






SESA: Fast Trajectory Compression Method Using Sub-trajectories Segmented by Stay Areas

Shota Iiyama^{1,3}, Tetsuya Oda^{2,3}, and Masaharu Hirota^{1,3}

¹ Graduate School of Informatics, Okayama University of Science,
1-1 Ridaicho, Kita-ku, Okayama, Japan
i21im01cs@ous.jp, hirota@ous.ac.jp

² Department of Information and Computer Engineering,
Okayama University of Science, 1-1 Ridaicho, Kita-ku, Okayama, Japan
oda@ous.ac.jp

³ Department of Information Science, Okayama University of Science,
1-1 Ridaicho, Kita-ku, Okayama, Japan

Abstract. The increase in trajectory data is associated with problems, such as increased storage costs and difficulty in analysis. One solution to these problems is trajectory compression. It reduces the data size by removing redundant positioning points from the original trajectory data. This study proposes SQUISH-E(μ) with Stay Areas (SESA), a fast batch compression method for trajectory data based on the stay area. A stay area is where the user stays for a certain period, such as the waiting time for a traffic light or bus. Moreover, SESA segments the trajectory into sub-trajectories using stay areas and applies SQUISH-E(μ) to each sub-trajectory. The experiments with trajectory datasets show that SESA achieves a compression time reduction of approximately 65% with little change in trajectory feature retention and approximately the same compression rate as the direct application of SQUISH-E(μ).

Keywords: Trajectory compression · GPS · Batch compression

1 Introduction

Devices with global positioning systems (GPS), such as smartphones, drones, and car navigation systems, are widespread. These devices generate an enormous amount of trajectory data that record an object's activities. Such trajectory data can be used to track the movements of people, vehicles, and other objects. Therefore, trajectory data are important sources of information for analyzing human behavior patterns [11, 13] and next location prediction [2, 12].

However, the vast amount of trajectory data causes problems. First, communicating trajectory data increases the amount of communication in the network. Second, storing a large amount of trajectory data incurs enormous storage costs. Third, because several positioning points in the trajectory data are redundant,

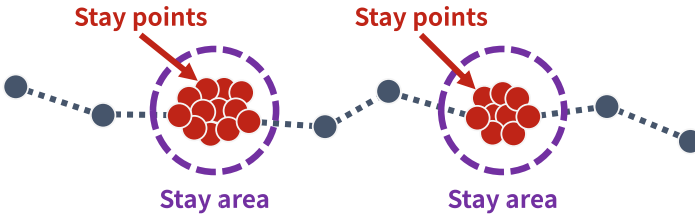


Fig. 1. Stay points and stay areas in a trajectory data.

the analysis requires extra time, which is essentially unnecessary. One solution to these problems is trajectory compression. Trajectory compression is a method that reduces the data size by removing redundant positioning points from the original trajectory data.

Several trajectory compression methods have been proposed. These include the Douglas-Peucker (DP) algorithm [3] and Spatial QUality Simplification Heuristic (SQUISH) [8]. However, such trajectory compression methods preserve the shape of the trajectory or movement speed of the object; nonetheless, they do not preserve the information on the area where the user stays. In this study, we refer to such an area as a stay area. A stay area provides valuable information, such as analyzing the area where a user stays and the stay of the associated traffic flow. Therefore, it is desirable to preserve the stay area of the original trajectory data when applying the trajectory compression method.

Here, we define the stay area and point in the trajectory data. Figure 1 illustrates trajectory data containing stay areas and points. The purple-dashed circles represent the stay areas, and the red points surrounded by circles represent the stay points. Stay points are positioning points (such as a bus stop or traffic lights) recorded when a user remains at a particular location for a certain period. Therefore, the stay area is defined by a set of consecutive stay points. Because the first and last stay points are sufficient to represent the information of a user's stay time and location, we consider the remaining points as redundant points in this study. In this study, we also refer to the first and last points of the stay area and positioning points where the shape of the trajectory and speed of movement have changed significantly as the feature points.

Even compression results from the existing trajectory compression methods that do not have a specific process for extracting the stay area may accidentally preserve the stay areas. However, the compression results do not necessarily preserve the first and last points of the stay area. This indicates that the compression results may not preserve information regarding the exact stay time of the user at that location. In addition, several current trajectory compression methods change the degree of compression by adjusting the parameters. Consequently, it may be possible to adjust these methods to produce compression results within a stay area. For example, when setting the parameters that decrease the compression rate, the compression results obtained by these methods may preserve both the first and last points of the stay area. However, because the compression

result preserves the stay points other than the first and last points of the stay area and other redundant points, its compression rate of the compression result is low. Furthermore, when setting the parameters that increase the compression rate, the compression method considers almost positioning points in the trajectory data as redundant points and removes them. This possibly removes the stay points. These facts make it difficult for the existing methods to compress trajectory data with a high compression rate, while preserving the stay areas.

In this study, we propose a batch method of SQUISH-E(μ) with Stay Areas (SESA) for fast trajectory compression. Furthermore, SESA uses Spatial Quality Simplification Heuristic - Extended (μ) (SQUISH-E(μ)) [9], which preserves the trajectory shape and moving speed, and Stay Area (SA) to extract stay areas based on the method for extracting stay points proposed in [14]. The SESA applies SQUISH-E(μ) to the sub-trajectories segmented by the stay area extracted by the SA. Consequently, the compression time of the SESA is fast, and the compression rate is high, while ensuring that the stay area in the trajectory data is preserved.

The contributions of this research are summarized as follows.

1. We propose a batch compression method SESA extended from SQUISH-E(μ). The SESA reliably preserves the stay area of trajectory data, because SESA applies SQUISH-E(μ) to sub-trajectories segmented by stay areas.
2. SESA can compress most trajectory data approximately faster than applying original SQUISH-E(μ) with almost no change in the compression rate of the locus and the preservation rate of the feature amount.

The rest of this study is organized as follows. Section 2 describes the previous compression methods related to our method. Section 3 describes the proposed method in detail. In Sect. 4, we describe an experimental evaluation of the proposed method. Section 5 provides a summary of the study.

2 Related Work

There are two types of trajectory compression methods: batch and online methods.

Batch compression methods compress the trajectory data using all the positioning points. Therefore, the batch compression method can be applied only to the trajectory data after positioning is complete. The advantage of batch compression methods is their high performance. This is because they use all positioning point information for compression. The DP [3] preserves the shapes of trajectories. Additionally, DP recursively computes the compression points and segments a trajectory based on the perpendicular Euclidean distance (PED). The top-down time-ratio (TD-TR) [7] is an extension of the DP. It uses the synchronized Euclidean distance (SED), which considers the timestamp of the positioning points. We elaborate the SED in Sect. 3.3. This method preserves the trajectory shape and speed of movement. However, the compression time of TD-TR is slower than that of DP because SED requires more computational



Fig. 2. Overview of SESA.

time than the PED. The top-down time-ratio Reduce (TD-TR Reduce) [5] is an extension of the TD-TR to reduce the compression time. This method extracts feature points from the trajectory data with movement direction and speed have changed and applies TD-TR to the extracted points. The method proposed in [4] preserves the contour of the trajectory data (particularly turning corners and U-turns), considering the distance, angle, and velocity of the trajectory data. Furthermore, SQUISH-E(μ) [9] is a trajectory compression method that uses a priority queue. This method adds positioning points to the priority queue and removes points with a low priority. The remaining points in the priority queue are regarded as the compression results. The method proposed in [6] to compress a trajectory while preserving feature points applies SQUISH-E(μ) and the method for extracting stay points proposed in [14] to trajectory data in parallel.

On the other hand, online compression methods compress trajectory data using only the acquired positioning and few other points. Therefore, an online compression method can be used during the positioning process. Online compression methods compress data sequentially; thus, their performance regarding the compression rate and preservation of the original trajectory features is often lower than that of batch processing. Dead-Reckoning [10] is a fast trajectory compression method that uses the speed of an object to predict the subsequent moving point and its distance from the actual positioning point to determine whether it is a compression point. Critical Point (CP) [1] is a trajectory compression method that focuses on the direction of movement. Regarding this method, the compression point is the location where the direction of movement of the user has changed.

The SESA of our proposed method compresses each sub-trajectory after the SA segments the trajectory data. Therefore, SESA can select any compression method, batch, or online. In this study, we use one of the primary methods for trajectory compression: SQUISH-E(μ).

3 Proposed Method

Figure 2 presents an overview of SESA. A trajectory is defined as $T = \{p_1, p_2, \dots, p_n\}$, and the i -th positioning point $p_i = (x_i, y_i, t_i)$ represents the longitude, latitude, and timestamp, respectively. First, we apply SA to extract the stay areas from the trajectory T . Second, we segment the trajectory into sub-trajectories using the first and last stay points of each stay area. Third, we apply SQUISH-E(μ) to each sub-trajectory to extract the compression points from them. Finally, we integrate the compression points extracted from each sub-trajectory and obtain the compression result T' .

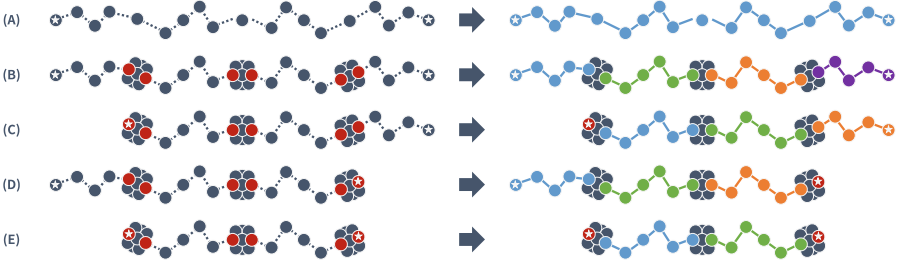


Fig. 3. Segmentation process.

3.1 Extraction of Stay Area

In this section, we explain the process of extracting stay areas from trajectory data using SA. The SESA uses SA to extract the stay areas from trajectory data T .

The SA defines the positioning points in the trajectory data that satisfy all of the following conditions as stay points:

$$\begin{aligned}
 a &< i \leq b \\
 \text{Distance}(p_a, p_i) &\leq T_{\text{distance}} \\
 \text{Distance}(p_a, p_{b+1}) &> T_{\text{distance}} \\
 \text{Interval}(p_a, p_b) &\geq T_{\text{time}}
 \end{aligned} \tag{1}$$

$\text{Distance}(p_a, p_i)$ is the Euclidean distance between positioning points p_a and p_i . T_{distance} is the threshold distance between the two positioning points. $\text{Interval}(p_a, p_b)$ is the difference in the timestamps between positioning points p_a and p_b . T_{time} is the threshold of the difference in the timestamps between the two positioning points.

After extracting the stay points, we regard each consecutive set of stay points in the trajectory data as stay areas. Subsequently, we remove all stay points, excluding each extracted first and last point of the stay area. We used each extract stay area to segment the trajectory into sub-trajectories.

3.2 Segmentation

Here, we describe the segmentation process of a trajectory to sub-trajectories. We segment the trajectory into sub-trajectories by the stay area extracted by SA, as shown in Fig. 3. In the figure, the segmentation process has five cases depending on the locations of extracted stay area. The red points are the first point and the last point of stay area. The points marked with a star are the first point p_1 and the last point p_n of the trajectory. Also, we apply SQUISH-E(μ) described in the next section to the segmented sub-trajectories.

3.3 SQUISH-E(μ)

Next, we describe the SQUISH-E(μ). SQUISH-E(μ) is a trajectory compression method that uses priority queues. Priority is the degree of change in the shape of the trajectory data when a positioning point p_i is dequeued from the priority queue. In SQUISH-E(μ), the shape differences in the trajectory before and after the dequeuing do not exceed the threshold μ . Therefore, this method preserves the shape and moving speed of the trajectory data.

Moreover, SQUISH-E(μ) comprises the following two steps.

step 1: Enqueue all the positioning points of a trajectory into the priority queue Q .

step 2: Dequeue the low-priority points from Q .

Step 1 enqueues the positioning point p_i of the trajectory to the priority queue Q . Here, we denote the priority of p_i as $priority(p_i)$. When enqueueing p_i to Q , the method initializes $priority(p_i) \leftarrow \infty$ and $\pi(p_i) \leftarrow 0$. The variable $\pi(p_i)$ is used to maintain the neighborhood's maximum priority of the removed points in the subsequent step. When enqueueing a positioning point other than the first point ($i \geq 3$), we change the priority $priority(p_{i-1})$ as follows:

$$priority(p_{i-1}) = SED(p_{i-1}, pred(p_{i-1}), succ(p_{i-1})), \quad (2)$$

where $pred(p_i)$ and $succ(p_i)$ are p_i 's closest predecessor and successor among the points in Q , respectively. The SED is the Euclidean distance between the positioning $p_i = (x_i, y_i, t_i)$ and pseudo $p'_i = (x'_i, y'_i, t_i)$ points. The pseudo-point p'_i is the point approximated on the line between the two positioning points $\{p_s(x_s, y_s, t_s) | 1 \leq s < i\}$ and $\{p_e(x_e, y_e, t_e) | i < e \leq n\}$. In this study, we calculate the SED between p_i and p'_i based on p_s and p_e , and we denote it as $SED(p_i, p_s, p_e)$. Moreover, $SED(p_i, p_s, p_e)$ is calculated as follows:

$$(x'_i, y'_i) = \left(x_s + \frac{t_i - t_s}{t_e - t_s}(x_e - x_s), y_s + \frac{t_i - t_s}{t_e - t_s}(y_e - y_s) \right) \quad (3)$$

$$SED(p_i, p_s, p_e) = \sqrt{(x_i - x'_i)^2 + (y_i - y'_i)^2}$$

Hence, the priorities of the first p_f^Q and last p_l^Q points of Q are $priority(p_f^Q) \leftarrow \infty, priority(p_l^Q) \leftarrow \infty$, which prevent them from being removed from Q . After inquiring about all the positioning points to Q , the method moves to Step 2.

In step 2, the method dequeues the positioning points with low-priority values in Q and compresses the trajectory using the following procedure.

1. We find the lowest priority positioning point p_j in Q .
2. We update $\pi(pred(p_j))$ and $\pi(succ(p_j))$ based on $priority(p_j)$. If $\pi(pred(p_j)) < priority(p_j)$, we set $\pi(pred(p_j)) \leftarrow priority(p_j)$. Furthermore, if $\pi(succ(p_j)) < priority(p_j)$, then $\pi(succ(p_j)) \leftarrow priority(p_j)$.

3. We dequeue p_j from Q and update the priorities of $pred(p_j)$ and $succ(p_j)$. Here, we let p_k be either $pred(p_j)$ or $succ(p_j)$. If p_k is the first or last point in Q , then the priority is not updated. Otherwise, $priority(p_k) = \pi(p_k) + SED(p_k, pred(p_k), succ(p_k))$.
4. We repeat the process until the minimum priority value in Q is larger than μ .

After the completion of Step 2, the points in Q are the compression results of SQUISH-E(μ).

3.4 Integration

The last process of SESA is the integration of the compression results from the sub-trajectories. We integrate the compression points of each sub-trajectory extracted using SQUISH-E(μ). Their integration result is the compression result T' obtained by the SESA. Considering (C), (D), and (E) in Fig. 3, T' also includes the first and last points of the trajectory as described in Sect. 3.1. Consequently, the SESA can reliably preserve the compression points of SQUISH-E(μ) and the stay areas.

4 Experiment

In this section, we evaluate the performance of the SESA of our proposed method and the conventional SQUISH-E(μ).

4.1 Experimental Conditions

We used the Microsoft GeoLife dataset [15–17] as the experimental data. The total number of trajectories was 18,670. They were recorded using transportation such as walking, cars, and trains. The mean of positioning points for each trajectory were approximately 1,295. The mean sampling rates of the trajectories were approximately 7 s.

We used three evaluation criteria: the SED error, compression rate, and compression time. The SED error evaluates the difference in the trajectory shape before and after the trajectory compression. The smaller the SED error value is, the better the compressed trajectory data that preserve the shape of the original trajectory data. The compression rate evaluates the percentage of the positioning points removed using the compression method. The higher the value of the compression rate is, the smaller the data size of the compressed trajectory. The compression time evaluates the time required for the trajectory compression method to compress the data. The smaller the compression time is, the faster the method compresses the trajectory data.

We evaluated each SED threshold of SESA and SQUISH-E(μ), varying from 10 m to 100 m for each 10 m. We adjusted the SA thresholds of the SESA, setting the thresholds for the stay distance and time to 15 m and 30 s, respectively.

4.2 Experiment Results

Figure 4, 5, and 6 show the evaluation results of the SESA of the proposed method and SQUISH-E(μ) with three evaluation criteria. The vertical and horizontal axes represent each criterion and SED threshold, respectively. The red and blue lines represent the performances of SESA and SQUISH-E(μ), respectively.

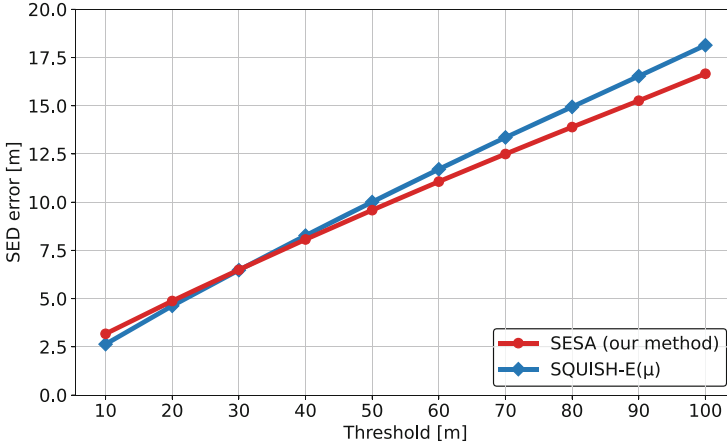


Fig. 4. SED Error.

SED Error. Figure 4 shows the SED errors of SESA and SQUISH-E(μ). When the SED threshold is less than 30 m, the SESA has more significant SED errors than SQUISH-E(μ). This is because SQUISH-E(μ) preserves several redundant points in the stay area. Moreover, the shape of the compression result is similar to that of the original trajectory, whereas the SESA preserves only the first and last points in the stay area. Therefore, SQUISH-E(μ), which has a more significant number of compression points, produces compression results closer to the shape of the original trajectory data than the SESA. However, the SESA reliably preserves the stay area without redundant stay points.

In addition, when the SED threshold is higher than 40 m, the SESA has lower SED errors than SQUISH-E(μ). Furthermore, the larger the threshold is, the more significant the difference. This is because SQUISH-E(μ) removes most of the positioning points and cannot preserve the stay area, whereas the SESA can preserve the stay area. When the SED threshold of SQUISH-E(μ) is significant, the method may excessively compress the trajectory data because it deletes the stay area to be preserved in addition to the redundant points. In contrast, the compression result by SESA is close to the shape of the original trajectory data because it preserves the first and last points of the stay area.

Compression Rate. Figure 5 shows the compression rates of SESA and SQUISH-E(μ). The compression rates of the SESA and SQUISH-E(μ) are

approximately identical. This difference is caused by the SESA, which preserves the stay points.

When the SED threshold is less than 10 m, SESA has a higher compression rate than SQUISH-E(μ). This is because SQUISH-E(μ) preserves several stay points in each stay area, whereas SESA preserves only the first and last points in each stay area. Thus, SESA has a higher compression rate than SQUISH-E(μ).

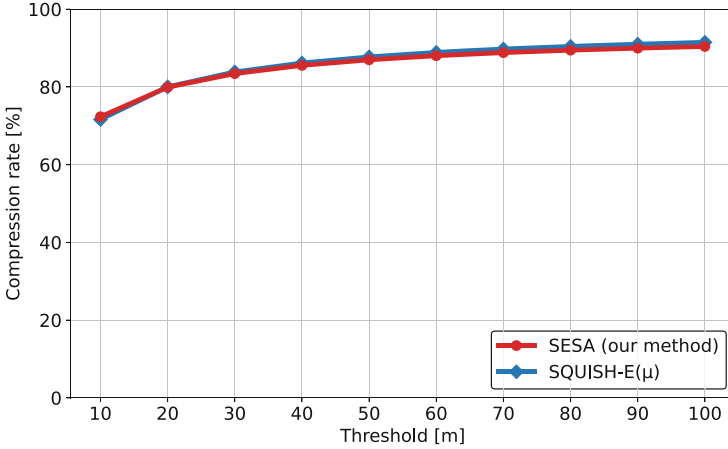


Fig. 5. Compression rate.

In addition, when the SED threshold is larger than 20 m, SESA has a lower compression rate than SQUISH-E(μ). This is because the SESA preserves the stay area, which cannot be preserved by SQUISH-E(μ). The compression points of SESA increased because the number of preserved stay areas increased.

Compression Time. Figure 6 shows the compression times of SESA and SQUISH-E(μ). Regarding all the SED thresholds, the compression time of SESA is approximately 65% less than that of SQUISH-E(μ).

This is because SESA segments the trajectory into sub-trajectories using SA. The time complexity of SQUISH-E(μ) is $\mathcal{O}(N \log N)$, which tends to have a shorter compression time because the length of the trajectory to be compressed decreases by the segmentation process. Consequently, despite the increase in the number of processes for extracting the area of stay, the compression time of the SESA is fast for most movement trajectories.

4.3 Discussion

Here, we discuss the time complexity and applicability of SESA.

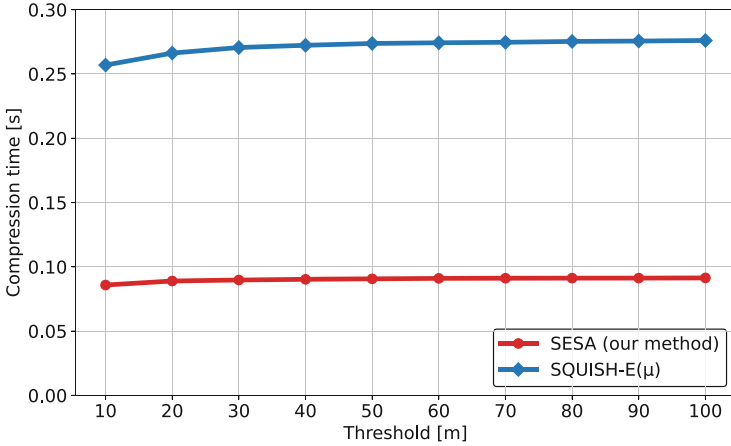


Fig. 6. Compression time.

Time Complexity. We compare the time complexities of SESA and SQUISH-E(μ). The complexity of SQUISH-E(μ) is $\mathcal{O}(N \log N)$, where N denotes the number of positioning points in the trajectory data. In addition, the time complexity of the SA is $\mathcal{O}(N)$.

The best complexity of SESA is $\mathcal{O}(N + N \log \frac{N}{|st|})$, where $|st|$ is the number of sub-trajectories in the trajectory data. Because the method applies SQUISH-E(μ) to each sub-trajectory, the compression time is fastest when the number of positioning points in each sub-trajectory is equal.

Subsequently, the worst case is that the SA does not find the stay area in the trajectory data. The time complexity of SA $\mathcal{O}(N)$ is added to SQUISH-E(μ). Consequently, the worst-case time complexity of SESA is $\mathcal{O}(N + N \log N)$. In addition, when the number of positioning points of the sub-trajectories segmented by SA is non-uniform, the time complexity of SESA approaches $\mathcal{O}(N + N \log N)$. Therefore, the time complexity of the SESA varies between $\mathcal{O}(N + N \log \frac{N}{|st|})$ and $\mathcal{O}(N + N \log N)$, depending on the location and number of extracted sub-trajectories in the trajectory.

Applicability. Because SESA segments the trajectory data by the stay areas into sub-trajectories, the compression time of SESA depends on the number of stay areas in the trajectory data. Therefore, to investigate the possibility of the SESA working effectively, we count the number of stay areas in the trajectory data. Figure 7 shows a scatter plot of the number of stay areas in 1,000 randomly sampled trajectory data of our dataset. Furthermore, the average number of stay areas in all trajectory data are 13.22. This indicates that most of the trajectory data may contain multiple stay areas. This is because, the trajectory data may include stay areas owing to events such as waiting at a traffic light or stopping at a stoplight during the user or object movement. Therefore, the SESA can

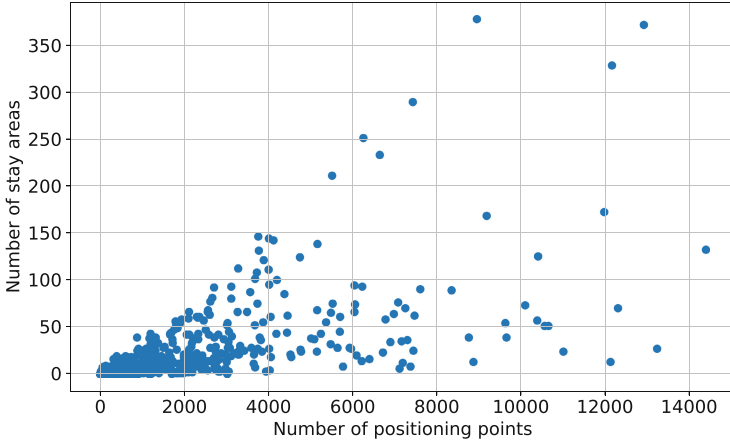


Fig. 7. The number of stay areas in each trajectory data.

potentially compress the trajectory data at high speeds although this depends on the location of stay areas in the trajectory data.

Furthermore, based on the compression time in Fig. 6, SESA is faster than SQUISH-E(μ) throughout the SED threshold. Therefore, SESA is a fast trajectory compression method.

5 Conclusion and Future Work

In this study, we have presented SESA, a fast trajectory compression method that is an extension of SQUISH-E(μ). The SESA creates sub-trajectories based on the user stay area extracted from the trajectory data and applies SQUISH-E(μ) to each sub-trajectory. In evaluation experiments, we compared SQUISH-E(μ) to SESA based on several evaluation criteria. The results show that SESA requires approximately one-third of the compression time of SQUISH-E(μ) with a slight change in the feature retention performance.

We plan to extend this study in the following directions. We will improve the SESA for a faster compression time. In addition, because SESA is faster with a more significant number of trajectory segments, we consider using feature points other than the stay area to extract the points segmented from the trajectory.

Acknowledgements. This work was supported by JSPS KAKENHI Grant Number JP19K20418 and 20K12081.

References

1. Barbeau, S., et al.: Dynamic management of real-time location data on GPS-enabled mobile phones. In: The Second International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies, pp. 343–348 (2008)

2. Chen, M., Zuo, Y., Jia, X., Liu, Y., Yu, X., Zheng, K.: CEM: a convolutional embedding model for predicting next locations. *IEEE Trans. Intell. Transp. Syst.* **22**(6), 3349–3358 (2021)
3. Douglas, D.H., Peucker, T.K.: Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: Int. J. Geogr. Inf. Geovisualization* **10**(2), 112–122 (1973)
4. Feng, S., Chen, L., Ma, M., Yang, A.: A turning contour maintaining method of trajectory data compression. *IOP Conf. Ser.: Earth Environ. Sci.* **513**(1), 012058 (2020)
5. Hansuddhisuntorn, K., Horanont, T.: Improvement of TD-TR algorithm for simplifying GPS trajectory data. In: 2019 First International Conference on Smart Technology Urban Development, pp. 1–6 (2019)
6. Iiyama, S., Oda, T., Hirota, M.: An algorithm for GPS trajectory compression preserving stay points. In: *Advances in Internet. Data & Web Technologies*, pp. 102–113. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-95903-6_12
7. Meratnia, N., de By, R.A.: Spatiotemporal compression techniques for moving point objects. In: Bertino, E., et al. (eds.) *EDBT 2004. LNCS*, vol. 2992, pp. 765–782. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24741-8_44
8. Muckell, J., Hwang, J.H., Patil, V., Lawson, C.T., Ping, F., Ravi, S.S.: SQUISH: an online approach for GPS trajectory compression. In: *Proceedings of the 2nd International Conference on Computing for Geospatial Research & Applications*, pp. 1–8 (2011)
9. Muckell, J., Olsen, P.W., Hwang, J.H., Lawson, C.T., Ravi, S.S.: Compression of trajectory data: a comprehensive evaluation and new approach. *GeoInformatica* **18**(3), 435–460 (2014). <https://doi.org/10.1007/s10707-013-0184-0>
10. Trajcevski, G., Cao, H., Scheuermann, P., Wolfsonz, O., Vaccaro, D.: On-line data reduction and the quality of history in moving objects databases. In: *Proceedings of the 5th ACM International Workshop on Data Engineering for Wireless and Mobile Access*, pp. 19–26. Association for Computing Machinery (2006)
11. Yang, L., Wu, L., Liu, Y., Kang, C.: Quantifying tourist behavior patterns by travel motifs and geo-tagged photos from flickr. *ISPRS Int. J. Geo-Inf.* **6**(11), 345 (2017)
12. Yao, D., Zhang, C., Huang, J., Bi, J.: Serm: a recurrent model for next location prediction in semantic trajectories. In: *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 2411–2414 (2017)
13. Yuan, Y., Medel, M.: Characterizing international travel behavior from geotagged photos: a case study of Flickr. *PLoS ONE* **11**(5), 1–18 (2016)
14. Zheng, V.W., Zheng, Y., Xie, X., Yang, Q.: Collaborative location and activity recommendations with GPS history data. In: *Proceedings of the 19th International Conference on World Wide Web*, pp. 1029–1038. Association for Computing Machinery (2010)
15. Zheng, Y., Li, Q., Chen, Y., Xie, X., Ma, W.Y.: Understanding mobility based on GPS data. In: *Proceedings of the 10th International Conference on Ubiquitous Computing*, pp. 312–321. Association for Computing Machinery (2008)
16. Zheng, Y., Xie, X., Ma, W.: GeoLife: a collaborative social networking service among user, location and trajectory. *IEEE Data Eng. Bull.* **33**(2), 32–39 (2010)
17. Zheng, Y., Zhang, L., Xie, X., Ma, W.Y.: Mining interesting locations and travel sequences from GPS trajectories. In: *Proceedings of the 18th International Conference on World Wide Web*, pp. 791–800. Association for Computing Machinery (2009)