline compression mode, providing a comprehensive evaluation of 25 trajectory simplification algorithms on 5 different data sets.

For offline compression mode, The well-known trajectory The Douglas-Peuker (DP) algorithm [4], compresses trajectory data by recursively divides the trajectory to decide which points should be retained according to user-defined perpendicular Euclidean distance (PED) threshold. Top-down time-ratio (TD-TR) [5] is an extension of the Douglas-Peucker, which uses Spatial Euclidean distance (SED) instead of PED. MRPA algorithm [6] is proposed to compress trajectories in O(N) computational time, where a new error metric called an integral square synchronous Euclidean distance (ISSD) was introduced.

For online compression mode, the Spatial QUalIty Simplification Heuristic Algorithm (SQUISH) [7] used a priority queue data structure. It compresses each trajectory by removing points from the priority queue that has the lowest priority until the desired compression ratio is achieved. Later, Spatial QUalIty Simplification Heuristic-Extended (SQUISH-E) was developed to ensure that the SED error is within a user-specific bound.

III. Metrics

This section describes the metrics for evaluating the simplification algorithm. In this study, the original trajectory T of length n is represented as a temporally ordered sequence of points $\{P_1, ..., P_n\}$, where each point $P_n(x_n, y_n, t_n)$ contains longitude x, latitude y and timestamp t

After the simplification, the simplified trajectory can be express as $T' = \{P_{s_1}, ..., P_{s_m}\}$ and $P_{s_n}(x_{s_n}, y_{s_n}, t_{s_n})$ where $m \leq n$ and $1 = s_1 < ... < s_m = n$

This section provides a comprehensive survey of both error metrics (Chapter III-A) and performance metrics (Section III-B) and a detailed discussion of these metrics (Section III-C).

A. Error Metrics

1) *Synchronized Euclidean Distance* (SED): The distance between the actual points P_k and its synchronized point $P'_k(x'_k, y'_k, t'_k)$ created by two points P_s and P_e at identical time stamps (see Figure 1) and can be calculated as follows:

$$SED(P_k) = \sqrt{(x_k - x'_k)^2 + (y_k - y'_k)^2}$$

where

$$x'_k = x_s + \frac{x_e - x_s}{t_e - t_s}(t_k - t_s)$$

$$y'_k = y_s + \frac{y_e - y_s}{t_e - t_s}(t_k - t_s)$$

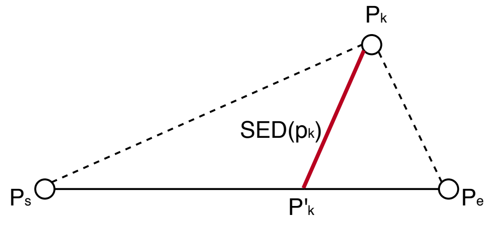

Fig. 1. Synchronized Euclidean Distance (SED)

2) *Trajectory Distance Reduction Ratio* (TDRR): Trajectory distance reduction ratio is the accumulated travel distance ratio of the simplified trajectory T' versus its original trajectory T and can be calculated as follows:

$$TDRR(T, T') = 1 - \frac{TDD(T)}{TDD(T')}$$

where

$$TDD(T_i) = \sum_{i=1}^{|T|-1} DISTANCE(P_i, P_{i+1})$$

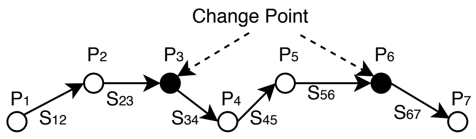

Fig. 2. Two speed change points

B. Performance Metrics

1) *Compression Ratio* (CR): Compression ratio is defined as the size of the simplified trajectory T' versus its original trajectory T and can be calculated as follows:

$$CR(T, T') = 1 - \frac{|T|}{|T'|}$$

2) *Compression Time*: Compression time is the amount of time taken for a trajectory to be simplified.

C. Discussion

The trajectory-simplifying algorithm's efficiency is defined as the combination of error metrics and performance metrics. For the further improvement of error metrics, In order to measure the average time-synchronized euclidean distance between the original trajectory T and its simplified trajectory T', we introduce Average Synchronized Euclidean distances (ASED). The ideal simplified trajectory should be highly compressed with minimum compression time, TDRR and ASED. (see Figure 3)

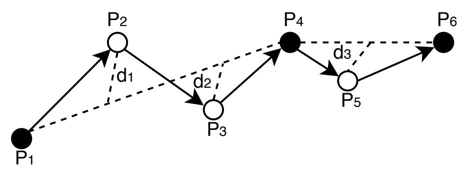

Fig. 3. Example of calculating ASED. Given Trajectory $T = \{P_1, P_2, P_3, P_4, P_5, P_6\}$ and simplified trajectory $T' = \{P_1, P_4, P_6\}$ ASED is calculated as $(d_1 + d_2 + d_3)/2$

IV. Feature Point Extraction

Some GPS tracking point is redundant in some applications. To retain only the important part where some events occur (e.g. travel mode transition), we proposed a feature point extraction model with a focus reducing the trajectory data based on several of the movement characteristics given as follows.

1) *Movement Speed*: In the transition mode, node movement speeds typically change significantly [8], this includes walking, cycling, driving vehicles or taking the train. As shown in Figure 4, The feature node is the place where the node changes transportation mode from driving to walking.
2) *Heading*: The heading changes accordingly to the current modes of transport. For example, when the mode of transport is walking rather than another mode of transport, the heading will change very often.

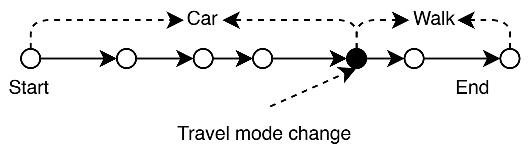

Fig. 4. Transition of transportation mode

In order to obtain the above-mentioned node points, we propose a process for the extraction of the feature points. To detect a feature point caused by switching between transportation modes. First, we specify the difference of movement speed as $\triangle_{i+1} = |SP_{i+1} - SP_i|$, in which SP_i is an average speed in the line segment $\overline{P_{i,i+1}}$. When $\triangle_{i+1}$ exceeds a certain threshold, P_{i+1} is kept as a feature point. Because speed changes are usually caused by switches between travel modes (see Figure 4). We propose the usage of the standard deviation of movement speeds as the speed threshold since the standard deviation can indicate changes in velocity. The speed threshold SP_{th} is expressed as

$$SP_{th} = \sqrt{\frac{\sum_{i=1}^{n-1}(SP_i - \overline{SP})^2}{n-1}}$$

where

$$\overline{SP} = \sum_{i=1}^{n-1} \frac{SP_i}{n-1}$$

To detect a feature point caused by a significant change in the node heading. δ_{i+1} was defined as the heading different between θ_i and θ_{i+1} and can be calculated as follows:

$$\delta_{i+1} = \begin{cases} 360 - |\theta_{i+1} - \theta_i|, & if\ |\theta_{i+1} - \theta_i| > 180 \\ |\theta_{i+1} - \theta_i|, & \text{otherwise} \end{cases}$$

where θ_i represents the current heading on the line segment $\overline{P_{i,i+1}}$, that can be calculated using The haversine formula. If δ_{i+1} exceeds a certain threshold δ_{th}, P_{i+1} is kept as a feature point.

When the node travels in a straight line at a constant speed (e.g. highway, tollway), the entire point will be ignored from the two feature point extraction method mentioned above. In order to prevent the above scenario from occurring, we introduce the skipping threshold $skip_{th}$. When the number of points that are ignored from the feature point extraction method above reaches the skipping threshold, the point is kept as a feature point.

V. TD-TR Reduce algorithm

In order to reduce the compression time of the current TD-TR algorithm (see Figure 5), we propose a TD-TR Reduce algorithm to simplify the trajectory by performing a traditional TD-TR algorithm on a set of extracted feature points. The following procedure is provided in algorithm 1 below:

1) Add the first from T to the feature point array $T_{feature}$ (line 3)
2) Calculate the standard deviation of speed in T and set it as the speed threshold SP_{th} (line 4)
3) Starting from $i = 1$, iteratively add point p_{i+1} to the feature point array $T_{feature}$ and set $skip$ parameter back to zero if the movement speed different $\triangle_{i+1}$ is greater than or equal to speed threshold SP_{th} or the heading different δ_{i+1} is greater than or equal to heading threshold δ_{th} (lines 5-7)
4) If the n point was not added to the feature point array $T_{feature}$, increase $skip$ value by one (line 10)
5) If the the $skip$ value reaches the certain skip threshold $skip_{th}$, add point p_{i+1} to the feature point array $T_{feature}$ and set $skip$ parameter back to zero (lines 12-15)
6) Add the last points in T to the feature point array $T_{feature}$ (line 18)
7) Perform TD-TR algorithm on feature points array $T_{feature}$ and stored it as a simplified trajectory T'. (line 19)
8) Finally, the simplified trajectory T' is returned. (line 20)

Algorithm 1 TD-TR Reduce algorithm

```
Input: $T = \{p_1,...,p_n\}$, heading threshold $\delta_{th}$,
    error threshold $\varepsilon$, skip threshold $skip_{th}$
Output: Simplified Trajectory $T'$
1: $T_{feature} = []$
2: $skip = 0$, $i = 1$
3: $T_{feature} = T_{feature}$ APPEND $p_1$
4: $SP_{th} \leftarrow$ SD of speed in $T$
5: while $i < n - 1$ do
6:     if $| \triangle_{i+1} | \geq SP_{th}$ or $\delta_{i+1} \geq \delta_{th}$ then
7:         $T_{feature} = T_{feature}$ APPEND $p_{i+1}$
8:         $skip = 0$
9:     else
10:        $skip = skip + 1$
11:    end if
12:    if $skip == skip_{th}$ then
13:        $T_{feature} = T_{feature}$ APPEND $p_{i+1}$
14:        $skip = 0$
15:    end if
16:    $i = i + 1$
17: end while
18: $T_{feature} = T_{feature}$ APPEND $p_n$
19: $T' = TDTR(T_{feature}, \varepsilon)$
20: return $T'$
```

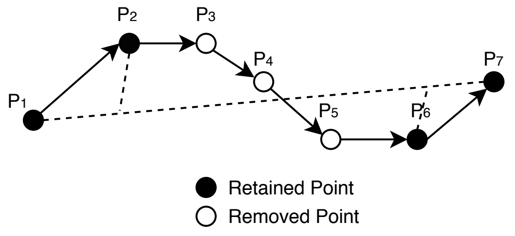

Fig. 5. TD-TR Algorithm

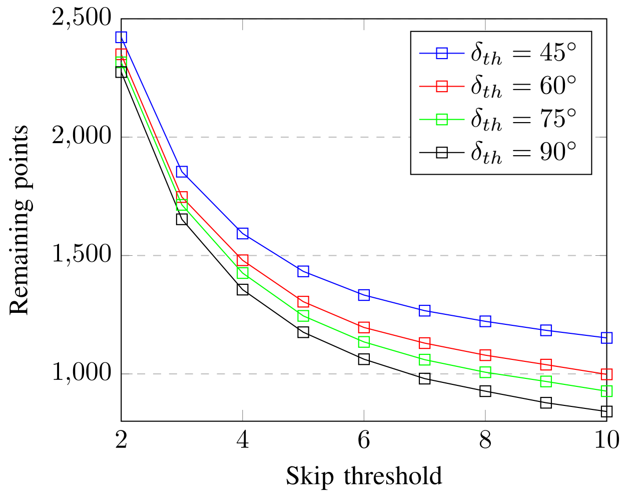

Fig. 6. Trajectory one

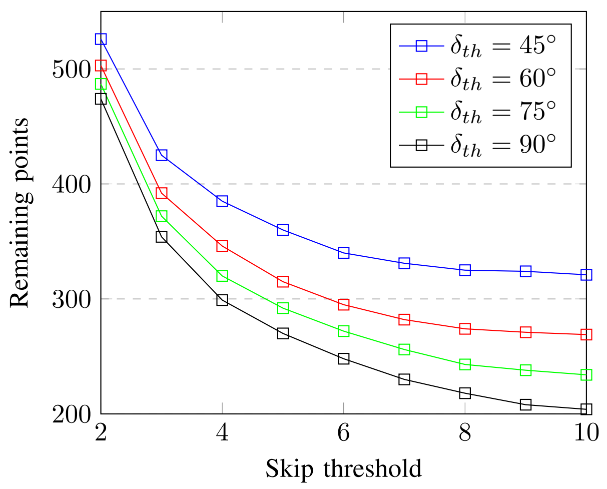

Fig. 7. Trajectory three

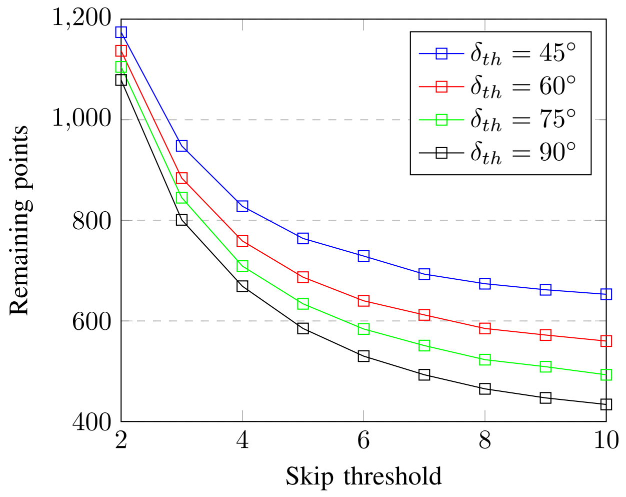

Fig. 8. Trajectory two

VI. EVALUATIONS

This section introduces a dataset and then our proposed algorithms are evaluated on the basis of three aspects: compression ratio, compression time and error metrics. Finally, we discuss the results and summarize the performance of the proposed algorithms. Three algorithms (DP, TD-TR, TD-TR Reduce) were written in Python, while MRPA was written in Matlab. The experiment is conducted on Windows 10 with 4 CPU cores (Intel i7-7700K with 4.20GHz) and 32 GB RAM.

A. *Dataset*

The Geolife dataset was collected by 182 participants in the (Microsoft Research Asia) Geolife project [9] for five years (from April 2007 to August 2012). Different transport modes, including biking, walking and traveling, are included in the data set. Most of the data collection has taken place in China, Beijing. More than 90% of trajectories are collected in a dense format, e.g. every 1 to 5 seconds or every 5 to 10 meters per point. The data set was cleaned to remove trajectories with high noise such as large jumps in time and space.

B. *Experiment Settings*

For this simulation, three trajectories are chosen to observe the effect of $skip_{th}$ and δ_{th} on the number of remaining points. Figures 6-8 show the number of stop points under different heading threshold and skip threshold. The details of each trajectory are shown in Table 1. From Figures 6-8, the curve decreases dramatically when $skip_{th}$ increases from 2 to 3 and from 3 to 4. The curve starts to changes slowly when $skip_{th} \geq 5$, so we set $skip_{th} = 5$. On the other hand, the number of remaining points decreases significantly when δ_{th} change from 45° to 60° and starts to change constantly when $\delta_{th} \geq 60°$, so we set δ_{th} to 60°.

TABLE I
TRAJECTORY DETAILS

Trajectory ID	# of Points	Start	Stop
1	4,164	2009-02-20 04:01:36	2009-02-20 14:51:36
2	1,937	2008-10-28 23:51:59	2008-10-29 11:25:00
3	838	2009-02-24 12:16:55	2009-02-24 13:35:55

C. *Comparison of Trajectory Simplification Algorithm*

Our proposed TD-TR Reduce algorithm was compared against three other algorithms DP, TD-TR, MRPA) in terms of compression ratio, compression time and simplification error. To simulate the result, 25 trajectories are selected from the *Geolife* dataset and the trajectory details are shown in Figure 9. Figures 10-13 display the simulation results.

As shown in Figure 10, on compression time, DP outperforms other algorithms, while the compression time of MRPA is significantly longer than other algorithms. The reason could be that MRPA error metric (LSSD) requires higher computation cost. Furthermore, TD-TR Reduce achieve shorter compression time than both of TD-TR and MRPA. This is because the feature point extraction technique was adopted, which reduces the computational time significantly.

In Figure 11, with regard to the compression ratio, the plots of DP and MRPA are close to each other, while the plot of TD-TR and TD-TR Reduce are slightly lower. Meanwhile, TD-TR Reduce outperforms the traditional TD-TR algorithm.

Figures 12, illustrates that both TD-TR and TD-TR Reduce achieve much lower trajectory distance reduction ratio, while DP obtains the highest trajectory distance reduction ratio among all algorithms.

In Figures 13, the ASED error of DP is generally higher than other algorithms and the curves of TD-TR Reduce are slightly higher than Traditional TD-TR. Particularly, TD-TR always achieves the lowest ASED error.

Therefore, TD-TR Reduce makes a favorable trade-off between the compression ratio, the trajectory distance reduction ratio and the ASED error while having up to 33% lower compression time on large trajectory compares to the traditional TD-TR algorithm.

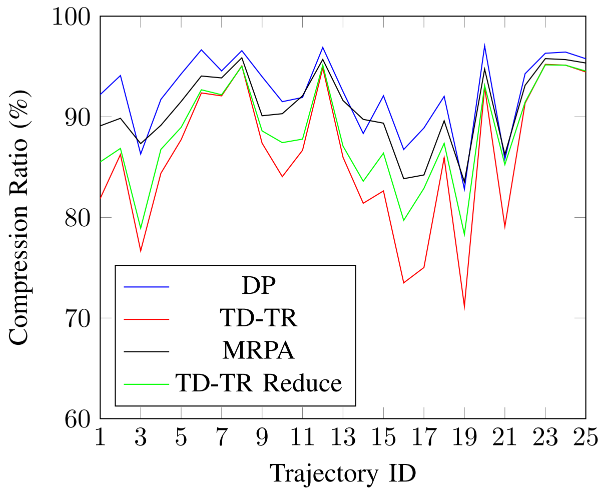

Fig. 11. Compression ratio

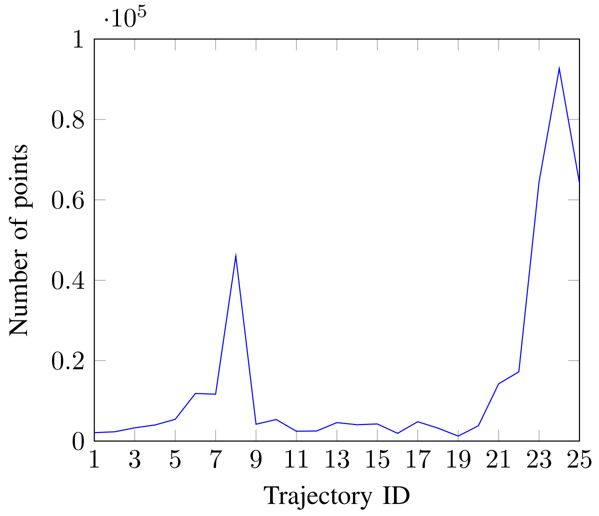

Fig. 9. Trajectory details

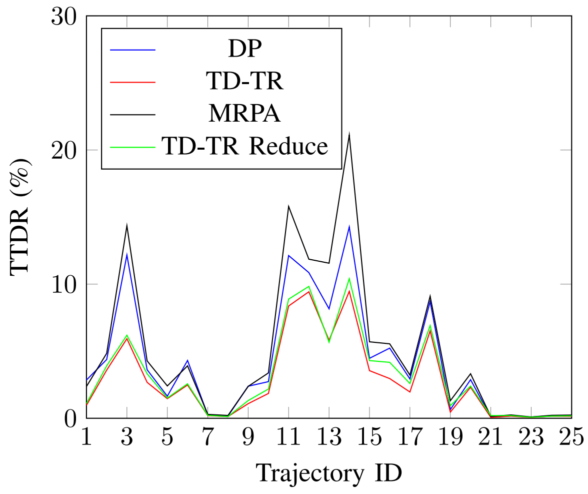

Fig. 12. Trajectory distance reduction ratio (TDDR)

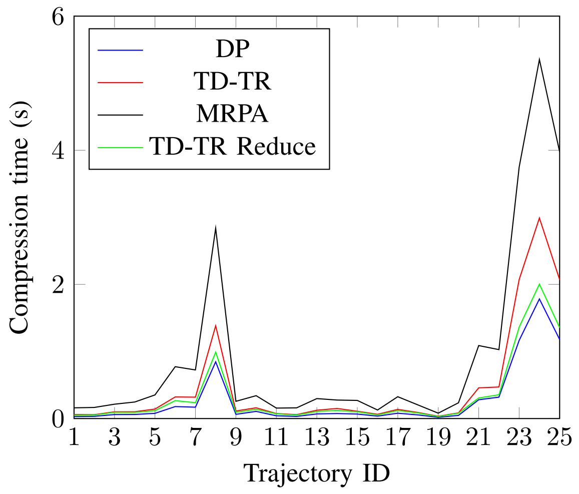

Fig. 10. Compression time

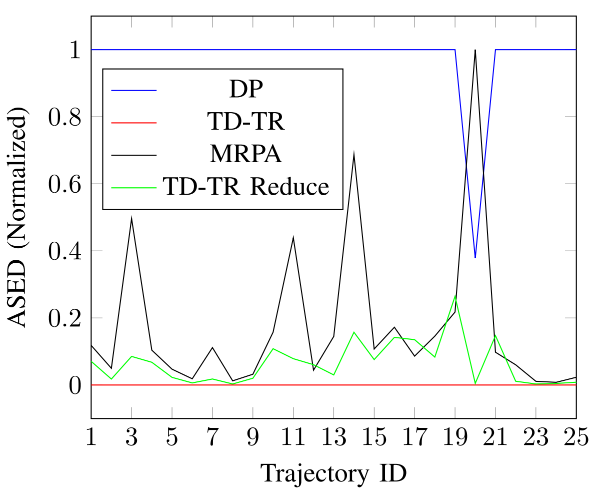

Fig. 13. Average SED error (ASED)

VII. Conclusion and Future Work

In this paper, we presented a new Trajectory Simplification Algorithm called TD-TR Reduce, which proposes a new method of data reduction based on the extraction of a feature point. Only the important points will be retained by using this data reduction method. The algorithm then performs the traditional TD-TR algorithm on the extracted feature point set. The experiments show that a favorable trade-off between the simplification rate, the distance reduction ratio and the ASED error with up to 33% lower compression time can be achieved through the proposed algorithm. Future studies should identify appropriate dynamic parameters for the heading threshold and skip threshold as well as investigate the effectiveness of TD-TR Reduce on different datasets.

References

[1] 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, Jul. 2013.

[2] N. Meratnia and R. A. de By, "Spatiotemporal Compression Techniques for Moving Point Objects," in Advances in Database Technology - EDBT 2004, Springer Berlin Heidelberg, 2004, pp. 765–782.

[3] D. Zhang, M. Ding, D. Yang, Y. Liu, J. Fan, and H. T. Shen, "Trajectory simplification," Proceedings of the VLDB Endowment, vol. 11, no. 9, pp. 934–946, May 2018.

[4] D. Douglas and T. Peucker, "Algorithms for the reduction of the number of points required to represent a line or its caricature,cartographica," Cartographica: The International Journal for Geographic Information and Geovisualization, vol. 10, no. 2, pp. 112–122, Dec. 1973.

[5] N. Meratnia and R. A. de By, "Spatiotemporal Compression Techniques for Moving Point Objects," in Advances in Database Technology - EDBT 2004, Springer Berlin Heidelberg, 2004, pp. 765–782.

[6] Minjie Chen, Mantao 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, May 2012.

[7] J. Muckell, J.-H. Hwang, V. Patil, C. T. Lawson, F. Ping, and S. S. Ravi, "SQUISH," in Proceedings of the 2nd International Conference on Computing for Geospatial Research & Applications - COM.Geo '11, 2011.

[8] Y. Zheng, Y. Chen, Q. Li, X. Xie, and W.-Y. Ma, "Understanding transportation modes based on GPS data for web applications," ACM Transactions on the Web, vol. 4, no. 1, pp. 1–36, Jan. 2010.

[9] Z. Yu, X. Xing, and W.-Y. Ma, "Geolife: a collaborative social networking service among user, location and trajectory," IEEE Data Engineering Bulletin, vol. 33, no. 2, pp. 32–40, 2010.