Research Article

Velocity-Preserving Trajectory Compression Based on Retrace Point Detection

Anbang Chen ^{1,2} and Linfeng Liu ^{1,2}

¹School of Computer Science and Technology, Nanjing University of Posts and Telecommunications, Nanjing 210023, China

²Jiangsu Key Laboratory of Big Data Security and Intelligent Processing, Nanjing University of Posts and Telecommunications, Nanjing 210023, China

Correspondence should be addressed to Linfeng Liu; liulf@njupt.edu.cn

Received 25 October 2020; Revised 19 July 2021; Accepted 30 August 2021; Published 20 September 2021

Academic Editor: Paolo Barsocchi

Copyright © 2021 Anbang Chen and Linfeng Liu. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

With the increasing development of GPS-equipped mobile devices such as smart phones and vehicle navigation systems, the trajectories containing valuable spatiotemporal information are recorded. Typically, plenty of trajectory records are generated and stored, making the device memory suffer a heavy storage pressure. Thus, it is a vital issue to compress the trajectories. The trajectory semantics are usually ignored or reduced in traditional trajectory compression techniques. In addition, most of existing trajectory compression algorithms only concern the position errors rather than the velocity errors of trajectories. This paper proposes a velocity-preserving trajectory compression algorithm based on retrace point detection (VPTC-RP) that can compress a set of trajectories by removing unnecessary redundancy points, while the skeleton of these trajectories is maintained as much as possible. In VPTC-RP, the retrace points and the velocity errors are taken to reflect the speeds and directions attached with the points. VPTC-RP first determines the retrace points based on the changed movement directions, and then, the retrace points are extracted from the original trajectories. Especially, the retrace points are put in a buffer, and the subtrajectories in the buffer are compressed according to the measured velocity errors. Simulations are carried out on the Geolife trajectory dataset, and the simulation results indicate that VPTC-RP can achieve a preferable tradeoff among the compression error, compression ratio, and running time.

1. Introduction

With the rapid development of wireless communication and mobile computing technologies, more and more devices (e.g., mobile phones, smart watches/bands, and auto navigators) are equipped with GPS modules, and thus, the amount of trajectory data collected by GPS-enabled devices is drastically increased. To exploit the valuable information hidden in the trajectory data, the issue of trajectory mining has attracted more and more attention, and the trajectory mining results can be applied to various fields, such as location recommendations [1], destination predictions [2], and personal navigations [3]. For example, the travel modes and the preferential restaurants of some people (nodes) can be reflected by their personal trajectories. Thus, with the previ-

ous trajectories, proper travel modes and restaurants can be recommended to the nodes.

Especially, note that the generous trajectory data brings a heavy storage burden. For example, when a GPS device records the positions at the interval of two seconds, it yields more than 10,000 points in one day (the size of recorded trajectories of 2000 moving nodes will reach 1 GB). Besides, much time [4] will be consumed to upload and download the trajectory data between the mobile devices and the servers, and thus, the real-time positions are hardly obtained and exploited.

Therefore, it is essential to select and discard some points from the original trajectories, while the main features (e.g., the movement speeds and directions of nodes) of trajectories are expected to be preserved. To address this issue,

several works have been employed to compress the GPS trajectory data using some trajectory compression strategies, and the trajectory storage pressure can be relieved accordingly.

The error measures evaluate the degree of deviations between the original trajectories and the compressed trajectories, and the main principle of trajectory compression is to compress the trajectories under a certain error measure. Most of trajectory compression algorithms need to employ some error measures, and the compressed trajectories contain different trajectory feature losses. Therefore, different error measures should be selected for different compression targets, such as the position information, speed information, and direction information. The main objectives of trajectory compression methods are to reduce the number of trajectory points and reduce the compression errors, while some existed algorithms only make a tradeoff between the compression accuracy and storage size.

Most of existing methods concern the position errors rather than the velocity features of trajectories. The velocity represents the moving distance in a particular direction per unit time. Velocity features must be considered several scenarios and applications [5, 6]. For instance, the velocity features of different nodes are much different, such as the nodes with various travel modes: walking and taking the vehicles of bikes, buses, or trains. The existing velocity-preserving trajectory compression methods [7, 8] usually attempt to reduce the speed errors, and these methods focus on some speed components in the velocity, while the direction information is usually neglected.

Moreover, some current works have ignored the semantic meanings of trajectories. Hence, semantic trajectory compression methods are used to solve this problem. Although the original trajectory can record the movements of a mobile object, it is hard to understand the meaning of the trajectory by the coordinates including latitudes and longitudes. The original trajectories can be converted into the points of interest sequences and become the semantic forms that people can easily understand. Schmid et al. [9] propose the concept of semantic trajectory compression, and they think the dynamic state of trajectory can be roughly expressed by some meaningful states and related events. In [10], the stay point is firstly proposed to extract the life patterns from original trajectories. A stay point represents a geographic region where the node stays for a specific period. Each stay point carries some semantic meanings and hidden knowledge. For example, the stay points can map the trajectories to a semantic level and then find related regions of interest (e.g., a restaurant, an office building, and department stores). Besides, by analyzing the semantic meanings of trajectories, the retrace behaviors of nodes can be found to compress the trajectories. Especially, the points where the directions of nodes change sharply are referred to as retrace points (e.g., someone lingers around a shopping region or some tourists visit the scenic spots), and these retrace points can help to compress the original trajectories.

The movement of a pedestrian node typically exhibits a strong randomness. Thus, this paper puts forward a trajectory compression algorithm based on retrace point detection to mine the semantics of trajectories. Specially, to overcome

the limitations of speed errors, we take into account the direction features of velocity and propose an online direction-preserving trajectory compression method. In this paper, the main contributions are given as follows:

(1) A new trajectory compression algorithm is proposed to compress the trajectories through extracting the retrace points based on the maximum angular deviations of trajectory segments. The proposed algorithm can remove the redundant trajectory points, while ensuring the trajectory accuracy

(2) The circumferential angle theorem and distance ratio between trajectory points are applied to define the retrace area where each trajectory point can be taken as a retrace point

(3) The velocity vector standardized Euclidean distance is evaluated to measure the velocity (speed and direction) error. More valuable information hidden in the trajectories can be reserved while compressing the trajectories according to the velocity error rather than the speed error

The rest of the paper is organized as follows: Section 2 gives some related works. The problem formulation is described in Section 3. Section 4 introduces the online velocity-preserving trajectory compression algorithm based on retrace point detection. Section 5 reports the simulation results for the performance evaluation of algorithms. Finally, Section 6 concludes this paper.

This work is a significant extension of our early work [11]. Specifically, many related works are reviewed. The proposed VPTC-RP is improved and explained with more details, e.g., a lemma and its proof are provided for defining the retrace region, and more simulations are conducted to clarify the merits of VPTC-RP. Besides, the time complexity of VPTC-RP is analyzed.

2. Related Works

In this section, we will briefly review the related algorithms of trajectory compression. These algorithms could be roughly classified into two categories: offline compression and online compression. Offline compression algorithms collect the complete trajectories and then delete the redundant trajectory points. The mobile nodes can collect real-time trajectory data and compress them at the same time when the nodes are in the moving states. Due to the integrity of the original trajectories, these algorithms are relatively easier to achieve global optimization and more suitable for offline data analysis, while online compression algorithms are more suitable for real-time applications.

The most famous offline trajectory compression algorithms called Douglas-Peucker (DP) algorithm [12] is put forward by Douglas and Peucker. DP algorithm is a heuristic algorithm which ensures that the max position error of the compressed trajectory is confined to a threshold. DP algorithm starts with the first point and the last point of each trajectory and recursively adds the points with max perpendicular Euclidean distance (PED), until the PED of each point reserved is below a preset error threshold. DP algorithm adopts a top-down strategy, and the compressed trajectory is of high quality. Compared with the original

FIGURE 1: A compressed trajectory by Douglas-Peucker algorithm.

trajectory, the compressed trajectory loses relatively little precision. However, the time complexity is very high, and it could reach $O(N^2)$ (N denotes the number of trajectory points). As shown in Figure 1, a trajectory starts from p_1 and ends with p_7 . The PEDs of p_4 and p_5 are greater than the threshold, and p_4 and p_5 are reserved.

Although DP algorithm can effectively compress the trajectories, it must run offline and require the full collected trajectories, which is not suitable for some real-time applications, i.e., the trajectories must be compressed immediately. To this end, a generic sliding-window algorithm and open window algorithm (OPW) [13] are proposed to improve the real-time issue, which compress the points in window moving on original trajectories. The sliding-window algorithm initializes with a segment called sliding-window between the first point and the third point; then, the algorithm calculates all position errors of points in the segment and adds the points with maximum error into the sliding-window repeatedly. Once one position error in the sliding-window is over threshold, the point over threshold is chosen as feature point to constitute compressed trajectory and the first point of new sliding-window. Different from the sliding-window algorithm, OPW calculates the sum of all position errors of points in the segment called open window. When the sum is larger than the threshold, the penultimate point in the open window is chosen as feature point to constitute compressed trajectory and the first point of new open window.

OPW is an online trajectory compression algorithm. Different from the offline compression algorithms, the online compression can handle local trajectories, and the error is larger than that of offline ones, while the average time complexity is much smaller.

Spatial QUALity Simplification Heuristic (SQUISH) algorithm [14] uses Time Synchronized Euclidean Distance (SED) error as constraint to compress the trajectories. It yields a short runtime, a high compression ratio, and a small trajectory error. However, SQUISH fails to guarantee the required compression ratio and compression error. SQUISH-E algorithm [15] is an extension of SQUISH, and it has the capability of minimizing the trajectory SED error under λ (compression ratio bound). Especially, SQUISH-E uses a priority queue to temporarily save the priority of points to be processed and repeatedly removes the points with the lowest priorities. When

a point is deleted, its two neighboring points will be adjusted according to the priority score of this point. Recently, Yang et al. [16] give some new error measurements and develop a two-component method to compress the trajectories. This method is derived from DP and can simplify the trajectories with a guaranteed position error, and it enhances the semantic components by a data enrichment strategy to restrict the speed error, which can preserve the original speed in the compressed representation as well.

Besides, in [17], an enhanced Douglas-Peucker (EDP) algorithm implements a set of enhanced spatial-temporal constraints to simplify the trajectory data. These constraints ensure that the essential properties of a trajectory be preserved through preserving critical points. Likewise, [18] analyzes the movement behaviors of nodes from the aspects of moving speeds, stop points, and moving directions, and then, a novel Trajectory Partition Method based on combined movement Features (TPMF) is proposed to partition the trajectories. TPMF first extracts the change points where the movement speeds of nodes are varied significantly and then extracts the stop points by detecting the speed variations of nodes. Finally, the Douglas-Peucker algorithm is applied to partition the subtrajectories according to the extracted change points and stop points. Lin et al. propose a one-pass error bounded trajectory simplification algorithm (OPERB) [19]. Based on a local distance checking method, OPERB maintains a directed line segment to approximate the buffered points and guarantees that the distance from the current point to the line segment is bounded. Reference [20] proposes a Joint Spatial-Temporal Trajectory Clustering Method (JSTTCM), where some spatial-temporal properties of the trajectories are exploited to cluster the trajectory segments. In [21], the hierarchical structure of the DP-based trajectory compression algorithm has been redesigned according to the GPU architecture and programming framework. It is able to significantly accelerate the compression of large-scale vessel trajectories while maintaining the required compression quality. Reference [22] proposes an unsupervised learning method which automatically extracts the low-dimensional features through a Convolutional Auto-Encoder (CAE). In particular, the informative trajectory images are first generated by remapping the raw vessel trajectories into two-dimensional matrices. Besides, a kind

of parallel algorithm utilizing Hopfield Neural Network is proposed in [23]. The proposed algorithm trajectory compression based on the Hopfield neural network (HNN-based) is a parallel algorithm, which evidently reduces the processing delay. The total compression error, called the integral square (ISE) between the origin trajectories and the compressed trajectories, is a sum of the squared Euclidean distances of trajectory segments. The objective of HNN-based algorithm is to find a subset of origin trajectories while minimizing the total compression error. Typically, the HNN-based algorithm employs a two-dimensional binary Hopfield neural network to locate and save the points in the compressed trajectory. The network consists of $N \times M$ mutually interconnected neurons. The rows and columns of the Hopfield network represent the points on the original trajectory and on the compressed trajectory, respectively. If the neuron state matrix achieves a stable state after updates, the total compression error is considered to be the minimum.

Most of the aforementioned works compress the trajectories spatially, and it is necessary to take into account the critical movement features such as the retrace points, moving speeds, and the directions. Moreover, a preferable trade-off between the compression ratio while the compression error is expected to be made as much as possible.

3. Problem Formulation

3.1. Trajectory Representation. A trajectory represents the path that a moving object travels over time. A spatial trajectory can be described either by the path geometry or by the sequential positions of the object.

Definition 1 (GPS trajectory). Generally, a GPS trajectory T consists of some spatial data (such as latitudes and longitudes). Besides, the timestamp t is always stored along with spatial data. A series of time-stamped position points (locations) form an ordered sequence, which is denoted by a tuple $p_n = \langle x_n, y_n, t_n \rangle$, where x_n and y_n represent the latitude and the longitude at time stamp t_n , respectively.

Definition 2 (trajectory segment). A trajectory segment, denoted by $\text{Tr}[1, n] = \{p_1, \dots, p_n\}$, is a sequence extracted from the original trajectory.

Definition 3 (compressed trajectory segment). A compressed trajectory segment, denoted by $\text{Tr}'[1, n] = \{ps_1, \dots, ps_m\}$, is the compression of $\text{Tr}[1, n] = \{p_1, \dots, p_n\}$. All points from a compressed trajectory segment are consecutive and contained in the original trajectory T .

Definition 4 (direction of trajectory segment line). A line segment in a trajectory segment $\text{Tr}[1, n]$ is expressed as a vector $\overrightarrow{p_i p_j}$, where $1 \leq i < j \leq n - 1$, the direction of $\overrightarrow{p_i p_j}$ is denoted by $\theta(\overrightarrow{p_i p_j})$, is calculated as the angle of an anti-clockwise rotation from the x -axis to the vector $\overrightarrow{p_i p_j}$. Figure 2 shows the case where $\theta(\overrightarrow{p_1 p_2}) = 1/4$ and the case where $\theta(\overrightarrow{p_3 p_4}) = 7\pi/4$.

FIGURE 2: An example of direction of trajectory segment line.

Definition 5 (maximum directional deviation). The index *maximum directional deviation* denotes the maximum of directional deviation of any two adjacent points $(\overrightarrow{p_s p_{s+1}})$ of a trajectory segment $\text{Tr}[1, n]$, where $i \leq s \leq j - 1$, and p_i, p_j represent the start point and end point of the compressed trajectory segment, respectively. $\Delta(\overrightarrow{p_i p_j})$ denotes the maximum angular difference, and the maximum directional deviation of a compressed segment $\overrightarrow{p_i p_j}$ is denoted by $\varepsilon(\overrightarrow{p_i p_j})$. We have the following:

$$\begin{aligned} \Delta(\overrightarrow{p_i p_j}) &= \max_{i \leq s \leq j-1} \left\{ \theta(\overrightarrow{p_s p_{s+1}}) \right\} - \min_{i \leq s \leq j-1} \left\{ \theta(\overrightarrow{p_s p_{s+1}}) \right\}, \\ \varepsilon(\overrightarrow{p_i p_j}) &= \min \left\{ \Delta(\overrightarrow{p_i p_j}), 2\pi - \Delta(\overrightarrow{p_i p_j}) \right\}. \end{aligned} \quad (1)$$

Definition 6 (compression ratio). Trajectory compression is aimed at extracting a set of sequential points $\text{Tr}'[1, n] = \{ps_1, \dots, ps_m\}$ from the trajectory segment $\text{Tr}[1, n]$, and $\text{Tr}'[1, n]$ can maintain the features of $\text{Tr}[1, n]$ as much as possible. The compression ratio CR is defined as $\text{CR} = m/n$, $n > m$, where n denotes the number of points in the original trajectory and m denotes the number of points in the compressed trajectory, respectively.

3.2. Error Measurement. Firstly, we introduce two types of position error metrics: Perpendicular Euclidean distance (PED) [5] and Time Synchronized Euclidean Distance (SED) [24], and then, we give some definitions for the error measurements.

3.2.1. Perpendicular Euclidean Distance. Given a trajectory segment $\text{Tr}[1, n]$ and its compressed representation $\text{Tr}'[1, n]$, the PED of $\text{Tr}'[1, n]$ with respect to a point $p_i(x_i, y_i, t_i)$ in $\text{Tr}[1, n]$ 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 closest point to p_i in $\text{Tr}'[1, n]$). If $\text{Tr}'[1, n]$ contains p_i , then $p'_i = p_i$.

For example, as shown in Figure 3(a), $p'_1 = p_1$ and $p'_4 = p_4$, where a trajectory containing $p_1, \dots, p_6$ is compressed by p_1, p_4 , and p_6 .

3.2.2. Time Synchronized Euclidean Distance. However, PED does not take into account the time dimension attached with

FIGURE 3: PED and SED errors are illustrated by $\text{Tr}[1, 6]$ and $\text{Tr}'[1, 6] = \{p_1, p_4, p_6\}$.

each trajectory. To this end, SED measures the distance of two points in original trajectory and compressed trajectory with the identical time stamps. With regard to the spatial errors, the SED between point $p_i(x_i, y_i, t_i)$ and its estimation $p'_i(x'_i, y'_i, t'_i)$ is denoted by $\text{SED}(p_i, p'_i)$, which is calculated as the distance from (x_i, y_i) to (x'_i, y'_i) . Thus, with regard to a trajectory segment $\text{Tr}[s, k]$, for any point $p_i(x_i, y_i, t_i)$, the estimation $p'_i(x'_i, y'_i, t'_i)$ is expressed as follows:

$$\begin{aligned} x'_i &= x_s + \frac{t_i - t_s}{t_k - t_s}(x_k - x_s), \\ y'_i &= y_s + \frac{t_i - t_s}{t_k - t_s}(y_k - y_s), \\ t'_i &= t_i. \end{aligned} \quad (2)$$

Figure 3(b) gives an example of SED, where the distance of $\overrightarrow{p_2 p'_2}$ denotes the SED of p_2 with regard to $\overrightarrow{p_1 p_4}$.

3.2.3. Speed Error. GPS data collected by positioning modules usually does not contain the speed information. Therefore, the speed is calculated based on the trajectory points and their relations, i.e., the speed can be obtained by the distance and the time interval between adjacent points. The speed between two adjacent points p_s and p_k is denoted by $\text{sp}(p_s, p_k)$, and $\text{sp}(p_s, p_k)$ is expressed as follows:

$$\text{sp}(p_s, p_k) = \frac{\text{dis}(p_s, p_k)}{t_s - t_k}, \quad (3)$$

where $\text{dis}(p_s, p_k)$ denotes the Euclidean distance between p_s and p_k . Likewise, if the points p_s and p_k are not adjacent, the speed between p_s and p_k is denoted by $\text{tsp}(p_s, p_k)$, and $\text{tsp}(p_s, p_k)$ is calculated as the average speed of traveling from p_s to p_k , i.e.,

$$\text{tsp}(p_s, p_k) = \frac{\sum_{i=0}^{k-s-1} \text{dis}(p_{s+i}, p_{s+i+1})}{t_s - t_k}. \quad (4)$$

Speed error [25] is a vital metric for various kinds of traffic applications. It measures the difference between the actual speed and the estimated speed. The speed error between p_s and p_k is denoted by $\text{SpdE}(p_s, p_k)$, which is written as follows:

$$\text{SpdE}(p_s, p_k) = |\text{sp}(p_s, p_k) - \text{tsp}(p_s, p_k)|. \quad (5)$$

For example, as show in Figure 3, $\text{sp}(p_1, p_4) = \text{dis}(p_1, p_4) / t_4 - t_1$, $\text{tsp}(p_1, p_4) = \text{dis}(p_1, p_2) + \text{dis}(p_2, p_3) + \text{dis}(p_3, p_4) / t_4 - t_1$, and $\text{SpdE}(p_1, p_4) = |\text{sp}(p_1, p_4) - \text{tsp}(p_1, p_4)|$.

3.2.4. Direction Preserving Velocity Error. In this paper, the velocity is different from speed in that the velocity reveals both travel speed and travel directions of the moving object. Suppose there are two adjacent points p_i and p_{i+1} , and $\overrightarrow{p_i p_{i+1}}$ represents a subsegment of the trajectory segment $\text{Tr}[1, n]$.

The velocity of $\overrightarrow{p_i p_{i+1}}$ is denoted by $V(\overrightarrow{p_i p_{i+1}})$, which is equal to the displacement in 2D Euclidean space divided by the time interval from points p_i to p_{i+1} :

$$V(\overrightarrow{p_i p_{i+1}}) = \frac{\text{dis}(p_i, p_{i+1})}{t_{i+1} - t_i}. \quad (6)$$

Besides, the segment $\overrightarrow{p_i p_{i+1}}$ also contains some direction information. The direction of the segment $\overrightarrow{p_i p_{i+1}}$ is marked as $\theta(\overrightarrow{p_i p_{i+1}})$. Thus, $V(\overrightarrow{p_i p_{i+1}})$ can be decomposed along two subvectors: $V(\overrightarrow{p_i p_{i+1}}) \cdot \cos \theta(\overrightarrow{p_i p_{i+1}})$ and $V(\overrightarrow{p_i p_{i+1}}) \cdot \sin \theta(\overrightarrow{p_i p_{i+1}})$, which are parallel to x -axis and y -axis, respectively. Then, $V(\overrightarrow{p_i p_j})$ can be represented by the point with the coordinate $(V_x(p_i p_j), V_y(p_i p_j))$ in the rectangular coordinate system, i.e.,

$$\begin{aligned} V_x(p_i p_j) &= |V(\overrightarrow{p_i p_j})| \cdot \cos \theta(\overrightarrow{p_i p_j}), \\ V_y(p_i p_j) &= |V(\overrightarrow{p_i p_j})| \cdot \sin \theta(\overrightarrow{p_i p_j}). \end{aligned} \quad (7)$$

Likewise, the coordinate of average travel velocity vector is with the similar form of average velocity. The average travel velocity $|V'(\overrightarrow{p_i p_j})|$ can be represented by the point with the coordinate $(V'_x(p_i p_j), V'_y(p_i p_j))$:

$$\begin{aligned} V'_x(p_i p_j) &= |V(\overrightarrow{p_i p_j})| \cdot \cos \theta(\overrightarrow{p_i p_j}), \\ V'_y(p_i p_j) &= |V(\overrightarrow{p_i p_j})| \cdot \sin \theta(\overrightarrow{p_i p_j}). \end{aligned} \quad (8)$$

Especially, note that the velocity component $|V'(\overrightarrow{p_i p_j})|$ is equal to the value of $\text{tsp}(p_i p_j)$. Let $\theta_{\text{ave}}(\overrightarrow{p_i p_j})$ denote the average direction of the trajectory segment $\overrightarrow{p_i p_j}$, and $\theta_{\text{ave}}(\overrightarrow{p_i p_j})$ is expressed as follows:

$$\begin{aligned} \theta_{\text{ave}}(\overrightarrow{p_i p_j}) \\ = \frac{\sum_{s=0}^{j-i-1} \theta(p_{i+s}, p_{i+s+1}) * \text{dis}(p_{i+s}, p_{i+s+1}) / \sum_{s=0}^{j-i-1} \text{dis}(p_{i+s}, p_{i+s+1})}{j-i}. \end{aligned} \quad (9)$$

Moreover, we introduce the Direction Preserving Velocity Error (DPVE). Given a trajectory T and its compressed trajectory $\text{Tr}[1, n] = \{ps_1, \dots, ps_m\}$, the compression velocity error of the segment $\text{Tr}[ps_k, ps_{k+1}]$ is denoted by $\text{DPVE}(ps_k, ps_{k+1})$, representing the standardized Euclidean distance between two velocity vectors $V(\overrightarrow{ps_k ps_{k+1}})$ and $V'(\overrightarrow{ps_k ps_{k+1}})$:

$$\text{DPVE}(ps_k, ps_{k+1}) = \text{dist}\left(V(\overrightarrow{ps_k ps_{k+1}}), V'(\overrightarrow{ps_k ps_{k+1}})\right), \quad (10)$$

where $\text{dist}(V(\overrightarrow{ps_k ps_{k+1}}), V'(\overrightarrow{ps_k ps_{k+1}}))$ is obtained by the following:

$$\begin{aligned} \text{dist}\left(V(\overrightarrow{ps_k ps_{k+1}}), V'(\overrightarrow{ps_k ps_{k+1}})\right) \\ = \sqrt{\frac{(V_x(p_i p_j) - V_x'(p_i p_j))^2}{m_1} + \frac{(V_y(p_i p_j) - V_y'(p_i p_j))^2}{m_2}}, \end{aligned} \quad (11)$$

where ps_k and ps_{k+1} denote two adjacent points in the compressed point set and m_i denotes the average of the i -th dimension of $V(\overrightarrow{ps_k ps_{k+1}})$ and $V'(\overrightarrow{ps_k ps_{k+1}})$.

4. Algorithm

In this section, we present a velocity-preserving trajectory compression algorithm based on retrace point detection (VPTC-RP). In VPTC-RP, the semantic meanings regarding the movement randomness are taken into account, while VPTC-RP exploits the retrace points and the velocity information to compress the trajectories.

4.1. Retrace Point Detection. In our work, we note that the sharp direction changes may frequently occur especially in the pedestrian trajectories. Moreover, the sharply changed direction of a point may indicate that the moving object is retracing of towards the previous points. In real scenes, some trajectory points often indicate that the pedestrian nodes pass through the landmarks with high weights, and the trajectories around these landmarks are with distinct features. According to this phenomenon, these trajectory points in the trajectories are much more vital than other trajectory points.

Thus, we define the term of retrace point: a node retraces towards a previous point, and several subsequent points are very close to the previous point. The previous point is referred to as an anchor point. To form the retracement, several points are probably located around the anchor point (e.g., someone lingers around some goods in a shopping mall). The points falling into such retrace region are taken

FIGURE 4: An example of Theorem 7.

as the retrace points, and the number of the retrace points is typically small.

Then, we introduce an index *retrace stability* to measure the proximity of the subsequent points with the anchor points, through which the retrace points can be found. The retrace points must be accompanied with the maximum directional deviation of trajectory segment which is larger than $\pi/2$. Therefore, the retrace point detection algorithm should check the maximum directional deviation of the trajectory segment. A theorem and a lemma are given for finding the retrace points:

Theorem 7. *An angle θ inscribed in a circle is half of the central angle 2θ that subtends the same arc on the circle.*

Proof. The angle will not be changed as its vertex has been moved to different positions on the circle, as proven in [26]. $\square$

According to Theorem 7, an example is shown in Figure 4, where p_1 , p_2 , and p_3 are located on the same circle, and there is $\angle p_1 p_3 p_2 = \angle p_1 p_3 p'_2$. Moreover, Lemma 8 can be derived from Theorem 7.

Lemma 8. *With regard to any arc on the circle, if a point located outside the region enclosed by the arc and the two endpoints of the arc, the angle which is formed by the point and the two endpoints is smaller than the inscribed angle. Similarly, the angle which is formed by the point inside the circle is larger than the inscribed angle.*

Proof. Given an arc $(\widehat{p_1 p_3})$ and the center of the circle O , as depicted in Figure 4, $\angle p_1 p_3 p_2$ denotes the inscribed angle of $\widehat{p_1 p_3}$. The point p'_3 and point p'_3 are located on the half of the line Op_3 . In $\triangle p_2 p_3 p'_3$, according to exterior angle theorem, the exterior angle $\angle p_2 p_3 O$ is equal to $\angle Op'_3 p_2 + \angle p_3 p_2 p'_3$. Likewise, $\angle p_1 p_3 O$ is equal to $\angle Op'_3 p_1 + \angle p_3 p_1 p'_3$. Furthermore, we obtain that

$$\begin{aligned}
\angle p_1 p_3 O + \angle p_2 p_3 O &> \angle p_3 p_2 p'_3 + \angle p_3 p'_3 O + \angle p_1 p''_3 p_3 + \angle p_1 p_3 p'_3, \\
\angle p_1 p_3 p_2 &> \angle p_2 p_3 p'_1 + \angle p'_3 p_1 p_3 + \angle p'_3 p_3 p_2.
\end{aligned} \tag{12}$$

□

Thus, the point in same inscribed angle can form an egg-shape on both sides of a line segment due to the symmetry. To clarify the algorithm, we provide the following definitions:

Definition 9 (retrace region). Given a line segment and an inscribed angle threshold, two arcs formed by the point in the inscribed angle of the line segment generate an egg-shape area, and the egg-shaped area enclosed by the dotted line is termed a retrace region. As show in Figure 4, the shadowed area represents a retrace region.

Definition 10 (retrace stability). Given an anchor point, a float point, and the line segment between the two points, the retrace stability S_{re} on the subsequent points is determined by two variables:

- (1) The distance ratio δ , which is defined as the ratio of two distances (ratio between the distance from the subsequent point to a float point and the distance to an anchor point)

$$\delta = \frac{\text{dis}(p_{fl}, p_{sub})}{\text{dis}(p_{an}, p_{sub})}, \tag{13}$$

where $\text{dis}(p_{fl}, p_{sub})$ and $\text{dis}(p_{an}, p_{sub})$ denote the distance from the subsequent point to an anchor point and the distance to a float point, respectively. Then, we explore the impact of δ on the relationship between the subsequent point and anchor point. To clarify this issue, we provide Theorem 11.

Theorem 11. When the distance ratio δ is larger, the subsequent point is closer to the anchor point.

Proof. Given a rectangular coordinate system, the coordinate of a float point $p_{fl}(-a, 0)$ and the coordinate of an anchor point $p_{an}(0, 0)$ taken as the origin are shown in Figure 5. Suppose the coordinate of subsequent point is denoted by $p_{sub}(x, y)$, and there is $\text{dis}(p_{fl}, p_{sub})/\text{dis}(p_{an}, p_{sub}) = k (k > 0, k \neq 1)$. Thus, we obtain the following equations:

$$\begin{aligned}
\frac{\sqrt{(x+a)^2 + y^2}}{\sqrt{x^2 + y^2}} &= k, \\
(k^2 - 1)(x^2 + y^2) + 2ax - a^2 &= 0.
\end{aligned} \tag{14}$$

□

We find that p_{sub} at the same δ moves on the circle with the center $(ak^2/k^2 - 1, 0)$ and the radius $R = ak/k^2 - 1$.

FIGURE 5: An example of all p_{sub} at the same δ .

FIGURE 6: An example of retrace points (p_5 , p_6 , and p_7).

Besides, note that $ak^2/k^2 - 1$ and $ak/k^2 - 1$ are monotone:

$$\frac{k^2}{k^2 - 1} = \frac{k^2 + 1 - 1}{k^2 - 1} = 1 + \frac{1}{k^2 - 1}. \tag{15}$$

Therefore, $ak^2/k^2 - 1$ is monotonically decreasing in the domain.

$k/k^2 - 1$ is an odd function, and the domain is $(0, 1) \cap (1, +\infty)$. The differential coefficient of $k/k^2 - 1$ is expressed as follows:

$$\frac{k}{k^2 - 1}' = -\frac{k^2 + 1}{(k^2 - 1)^2} < 0. \tag{16}$$

Similarly, $ak/k^2 - 1$ is monotonically decreasing in the domain.

Thus, the circle generated the subsequent point at a larger distance ratio δ will be involved in that at a smaller δ , i.e., the subsequent point is closer to the anchor point.

Moreover, when $k = 1$, the subsequent point is on the perpendicular bisector of the line segment $p_{fl} p_{an}$. Obviously, when $k > 1$, the subsequent point is on the right of perpendicular bisector. Therefore, with the increase of δ falling into the interval $(0, +\infty)$, the subsequent point is closer to the anchor point.

Input: trajectory $Tr[1, n]$, ratio threshold δ_{th} , inscribed angle threshold θ_{th} .
Output: retrace points RP.

1. $i=1, j=3, RP = \Phi$.
2. while $j < n$ do
3. Calculate Maximum direction deviation $\varepsilon(\overrightarrow{p_i p_j})$;

The selected pair of the anchor point p_s and float point p_e (with the maximum direction deviation);

4. if $\varepsilon(\overrightarrow{p_i p_j}) > \pi/2$ then
5. Calculate inscribed angle θ and ratio threshold δ ;
6. if $\theta \geq \theta_{threshold}$ and $\delta \geq \delta_{threshold}$ then
7. $RP \cup p_e$;
8. $r = \text{dis}(p_e, p_s)$;
9. while $e+1 \leq n$
10. $k=e+1$;
11. if $\text{dis}(p_e, p_k) \leq r$ then
12. $RP \cup p_k$;
13. $e=e+1$;
14. else:
15. break;
16. end if
17. end while
18. else:
19. $i=e$;
20. $j=i+2$;
21. end if
22. end if
23. else
24. $j=j+1$;
25. end if
26. end while

Return retrace points set RP.

ALGORITHM 1: Retrace point detection process.

Input: trajectory $Tr[1, n]$, ratio threshold δ_{th} , inscribed angle threshold θ_{th} , velocity error μ .
Output: compressed trajectory $Tr'[1, n]$.

1. $i = 1, j=3, Tr[1, n] = \Phi$.
2. while $j < n$ do
3. Calculate Maximum direction deviation $\varepsilon(\overrightarrow{p_i p_j})$;

selected pair of anchor point p_s and float point p_e (with the maximum direction deviation);

4. if $PRD(i, j, r, \theta) \neq \Phi$
5. $Tr'[1, n] \cup PRD(i, j, r, \theta)$;
6. $i = \text{the last index of set } PRD(i, j, r, \theta)$;
7. $j=i+2$;
8. else:
9. Calculate velocity error in buffer λ ;
10. if $\lambda \geq \mu$ then
11. $Tr'[1, n] \cup p_j$;
12. $i=j$;
13. $j=i+2$;
14. else:
15. $j=j+1$;
16. end if
17. end if
18. end while

Return compressed trajectory $Tr'[1, n]$.

ALGORITHM 2: Trajectory velocity-preserving process.

(2) The inscribed angle threshold θ of the line segment $\overrightarrow{p_{an}p_{fl}}$ is defined as

$$\theta = \arccos \left\{ \frac{\text{dis}(p_{an}, p_{sub})^2 + \text{dis}(p_{fl}, p_{sub})^2 - \text{dis}(p_{fl}, p_{an})^2}{2 * \text{dis}(p_{an}, p_{sub}) * \text{dis}(p_{fl}, p_{sub})} \right\} \quad (17)$$

Then, $S_{re}(p_{an}, p_{fl})$ is expressed as (δ, θ) .

Definition 12 (retrace point). If $S_{re}(p_{an}, p_{fl}) \geq (\delta_{th}, \theta_{th})$, where δ_{th} and θ_{th} denote the preset thresholds of δ and θ , respectively, then the subsequent point is considered to be located in the retrace region. We use the line between the subsequent point and the anchor point as the radius to generate a circle (including the circle periphery) where each point falling into this region is taken as a retrace point.

An example of retrace points is given in Figure 6, where p_5 has a trend of retracing towards p_1 . Hence, p_1 is regarded as an anchor point, and p_4 is a float point. If $S_{re}(p_1, p_4) \geq (\delta_{th}, \theta_{th})$, thus p_5 is a retrace point. Then, in the circle region with radius $\overrightarrow{p_1p_5}$, p_6 and p_7 are taken as retrace points.

The pseudocode of RPD is given in Algorithm 1.

The retrace point detection steps are described as follows:

Step 1. The process starts by defining a segment between the first point p_1 and the third point p_3 in $\text{Tr}[1, n]$, where p_1 is taken as the first anchor point p_i and p_3 is taken as the first float point p_j . With regard to each segment between the anchor point and the float point $\overrightarrow{p_i p_j}$, the direction deviation $\varepsilon(\overrightarrow{p_i p_j})$ is calculated, and then, the subsequent points in $\text{Tr}[1, n]$ will play the roles of p_i and p_j , and the direction deviations will be processed as well.

Step 2. The maximum direction deviation in $\text{Tr}[1, n]$ is selected, and the selected pair of anchor point and float point (with the maximum direction deviation) is recorded as p_s and p_e , respectively. Especially, if $\varepsilon(\overrightarrow{p_i p_j})$ is larger than $\pi/2$, then p_i and p_j will be directly recorded as p_s and p_e , which is due to the fact that this derivation indicates a trend of retracing towards p_s .

Step 3. If the retrace stability of p_e is larger than a preset threshold, then a retrace point detection is performed to find the retrace points ($p_r \in \text{RP}$), so that the inequality $\text{dis}(p_s, p_r) \leq \text{dis}(p_s, p_e)$ is satisfied.

Step 4. The above steps will be executed until all points in $\text{Tr}[1, n]$ have been processed, and the retrace points have been found.

4.2. Trajectory Compression. In VPTC-RP, the retrace points are first deleted based on the results of the retrace point detection, and then, the velocity information is preserved as much as possible for the trajectory compression. Direc-

TABLE 1: Trajectory details.

TraID	Raw trajectories	Raw points
#tra1	200	232,370
#tra2	150	148,582
#tra3	100	68,844

tion Preserving Velocity Error (DPVE) is proposed to preserve the information of velocity (speed and direction) which is represented by the standardized Euclidean distance between two velocity vectors, and this measurement can reserve the speed information preferably. Moreover, the retrace points are some redundant trajectory points where the directions of nodes change sharply. Therefore, by adopting the above mechanisms, the information of speed and direction can be preserved in the compressed trajectories simultaneously.

The pseudocode of VPTC-RP is given in Algorithm 2:

Step 5. The retrace points are found according to Steps 1–4. With regard to each pair of anchor point and float point, two cases are discussed: (a) If $\varepsilon(\overrightarrow{p_i p_j}) > \pi/2$, suppose the point p_r is the last point in the current retrace point set, then the p_s of p_r will be added into the compressed trajectory. (b) If $\varepsilon(\overrightarrow{p_i p_j}) \leq \pi/2$ or the retrace points cannot be found, DPVE(p_i, p_j) is calculated and the result is marked as λ . If λ is larger than a preset threshold, then p_j is added into the compressed trajectory $\text{Tr}'[1, n]$.

Step 6. When all points in the trajectory have been processed and compressed, the compressed points will be output.

According to the steps of VPTC-RP, VPTC-RP adapts the sliding-window strategy. If VPTC-RP fails to find the retrace points, the process of VPTC-RP is similar to the sliding-window algorithm. Therefore, the worst-case time complexity of VPTC-RP is written as $O(n\beta)$, where β is the maximum buffer size and n is the number of points in the original trajectory.

5. Experiment Analysis

5.1. Simulation Settings. In this section, we conduct an extensive simulation study of VPTC-RP. We evaluate VPTC-RP based on the Geolife dataset. All simulations are implemented on a computer equipped with Windows 10, 1.60GHz CPU and 8 GB memory. The trajectory compression algorithms are realized by Python language.

5.2. Dataset. GPS trajectory dataset is cited from the (Microsoft Research Asia) Geolife project [27], which collects the trajectories of 182 users during five years (from April 2007 to August 2012). This dataset contains 17,621 trajectories with a total distance of 1,292,951 kilometers and a total duration of 50,176 hours, which recorded a broad range of users' outdoor movements. These trajectories are recorded by different GPS loggers and GPS phones and have a variety

FIGURE 7: Number of retrace points vs. distance ratio threshold. (a) #tra1; (b) #tra2; (c) #tra3.

of sampling rates. Especially, 91.5% of the trajectories are logged in a dense representation, e.g., every 1~5 seconds or every 5~10 meters per point.

5.3. Parameter Settings. After filtering some abnormal data, we randomly select 450 trajectories for the simulations. These trajectories are collected in one day and from 20 different users, which are enough to exhibit the behaviors of nodes. The total number of trajectory points is 449,796. For the effect of parameter variations, we divided all the trajectories into three groups. The details of each group are shown in Table 1.

Firstly, we observe the impacts of the inscribed angle threshold θ and the distance ratio δ on the number of retrace points, and the simulation results are given in Figure 7.

From Figures 7(a)–7(c), we can observe that the curves decrease rapidly with the distance ratio threshold increasing from 1.1 to 1.4, which is attributed to the fact that in VPTC-RP, more points are treated as the retrace points when a smaller distance ratio threshold δ is set. When $\delta \geq 1.4$, the

curves descend slowly, especially when $1.5 \leq \delta \leq 1.9$, the curves remain almost stable, which indicates that the number of retrace points fluctuates very slightly, and the reason is that the trajectory points become denser in retrace regions when $1.5 \leq \delta \leq 1.9$, and thus, the value of δ should be selected from the interval [1.4, 1.6]. Moreover, when $\delta = 1.4$ or $\delta = 1.6$, the number of extracted retrace points changes obviously compared with that when $\delta = 1.5$. Hence, we set $\delta_{th} = 1.5$ in the following simulations.

As shown in Figures 8(a)–8(c), the number of retrace points remains almost unchanged when $0.2\pi \leq \theta \leq 0.4\pi$, due to the fact that when the value of θ is small, i.e., it does not have an obvious impact on the number of retrace points when $\theta \in [0.2\pi, 0.4\pi]$. Besides, we can find that when $0.4\pi \leq \theta \leq 0.8\pi$, the number of retrace points decreases rapidly with a larger value of θ . In addition, when $0.4\pi \leq \theta \leq 0.8\pi$, the number of extracted retrace points is not large enough, and thus, we set $\theta_{th} = 0.2\pi$ for extracting more retrace points. The main simulation parameters are provided in Table 2.

FIGURE 8: Number of retrace points vs. inscribed angle threshold. (a) #tra1; (b) #tra2; (c) #tra3.

We will compare VPTC-RP with other algorithms in terms of several evaluation metrics:

(i) *The Compression Error between the Compressed Trajectories and the Original Trajectories.* Based on the compression error, we define a new metric termed *average velocity error* to measure the velocity error between two adjacent compressed points. For example, with regard to a trajectory $\text{Tr}[1, n]$ and its compressed representation $\text{Tr}'[1, n]\{ps_1, \dots, ps_m\}$, the expression of average velocity error is written as follows:

$$\text{AVE}(\text{Tr}[1, n], \text{Tr}'[1, n]) = \frac{\sum_{k=1}^{m-1} \text{DPVE}(ps_k, ps_{k+1})}{m-1}, \quad (18)$$

where ps_n denotes the n -th compressed point in the compressed trajectory point set and $\text{DPVE}(ps_k, ps_{k+1})$ represents the velocity error between ps_k and ps_{k+1} .

TABLE 2: Simulation parameters (CR = 50%).

Parameters	Description	Values
δ_{th}	Distance ratio threshold	1.5
θ_{th}	Inscribed angle threshold	0.2π
DPVE	Velocity error	0.35

TABLE 3: Parameter settings of other algorithms (CR = 50%).

Algorithms	SED/PED	λ
DP	0.68 m	N/A
OPW	1.15 m	N/A
SQUISH-E	0.46 m	5

(ii) *The Running Time Which Represents the Execution Time of Trajectory Compression Process.* It measures the time complexity.

FIGURE 9: Number of retrace points vs. compression ratio. (a) $\delta = 1.1$ and $\theta = 0.2\pi$; (b) $\delta = 1.2$ and $\theta = 0.2\pi$; (c) $\delta = 1.3$ and $\theta = 0.5\pi$.

(iii) *The Number of Retrace Points Which Are Preserved Mistakenly.*

5.4. Number of Residual Retrace Points. In this section, VPTC-RP is compared with four algorithms (DP, SQUISH-E, OPW, and HNN-based algorithm). The parameter settings of DP, SQUISH-E, and OPW are given in Table 3, and the following simulations are run on the trajectory #tra1. Besides, HNN-based algorithm employs a Hopfield network consists of $N \times M$ mutually interconnected neurons. The rows and columns of the Hopfield network represent the points on the original trajectory and the positions of points on the compressed trajectory, respectively. For example, a neuron $V_{x,i}$ in a firing state indicates that the point p_x on the original trajectory is the i -th point on the compressed trajectory. The state of a neuron in each column (except for column 1 and column M) with the maximum input compared with other neurons on the same column is set to 1, and other neurons on this column are set to 0. Firstly, we observe the number of residual retrace points after the compressions, and the simulation results are reported in Figure 9.

As shown in Figure 9, the number of residual retrace points is observed under different compression ratios. The curve of SQUISH-E is lower than those of other algorithms when the compression ratio is smaller than 50%, and the curve of SQUISH-E is higher than those of DP, HNN-based algorithm, and OPW when the compression ratio is larger than 50%. Particularly, the curves of DP and OPW are always very close to each other. Moreover, HNN-based algorithm obtains the smallest number of residual retrace points among other algorithms when the compression ratio is larger than 50%. This is attributed to the fact that SQUISH-E can compress the trajectories according to the SED measurement which considers the temporal information, and hence, the number of residual retrace points is significantly reduced. When the compression ratio is set very large, DP can compress some dense points in a small region where the spatial error is extremely low. Although HNN-based algorithm can achieve the optimal results, the trajectories are compressed under squared Euclidean distances which ignore temporal information. Thus, SQUISH-E outperforms HNN-based algorithm when the compression

FIGURE 10: Running time comparison.

ratio is very small. These phenomena imply that many retrace points have been ignored when the trajectories are compressed.

5.5. Algorithm Comparisons. In Figure 10, the running time of different algorithms is observed. The running time of SQUISH-E is always shorter than those of DP, OPW, VPTC-RP, and HNN-based algorithm. This is due to the use of a priority queue in SQUISH-E which enables the fast removals of points. Besides, VPTC-RP consumes a shorter running time than those of DP and OPW, when the compression ratio is smaller than 30%. This is attributed to the fact that VPTC-RP compresses the trajectories based on the retrace points, and hence, the number of computations is significantly reduced, especially when there are more retrace points in the original trajectories. The running time of HNN-based algorithm is much longer than others, because the number of iterations is extremely large.

Figure 11 illustrates that VPTC-RP achieves a smaller average velocity error than those of DP, OPW, SQUISH-E, and HNN-based algorithm, and this is because VPTC-RP compresses the trajectories by exploiting the velocity information, and it preserves the trajectory velocity as much as possible. When the compression ratio is larger than 40%, these curves are very close to each other, and the reason is that lots of points must be deleted from the initial trajectories.

Besides, note that the average velocity error of OPW is worst when the compression ratio is smaller than 40%, and this is because OPW does not take into account the velocity information and the semantic meanings of trajectories, and thus, OPW achieves a smaller compression ratio. In addition, it can be found that the average velocity error of HNN-based algorithm outperforms DP, OPW, and SQUISH-E, when the compression ratio is larger than 50%. This phenomenon is attributed to the fact that HNN-based algorithm is a meta-heuristic algorithm.

FIGURE 11: Average velocity error comparison.

Therefore, VPTC-RP makes a preferable tradeoff between the running time and the average velocity error through detecting the retrace points which include the velocity information and the semantic meanings.

6. Conclusion

In this paper, we investigate the problem of compressing the trajectories based on the retrace point detection. We define the retrace points which denote the positions where the moving directions change sharply. After the detection of retrace points, VPTC-RP (velocity-preserving trajectory compression algorithm based on retrace point detection) compresses the trajectories under DPVE. VPTC-RP can preserve both the velocity information and the semantic meanings of the trajectories as much as possible. In VPTC-RP, the adopted Direction Preserving Velocity Error (DPVE) focuses on preserving velocity information, which could fail to capture some information such as position information. Besides, when the interval between the two trajectory points is extremely long, the retrace point detection could fail as well.

Future research will take advantage of the knowledge regarding the real road networks to further improve the detection accuracy of the retrace points. In addition, we will investigate the classifications of transportation modes based on the obtained compressed trajectories.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Disclosure

This work is a significant extension of our early work which has been published in IEEE ICCT 2020.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research is supported by the National Natural Science Foundation of China under Grant Nos. 61872191 and 61872193, the National Key R&D Program of China No. 2019YFB2101700, and the Six Talents Peak Project of Jiangsu Province under Grant No. 2019-XYDXX-247.

References

- [1] Y. Takeuchi and M. Sugimoto, "CityVoyager: an outdoor recommendation system based on user location history," in *Proceedings of the 3rd International Conference on Ubiquitous Intelligence and Computing*, pp. 625–636, Wuhan, China, 2006.
- [2] L. Chen, M. Lv, and G. Chen, "A system for destination and future route prediction based on trajectory mining," *Pervasive and Mobile Computing*, vol. 6, no. 6, pp. 657–676, 2010.
- [3] Y. Yu, J. Kim, K. Shin, and G. S. Jo, "Recommendation system using location-based ontology on wireless internet: an example of collective intelligence by using 'mashup' applications," *Expert Systems with Applications*, vol. 36, no. 9, pp. 11675–11681, 2009.
- [4] P. Sun, S. Xia, G. Yuan, and D. Li, "An overview of moving object trajectory compression algorithms," *Mathematical Problems in Engineering*, vol. 2016, Article ID 6587309, 13 pages, 2016.
- [5] 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, 2010.
- [6] S. Reddy, M. Mun, J. Burke, D. Estrin, M. Hansen, and M. Srivastava, "Using mobile phones to determine transportation modes," *ACM Transactions on Sensor Networks*, vol. 6, no. 2, pp. 1–27, 2010.
- [7] C. Lin, C. Hung, and P. Lei, "A velocity-preserving trajectory simplification approach," in *2016 Conference on Technologies and Applications of Artificial Intelligence*, pp. 58–65, Hsinchu, Taiwan, 2016.
- [8] J. J.-C. Ying and J.-H. Su, "On velocity-preserving trajectory simplification," in *Asian Conference on Intelligent Information and Database Systems*, pp. 241–250, Da Nang, Vietnam, 2016.
- [9] F. Schmid, K. F. Richter, and P. Laube, "Semantic trajectory compression advances," in *International Symposium on Spatial and Temporal Databases*, Springer, Berlin, Heidelberg, 2009.
- [10] Y. Ye, Y. Zheng, Y. Chen, J. Feng, and X. Xie, "Mining individual life pattern based on location history," in *Proceedings of the 2009 Tenth International Conference on Mobile Data Management: Systems, Services and Middleware*, pp. 18–20, Taipei, Taiwan, 2009.
- [11] A. Chen and L. Liu, "An online trajectory compression based on retrace point detection," in *Proceedings of the IEEE 20th International Conference on Communication Technology (IEEE ICCT 2020)*, Nanning, China, 2020.
- [12] D. Douglas and T. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *The International Journal for Geographic Information and Geovisualization*, vol. 10, no. 2, pp. 112–122, 1973.
- [13] E. Keogh, S. Chu, D. Hart, and M. Pazzani, "An online algorithm for segmenting time series," in *Proceedings 2001 IEEE International Conference on Data Mining*, pp. 289–296, San Jose, CA, USA, 2001.
- [14] J. Muckell, J. 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 and Applications*, pp. 1–8, Washington, DC, USA, 2011.
- [15] J. Muckell, P. W. Olsen Jr., 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.
- [16] M. Yang, X. Yan, X. Zhang, and X. Li, "Constrained trajectory simplification with speed preservation," *Cartography and Geographic Information Science*, vol. 47, no. 2, pp. 110–124, 2019.
- [17] H. Qian and Y. Lu, "Simplifying GPS trajectory data with enhanced spatial-temporal constraints," *ISPRS International Journal of Geo-Information*, vol. 6, no. 11, 2017.
- [18] J. Tang, L. Liu, and J. Wu, "A trajectory partition method based on combined movement features," *Wireless Communications and Mobile Computing*, vol. 2019, Article ID 7803293, 13 pages, 2019.
- [19] X. Lin, S. Ma, H. Zhang, T. Wo, and J. Huai, "One-pass error bounded trajectory simplification," *Proceedings of the VLDB Endowment*, vol. 10, no. 7, pp. 841–852, 2017.
- [20] J. Tang, L. Liu, J. Wu, J. Zhou, and Y. Xiang, "Trajectory clustering method based on spatial-temporal properties for mobile social networks," *Journal of Intelligent Information Systems*, vol. 56, no. 1, pp. 73–95, 2021.
- [21] Y. Huang, Y. Li, Z. Zhang, and R. W. Liu, "GPU-accelerated compression and visualization of large-scale vessel trajectories in maritime IoT industries," *IEEE Internet of Things Journal*, vol. 7, no. 11, pp. 10794–10812, 2020.
- [22] M. Liang, R. W. Liu, S. Li, Z. Xiao, X. Liu, and F. Lu, "An unsupervised learning method with convolutional auto-encoder for vessel trajectory similarity computation," *Ocean Engineering*, vol. 225, p. 108803, 2021.
- [23] C. Chen, C. Xu, C. Xu, D. Liu, and L. Tian, "Trajectory data compression using Hopfield neural network," in *2018 International Conference on Computational Science and Computational Intelligence (CSCI)*, pp. 186–190, Las Vegas, NV, USA, 2018.
- [24] N. Meratnia and R. A. de By, "Spatiotemporal compression techniques for moving point objects," in *Advances in Database Technology - EDBT 2004*, pp. 765–782, Springer, Berlin, Heidelberg, 2004.
- [25] P. Hao, C. Yao, Q. Meng, X. Yu, and X. Li, "A heading maintaining oriented compression algorithm for GPS trajectory data," *Informatica*, vol. 30, no. 1, pp. 33–52, 2019.
- [26] M. B. Roscoe, "Discovering the inscribed angle theorem," *The Mathematics Teacher*, vol. 105, no. 7, p. 514, 2012.
- [27] 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. 32–40, 2010.