



# A New Trajectory Reduction Method for Mobile Devices Operating Both Online and Offline

Samet Diri<sup>1</sup> · Mehmet Yildirim<sup>2</sup>

Received: 1 March 2023 / Accepted: 11 March 2024  
© The Author(s) 2024

## Abstract

Highly accurate location data have become essential in nearly all contemporary global applications, including but not limited to route planning, processing traffic data, identifying common routes, map matching, and enhancing agricultural productivity. However, the abundance of unnecessary and redundant data leads to various challenges, especially concerning storage, processing, and transmission. Despite the existence of numerous studies aimed at addressing these GNSS data management challenges, the reduction problem is either partially resolved, or enhancements are made to existing solutions in nearly all of them. In this study, a novel reduction method is introduced, offering both a high reduction rate and accuracy, suitable for operation on mobile devices in both offline and online modes. The proposed method uses windowing with reference points during the decision phase to decrease the number of points. By utilizing the angle and its threshold between the decision point and reference points, we achieved a method characterized by low algorithmic complexity and a high reduction rate, suitable for online operation on mobile devices. Experiments and comparisons revealed that the proposed method had a 91.01% reduction in GNSS data which is 7.73% lower, a  $5.8744e-04$  RMSE error which is  $2e+7$  times better, and a 14.54 ms running time which is 25% faster than RDP algorithm. The results indicate that incorporating the proposed method into current methodologies could be beneficial, particularly in scenarios where real-time, high-precision location data are essential.

**Keywords** Line simplification · Trajectory compression · GNSS data · Douglas–Peucker algorithm · Mobility data

## 1 Introduction

Technological advances over the years have also enabled the production of transportation that are increasingly powerful, efficient, and facilitate transportation. In addition, developments in the field of communication have given a new direction to mobility. One of the most important developments is satellite systems. Satellite systems can be used individually or as a set, depending on their intended use. The Global Navigation Satellite System (GNSS) is a generic term that describes any constellation. GNSSs provide services on a global or regional basis. These are positioning, navigation,

and timing (PNT) services [1–3]. Currently, China has BeiDou, the European Union’s Galileo, Russia’s GLONASS, the United States’ GPS, India’s IRNSS, and Japan’s QZSS. All of them are actively serving both regionally and globally.

The free, open, and reliable nature of GNSSs and the advances in obtaining position information using satellites and mobile computing techniques have enabled the development of hundreds of applications that affect today’s modern life and are an integral part of it. Using these applications, a considerable amount of trajectory data is generated, representing the mobility of various moving objects such as humans, animals, land, sea, and aircraft [7–12].

The trajectory of a moving object is a successive time sequence of location points [10]. A trajectory is usually represented as a series of line segments connecting successive pairs of locations observed for each object (see Fig. 1). The trajectory  $T$  of a moving object on the Euclidean plane is an infinite series of time-stamped positions.

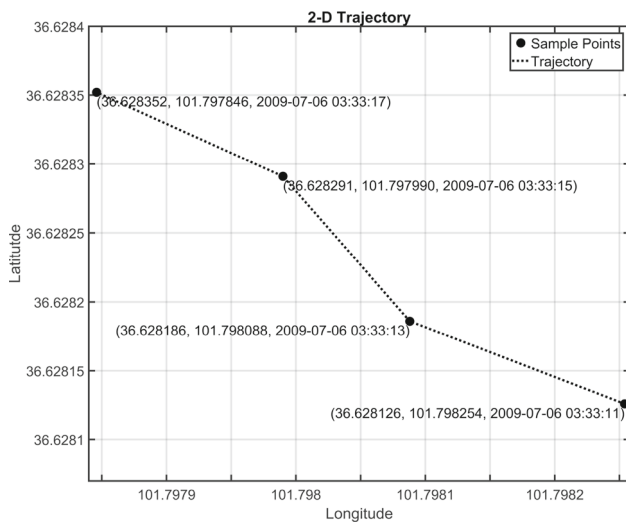
That is, sets of values  $(id, t_i, x_i, y_i)$ ,  $i \in \{1, \dots, N, \dots\}$  where  $t_i$  corresponds to the timestamp referring to the instantaneous time for an object defined as  $(x_i, y_i)$  points [10].

✉ Samet Diri  
samet.diri@kocaeli.edu.tr  
Mehmet Yildirim  
myildirim@kocaeli.edu.tr

<sup>1</sup> Computer Research Centre, Kocaeli University, Umuttepe, 41001 İzmit, Kocaeli, Turkey

<sup>2</sup> Information System Engineering, Kocaeli University, Umuttepe, 41001 İzmit, Kocaeli, Turkey





**Fig. 1** Sample trajectory data from the 20090706033310.plt file in the Geolife [4–6] dataset

Unlike a trajectory, a path is the route between two points regardless of time. Here, the main difference between the path and the trajectory is that the trajectory contains a timestamp [13].

In addition to location information such as latitude, longitude, and altitude, trajectories contain multiple values and hemispheres where data is taken, as well as satellite information. Due to frequent sampling by numerous different applications, extra and powerful hardware is needed to process the large amount of data obtained. Moreover, a large volume of disk space for storage, high bandwidth for transferring data, and more energy than necessary during the operation are required for all these systems [7, 10, 14, 15]. This situation causes an increase in the large amount of data currently stored. More methods and applications have been developed in different fields, such as data processing, reduction, and storage, to cope with the increasing amount of data and use it wisely [2, 3, 16]. These developed methods are used not only in big data processing and data mining but also in many other applications such as image processing, geographic information systems (GISs), and computer graphics [9, 17–20].

Trajectory reduction refers to the methods used to decide which points from the raw trajectory should be discarded or which points should be retained in the raw trajectory [14]. For example, when it is necessary to store and process the trajectory data obtained by taking samples at 5-second intervals of a fleet of 1000 vehicles, the storage and time costs will increase considerably, and the operations will be complicated [4]. Using an appropriate method, a correct representation of an original trajectory can be brought by removing unnecessary points while preserving the major distinctive elements [15]. For the vehicle fleet in the example above, a trajectory

set consisting of  $1000 * 24 * 60 * (60 / 5)$ , that is, 17,280,000 points, will be acquired in a day. However, using trajectory reduction methods, this number of points can be reduced to 1%–20% of this number.

One of the important considerations for trajectory reduction algorithms is to decrease the amount of the trajectory data while shrinking information loss [7, 21]. The raw points, in other words, the remaining points on the original trajectory, are called simple trajectories [14]. Although this simplified trajectory can be used in calculations more efficiently than the original ones, the results of both will not be quite different from each other. In addition, the processing time of the calculation with the simplified trajectory will be much shorter than that of the original trajectory [14]. The most advanced trajectory reduction methods often involve either high algorithmic complexity [22, 23] or insufficient reduction rates [24, 25]. This leads to the quick depletion of memory, computational power, energy resources, and storage [26].

Shape and displacement refer to the differences in shape and position between the simplified and the versions of a trajectory. This difference is due to the method used in the algorithms. The positional accuracy of the original trajectory affects the positional accuracy of its simplified version [27]. Topological errors that can be made during simplification include coincident lines, line crossings, and collapsed zero-length lines. Therefore, the performances of various trajectory simplification methods need to be evaluated in different ways [12, 28].

In this study, a trajectory data reduction system is presented that can work both online and offline alternately and transfer the reduced and accumulated data to the cloud when it has Internet access. An effective reduction method with low computational complexity has been proposed to work online on mobile devices. The proposed method has an acceptably low dissimilarity rate, consumes less energy, and can operate at low bandwidth. To illustrate the proficiency and operation of the method proposed in this study, a mobile application has been developed that collects and reduces GNSS trajectory data on a mobile device. It sends reduced data to a database in the cloud while online. However, it stores the reduced data on the mobile device until it is online again. The proposed method has been tested on this developed mobile application and has been found to be very successful.

The rest of this paper is organized as follows: In Sect. 2, we present the studies conducted in the literature. In Sect. 3, we introduce the proposed method in this study. In Sect. 4, the features of the mobile application developed for testing the online operation of the method are described. In Sect. 5, we compare and analyze the proposed method with the classical RDP algorithm in terms of reduction results, reduction rate, root-mean-square error (RMSE), and runtime. In Sect. 6, the novelty of the proposed method, its robustness, and superiority compared to methods in the literature are explained.

Additionally, the results and future research directions are discussed.

## 2 Literature Review

Douglas and Peucker (1973) proposed a method known as the iterative endpoint-fitting algorithm, also called the Douglas–Peucker (RDP) algorithm. The Douglas–Peucker algorithm segments the data and removes points that are within the  $\epsilon$  parameter distance from each other. This effectively reduces the points on the trajectory. The degree of simplification is controlled by the  $\epsilon$  parameter, which defines the maximum distance between the original points and the simplified curve [29]. Although it has a high reduction rate, the shortcomings of this method lie in visual difference due to reduction based on the epsilon value and its offline operation.

Katsikouli et al. (2014) proposed an algorithm that can reduce large volumes of location and sensor data online. This method utilizes topological persistence to identify significant features in the data stream at a large scale. Additionally, they demonstrated that the reduction for each new location data point using this method can be achieved with  $O(1)$  time complexity [30]. While producing a high reduction rate in a short running time, this method's weakness is the visual difference.

Qian and Lu (2017) proposed a method that simplifies GPS location data using the enhanced Douglas–Peucker (EDP) algorithm. The EDP algorithm, working in conjunction with Enhanced Spatial-Temporal Constraints (ESTC), simplifies location data while preserving critical points. The study suggests that the speed profile can uniquely define a trajectory and can be used to evaluate the effectiveness of simplification. When applied to two sample GPS trajectories, the proposed ESTC-EDP simplification method yielded more effective results compared to the traditional DP algorithm [31]. This method's weakness is the possibility of not selecting some critical points and providing a relatively lower reduction rate.

Chen et al. (2019) presented an online navigation reduction framework that attempts to detect the position change with the spatial-directional matching (SD-Matching) algorithm that aligns the sparse and noisy GPS location points working in the mobile environment to the road network. The framework stages consist of online trajectory mapping and motion reduction [32]. The inclusion of preprocessing stages in this method may result in a slow operation.

Xu et al. (2019) proposed a new method that can be applied to vehicle trajectory data by combining the crossroad-rank (CRRank) algorithm and the origin–destination entropy with flow (ODEF) algorithm. ODEF works based on entropy to assess the centrality and calculate the importance of a point's traffic flow. CRRank depends on eigenvector centrality, which seizes reciprocally reinforcing affinities of

intersection, path, and OD pairs. Furthermore, CRRank also evaluates the irreplaceability of a point and the spatial associations between adjacent points [33]. Due to the presence of preprocessing stages in this method, it can run slowly.

Zhang et al. (2019) proposed a trajectory reduction method, which they named WDA, using a combination of the road width (W), the vehicle driving direction (D), and the turning angle (A). This method decides whether the vehicle travel points are linear concerning W and D within an allowable error range and then chooses the points at which a vehicle turns considerably relating to A [25]. The algorithm assesses the limitations of road shapes on the trajectories of vehicles. Thus, it can detect whether the motion forms an approximate curve or a straight line that points to a rotating track.

Liu et al. (2020) proposed a new line simplification algorithm that relies on RDP-based, monotonic chains, and duality. First, to simplify the original lines, the method uses the traditional D-P algorithm and then divides the reduced lines into monotonous chains. The dichotomy is then operated to effectively investigate the junction locations of the chains and to process the crossing chains. By doing so, the method solves its intersection problems [20]. While the proposed method addresses self-intersection problems, it does not take into account issues related to area preservation after polyline simplification. Similar to the RDP algorithm, the proposed method neglects the bending characteristics of curves.

Iiyama et al. (2022) proposed an approach named SQUISH-E(mu) with Stay Areas (SESA), which is a rapid batch compression technique for trajectory data based on identified stay areas. A stay area pertains to a location where a user remains stationary for a specific duration, such as waiting at a traffic light or a bus stop. Additionally, SESA divides the trajectory into smaller sub-trajectories using these identified stay areas and employs the SQUISH-E(mu) compression method on each sub-trajectory. Through experiments conducted on trajectory datasets, it has been demonstrated that SESA manages to reduce compression time by about 65% [34]. Despite the improvement in speed, the method still offers relatively slow performance due to its complexity.

Al Jawarneh et al. (2023) proposed a method named GeoRAP to process geospatial data streams online. A method named GeoRAP, designed to process geospatial data streams online, has been proposed. GeoRAP simplifies the working area using the RDP algorithm and then minimizes response time while maintaining accuracy through a spatial stratified sampling method. This method is developed to excel in various online and batch geospatial processing workloads, especially with large datasets, aiming to provide high throughput and accuracy [35]. Being an RDP-based method, it operates only offline.

Sasaki et al. (2023) introduced a framework for a model-less feedback system driven by tourist tracking data. The

system utilizes tourist routes collected through mobile applications to visualize the differences between geomeia recommendations and the actual routes chosen by tourists. The mobile collaborative framework is used to reorganize tourist routes. This process includes extracting the location where user-generated content (UGC) is recorded, abstracting locations where tourists stay, discarding locations where users remain stationary, and simplifying the remaining location points. Subsequently, the heatmap system visualizes heatmaps for hot streets, UGC-oriented hot spots, and indoor-oriented hot spots. Experimental results indicate that this method can create a more suitable route for hot street visualization compared to the raw and simplified routes in terms of geometry [36]. Due to the relatively complex nature of the method, its weakness is running time.

Xiong et. al (2023) proposed a method that utilizes the Spark big data processing engine to compress large amounts of GPS trajectory datasets in parallel. By parallelizing classical trajectory compression algorithms such as RDP, Top-Down Time-Ratio (TD-TR), SQUISH, and Velocity-Aware Douglas–Peucker (V-DP), the study evaluates the performance of these algorithms. The experiments demonstrate that, with a Spark cluster of 14 nodes, the dataset can be compressed in just 438 s. Furthermore, the parallel algorithms achieve an average storage cost savings of 26%, reaching up to 40% [37]. The method cannot distinguish between trajectories coming from the same source. It needs improvement with processes such as an appropriately sized time window, vehicle trajectory prediction, and candidate road segment pre-fetching.

Zhao et al. (2023) proposed a method to solve the maximum angular prediction problem to achieve real-time motion planning for tractors and multiple towed vehicles. The aim was to detect discrepancies between planned and executed trajectories and to reduce the trajectory for obstacle-free path planning. A trajectory reduction method was suggested, taking into account the dynamics of the trajectory, and a controller-based reduction approach was proposed. This was intended to reduce the mismatch between the original and reduced trajectories. The method led to the efficient creation of trajectories in a dynamic environment without obstacles [38]. The increase in the length of the tractor introduces noise in terms of position accuracy. Therefore, the method is relatively less resilient to noise.

### 3 Proposed Trajectory Reduction Method

To reduce the GNSS trajectory data, we developed a method that is classified as *Unconstrained Extended Local Processing*, which is one of the line simplification classes proposed by McMaster [39]. The components and symbols of the trajectory reduction used in this study are shown in Fig. 2.

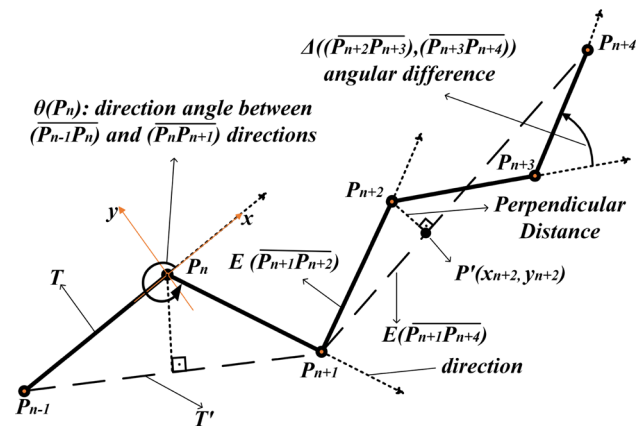


Fig. 2 Components and symbols of a trajectory reduction used in the study

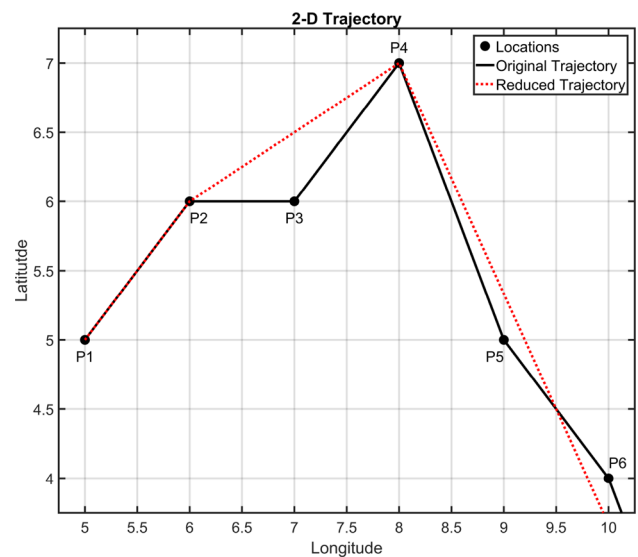


Fig. 3 Original and reduced trajectories

According to the figure, the original trajectory is symbolized as  $T$ , and its simplified version is  $T'$ ,  $P_n(x, y)$  is a point, and its coordinates at timestamp  $n$ ,  $E(P_{n+1}P_{n+2})$  is the vertex between points  $P_{n+1}$  and  $P_{n+2}$ ,  $\theta(P_n)$  is the direction angle between the lines  $(P_{n-1}P_n)$  and  $(P_nP_{n+1})$ ,  $\Delta(P_n)$  is the angular difference between the lines  $(P_{n-1}P_n)$  and  $(P_nP_{n+1})$ .

The method determines whether the new incoming location data,  $P_n(x, y)$ , are a turning point each time it reads from a GNSS receiver while operating online or reads GNSS data from a file while operating offline. A location data that is determined as a turning point is saved in a database. The original and reduced trajectories shown in Fig. 3 are used as an example of the turning point determination process with the proposed method. In the decision phase, the three-cell window system shown in Fig. 4 is used.

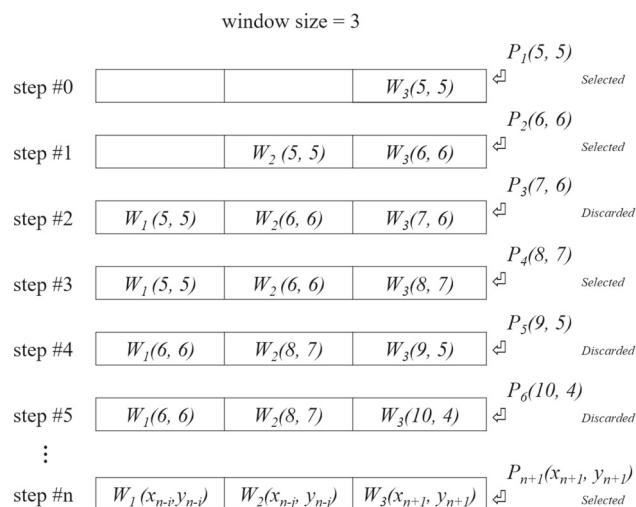


Fig. 4 Proposed windowing system for the sample trajectory data

In the first step, the starting point of the trajectory is added to the third cell of the empty window. In the second step, the first point is shifted to the left of the second cell, and the second point is added to the third cell. In the third step, the points in the second and third cells are shifted to the left, and the third point is added to the third cell. At this stage, the directions of the edge between the point in the first cell and the point in the second cell and the edge between the points in the second and third cell are calculated. The direction angles between these two edges are determined. If the difference between these angles is less than a predetermined threshold level, it is assumed that there is not much difference between the directions of the two edges, and the point in the third cell is discarded. If the difference between the direction angles is greater than the threshold value, as in step #4 in Fig. 4, a turning in the trajectory is considered, and the fourth point is selected as a new point of the reduced trajectory. Each selected point is saved in the database. The coordinates of each selected point of the reduced trajectory, its angular location relative to the previous point, and the displacement amount are stored in the database.

The method starts by determining the window size and threshold value. The window size is important in terms of the desired precision when determining the turning point. An excessively large window size will require numerous reference points, resulting in a low reduction ratio and a higher similarity to the original trajectory. Therefore, a 3-cell or 4-cell window size is adequate. Lowering the threshold value results in a reduced trajectory that is closer to the original trajectory but with a lower reduction rate.

The error due to line simplification, i.e., the error due to the difference in position between the original trajectory  $T$  and the simplified version  $T'$ , which is called the displacement error, is measured by the perpendicular distance of the

original point to its simplified trajectory, as shown in Fig. 2. The overall performance of the simplification can be calculated by the RMSE in Eq. (1). In the equation,  $x_n$  and  $y_n$  are the coordinates of the original point  $P_n$ , and  $x_n^1$  and  $y_n^1$  are the coordinates of perpendicular point  $P_n^1$  on the simplified trajectory.

$$RMSE = \sqrt{\frac{1}{N} \sum_N ((x_n - x_n^1) + (y_n - y_n^1))^2} \tag{1}$$

To reduce GNSS data, a method that is better than the existing methods in the literature and can work online has been developed. The method is explained by the pseudocode shown in Algorithm 1.

**Algorithm 1** Pseudocode of the proposed method

```

Require:  $W$ : window,  $\omega$ : window size,  $\rho_{init}$ : angle threshold
Output:  $T'$ : reduced trajectory
1: while  $p \leftarrow \text{READ}(\text{GNSSReceiver})$  do
2:   for  $i \leftarrow 1$  to  $\omega$  do
3:      $\rho_{threshold} \leftarrow \frac{\rho_{init}}{2^{(i-1)}}$ 
4:      $\theta_i \leftarrow \arctan(\frac{p}{W_i}) * \frac{180}{\pi}$ 
5:     if  $\theta_i \geq \rho_{threshold}$  then
6:        $W \leftarrow \text{SHIFTLEFT}(W, \omega - 1)$ 
7:        $W_\omega \leftarrow p$ 
8:        $T' \leftarrow p$ 
9:       break
10:    end if
11:  end for
12: end while
    
```

In Algorithm 1, window size and angular threshold values are used as input parameters. The number of angular thresholds varies with window size. Angular threshold values are assigned by dividing each cell of the window in half, starting with the initial threshold value being assigned to the first cell. This is done to increase precision and is given in Eq. (2).

$$\rho_{threshold} = \frac{\rho_{init}}{2^{(i-1)}} \tag{2}$$

In Eq. (2),  $\rho_{threshold}$  represents the threshold value to be used for the currently calculated window cell. On the other hand,  $\rho_{init}$  is used to decide according to the angular position of  $W_1$  and is initially determined by considering the window size.  $i$  shows the index value of the window cell to be decided. For example, for a 3-cell window, if the angular threshold value for  $W_1$  is set as  $\rho_{init} = \frac{\pi}{2}$ , for  $W_2$ ,  $\rho_2 = \frac{\pi}{2} * \frac{1}{2^{(2-1)}} = \frac{\pi}{4}$ , for  $W_3$ , it is determined as  $\rho_3 = \frac{\pi}{2} * \frac{1}{2^{(3-1)}} = \frac{\pi}{8}$ .

At the output of the method, a reduced coordinate sequence is produced. The method runs as long as there is data acquisition. If the method is used online, data are streamed from the device's GNSS receiver. Otherwise, data are read from a file while operating offline. When the point data is acquired from a sensor or read from a file, it is checked

within an iteration whether the threshold value of that cell is exceeded for each of the cells in the window. If any of the threshold values are exceeded, the point in that cell is considered to be a turn, the iteration is terminated, and the turning point is added to the reduced trajectory sequence. After determining the turning point, the points in the window are shifted left by one, and the next point in the original trajectory is added to the window. If the newly acquired point is not the turning point, no changes are made to the window.

To determine the relative position of two points in Algorithm 1, the calculation given in Eq. (3) is performed.

$$\theta = \tan^{-1} \left( \frac{x}{y} \right) * \frac{180}{\pi} : \{x, y \mid x = x_n - x_{n-i}; \quad (3)$$

$$y = y_n - y_{n-i}\}$$

In this equation,  $\theta$  indicates the angular value of a point in the plane, the latitude and longitude information of each point in the  $x$  and  $y$  plane.  $x_{n-i}$  and  $y_{n-i}$  variables are the longitude and latitude information of the cell to be used in the calculation by reference in the window, and  $x_n$  and  $y_n$  are the longitude and latitude information of the point whose angular position is to be calculated according to the reference point.

In  $n$  iterations, a turning decision is made for each new point obtained from the GNSS receiver or a file. Therefore, the algorithmic complexity required for the selection of each point in the developed method is  $O(n)$ , as it will be proportional to the size of the data. The complexity of the pseudocode shown in Algorithm 1 is  $O(n)$ , as the algorithm will run as long as there is a data stream from the GNSS receiver. In the studies conducted in the literature, the algorithmic complexity of the developed methods is  $O(n)$ ,  $O(n^2)$ , and  $O(n \log n)$ . The method developed in this study performs more efficiently than the existing methods in the literature [25, 32, 40] because of its low algorithmic complexity. Therefore, when the point is obtained, it can be detected whether it is a turn while running online. The method shown in Algorithm 1 can also run offline on existing data or datasets instead of reading data from the GNSS receiver. In this way, an effective reduction can be performed on both online and offline data.

#### 4 Testing the Developed Mobile Application and Method

In this study, a mobile application with edge computing [41] was developed to test and visualize that the developed method also works online on mobile devices. Android Studio (version 2020.3.1) IDE, Dart (version 211.7808) language, and Flutter (version 65.2.2) SDK software were used for application development. The application performs data reduction

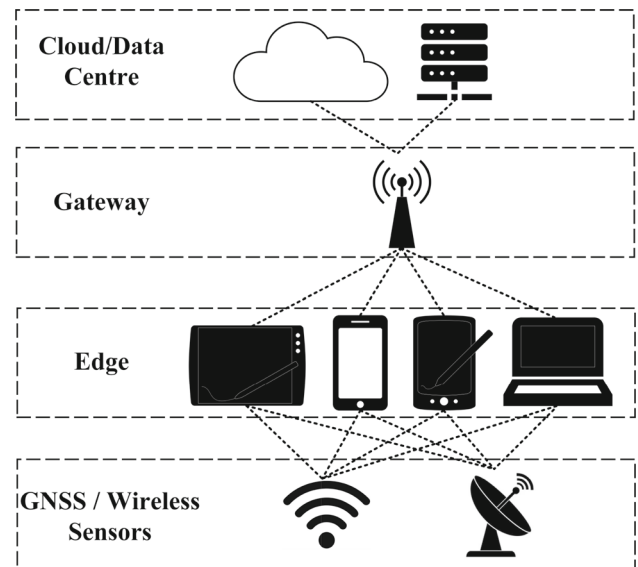


Fig. 5 The diagram of the edge system used in this study

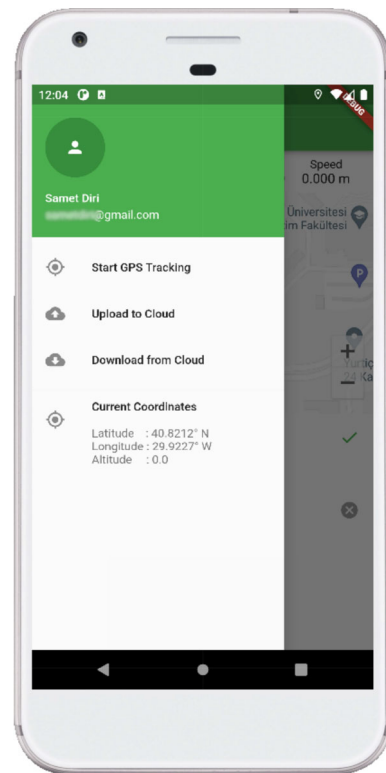


Fig. 6 Mobile application's main menu

on the mobile device by using edge computing when data are read from the sensors of the device and then stored in the database existing in the cloud via a gateway. An example diagram of the application is shown in Fig. 5.

The developed application, whose interface is shown in Figs. 6 and 7, has been tested in real environments on the Samsung Galaxy M51 (SM-M515F) mobile device with

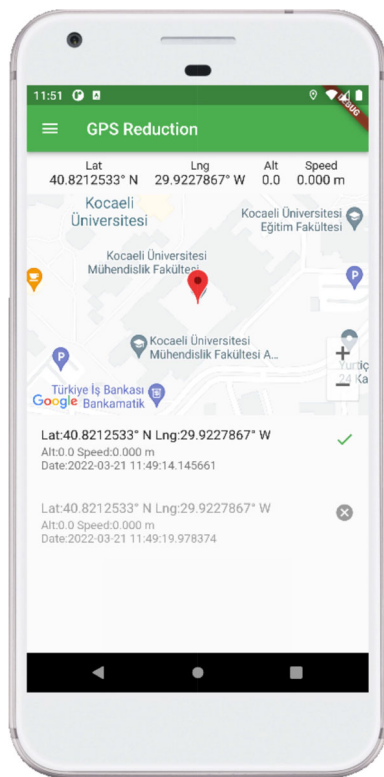


Fig. 7 Mobile application's home screen

the Android 11 operating system. Using the application, Google's cloud-based Firebase Real-time Database [42, 43] was used to store the retained points. Thus, a real-time trajectory data stream is provided from moving objects. Using a cloud-based database, more than one trajectory stream can be created simultaneously. Deng et al. (2017) mentioned that if multiple trajectory flows occur simultaneously while recording trajectories to the cloud database, flow inconsistency may occur at the input and output [29]. In addition, they presented their studies with the assumption that the method they developed works in the absence of coincidence. To overcome the problem mentioned here, users must register to use the application. In this way, users must log in, and trajectories are stored separately for each user. In addition, to distinguish between two trajectory records belonging to each user, a field with the name *isStart* that stores the true value is added to the first record added to the database when each trajectory record is started. Therefore, multiple trajectories of each user can be stored in the same database. The *altitude*, *isSelected*, *isStart*, *latitude*, *longitude*, *speed*, and *timeStamp* values of each point are stored in the cloud database. The timestamp value of the GNSS's atomic clock is stored in the *timeStamp* field. By doing so, we aimed to prevent the time shift that will be encountered in the process of reassembling the data that is thought to be done in the future. Because we want to create a dataset for testing the method offline and

comparing results online, all the points obtained are saved in the database. The *isSelected* field stores a Boolean value to distinguish the points that are detected as a turn during the online test of the method. Using this field, the consistency of the developed reduction method was checked.

The application consists of a login screen, a menu, and a main screen. By logging in, only the logged-in user on Firebase can access and add their data. When the *Start GPS Tracking* button is pressed from the main menu, the GNSS location data acquisition and reduction process is initiated. Likewise, when the application is run, the GNSS location data acquisition and reduction process starts automatically. The *Upload to Cloud* button allows the transfer of reduced data that cannot be transferred to the cloud database in environments where there is no internet connection or where it is weak. Clicking the *Download from Cloud* button, the data saved in the cloud database by the user is downloaded to the mobile device and displayed on the map. The reduced and/or downloaded GNSS location data from the cloud database is visualized as shown in the screen shown in Fig. 6 by programming the Google Maps API. Again, at the top of the main screen, the latitude, longitude, altitude, and speed information of the most recently obtained location are displayed. In addition, the latitude, longitude, altitude, speed, and time stamp information of past locations are listed at the bottom of the screen. On the map, the origin, reduced points, and selected points are, respectively, depicted with blue, red, and green markers. Likewise, in the list of visited locations, selected points are highlighted with a green tick mark, and reduced points are shown with a pale gray cross mark. It has been observed that the developed application is efficient in terms of testing the method and obtaining successful results.

## 5 Results of the Proposed Method

In the literature, there are studies on GNSS data reduction or line simplification. In this study, a method that has been observed to work efficiently in the field of GNSS data reduction has been developed. It is thought that the proposed method can be used effectively and easily not only for GNSS data but also in areas such as computer graphics and cartography. Because the data in these fields, such as GNSS data, incorporate a coordinate system.

The proposed method and the RDP algorithm, which is widely used in the literature, are compared. The parameters of both methods and the comparison results are shown in Table 1. The comparison was made using the Microsoft Geolife dataset file "20090522094234.plt". Moreover, examples of visual comparisons are shown in Figs. 8 and 9.

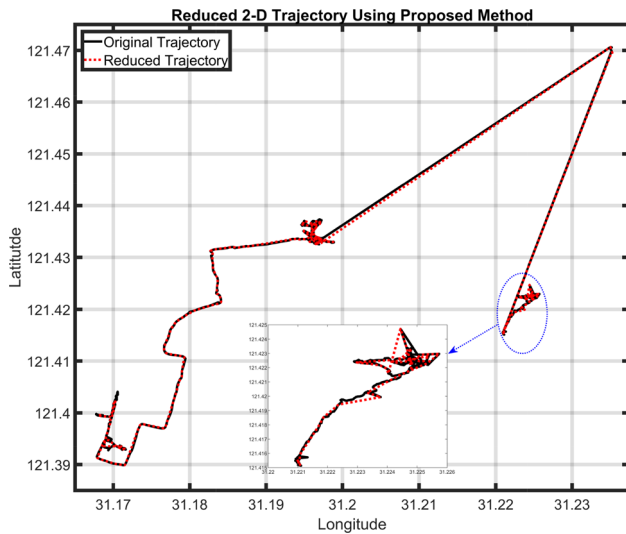
Figure 8 depicts the dataset output when the proposed method is run with an angular threshold of  $\theta = 5^\circ$  and a three-cell window, according to the results in Table 1. On the

**Table 1** Parameters and comparison results of the proposed method and RDP on “20090522094234.plt”

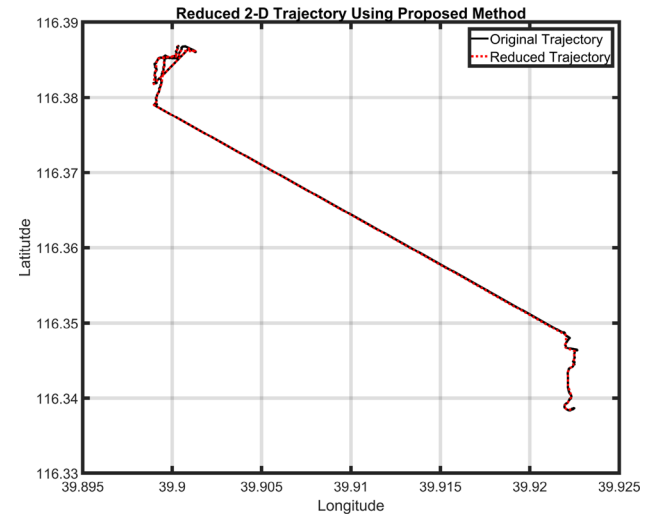
	Proposed method	RDP Algorithm
# of coordinates	2560	2560
# of reduced coordinates	230	35
Reduction rate	91.01 %	98.63 %
Runtime (ms)	14.54	19.24
RMSE value	5.8744e−04	1.1176e+03
Threshold	$\theta = 5^\circ$	$\epsilon = 0.001$
Window size	3	–

**Table 2** Parameters and comparison results of the proposed method and RDP on “20081028002304.plt”

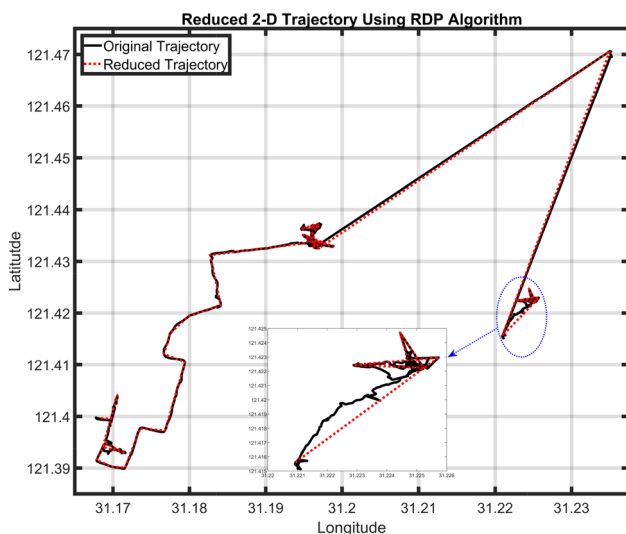
	Proposed method	RDP Algorithm
# of coordinates	328	328
# of reduced coordinates	64	37
Reduction rate	80.49 %	88.72 %
Runtime (ms)	12.6	18.81
RMSE value	1.1442e−04	1.3558e+02
Threshold	$\theta = 10^\circ$	$\epsilon = 0.0001$
Window size	3	–



**Fig. 8** Visual comparison of original and reduced trajectories for the proposed method



**Fig. 10** Visual comparison of original and reduced trajectories for the proposed method

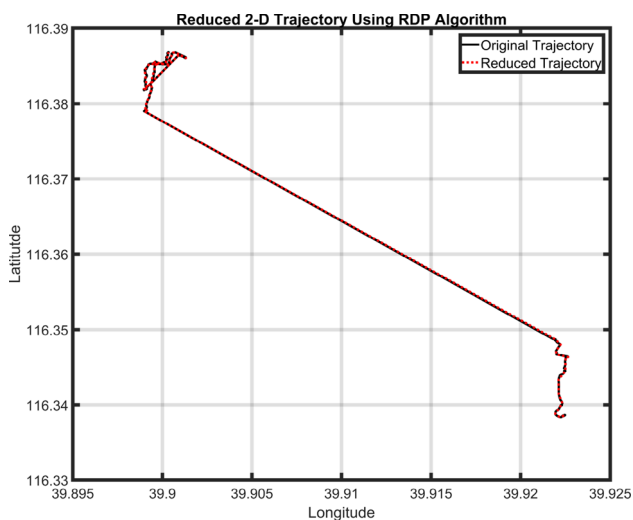


**Fig. 9** Visual comparison of original and reduced trajectories for the RDP Algorithm

other hand, Fig. 9 depicts the dataset output when the RDP algorithm is run with a threshold of perpendicular distance  $\epsilon = 0.001$ . Under these conditions, the proposed method reduces the dataset consisting of 2560 points to 230 points by processing it in 0.014539 s. It was observed that the reduction rate of the proposed method was 91.01%, and the RMSE value was 5.8744e−04. The RDP algorithm processed the same dataset in 0.019237 s, reducing it to 35 points. It was observed that the reduction rate of the RDP algorithm was 98.63%, and the RMSE value was 1.1176e+03. According to these results, the reduction rate of the RDP is 7.73% better than the proposed method in this study. However, the proposed method has approximately  $2e+7$  times lower RMSE error than the RDP. Furthermore, our method is 24.4% faster than the RDP.

Another comparison of the results and the parameters of both methods are shown in Table 2. The comparison was made using the Microsoft Geolife dataset file “20081028002304.plt”. Moreover, examples of visual comparisons are shown in Figs. 10 and 11.





**Fig. 11** Visual comparison of original and reduced trajectories for the RDP Algorithm

Figure 10 depicts the dataset output when the proposed method is run with an angular threshold of  $\theta = 10^\circ$  and a three-cell window, according to the results in Table 2. On the other hand, Fig. 11 depicts the dataset output when the RDP algorithm is run with a threshold of perpendicular distance  $\epsilon = 0.0001$ . Under these conditions, the proposed method reduces the dataset consisting of 328 points to 64 points by processing it in 0.012565 s. It was observed that the reduction rate of the proposed method was 80.49%, and the RMSE value was  $1.1442e-04$ . The RDP algorithm processed the same dataset in 0.018817 s, reducing it to 37 points. It was observed that the reduction rate of the RDP algorithm was 88.72%, and the RMSE value was  $1.3558e+02$ . Based on these findings, the RDP outperforms our proposed method by

8.23% in terms of reduction rate. Nevertheless, our method exhibits a lower RMSE error compared to the RDP algorithm. Additionally, the proposed method demonstrates a 49.3% faster running time than RDP.

## 6 Conclusion

GNSS data from applications developed for different purposes are obtained from various sources. Utilizing excessive amounts of unnecessary data poses various challenges for computing systems, notably in terms of increased storage requirements, heightened processing demands, and the generation of unwanted noise. In this study, we proposed a novel GNSS data reduction method, providing high reduction rates, low error rates, and faster processing. This method was designed to operate seamlessly, both offline and online on mobile devices for GNSS data. In the proposed method, the decision phase involves the use of windowing with reference points to reduce a point. The proposed method has an acceptably low dissimilarity rate and can operate in low bandwidth. To demonstrate the operation and efficiency of the proposed method, we developed a mobile application which functions as an edge system, collecting and reducing GNSS trajectory data directly on the mobile device. The mobile device stores the reduced data on the device until it goes online. When online, it transmits the data to a cloud-based database. The developed mobile application, incorporating the proposed method, has undergone testing, demonstrating its notable success.

In existing literature, displacement, distortion, and runtime are commonly used measures for evaluating the performance of algorithms developed for GNSS data reduction.

**Table 3** Comparison results of the proposed method and other methods in the literature [44]

Line simplification Algorithm	Online Processing	Visual difference	Displacement in the 'certain' case	Displacement in the uncertain case	Shape distortion in the 'certain' case	Shape distortion in the uncertain case	Reciprocal of the computation time case
Proposed method	+	***	***	***	***	***	***
Douglas–Peucker	-	***	***	***	***	***	*
Zhao–Saalfeld	-	***	**	**	**	*	***
Reumann–Witkam	-	***	**	**	**	**	*
Visvalingham–Whyatt	-	***	**	**	**	*	*
Distance between points	+	***	*	*	*	*	***
Opheim	-	***	*	*	*	*	*
Lang	-	***	**	**	**	*	**
Perpendicular distance	-	**	*	*	*	*	***
nth point	+	*	*	*	*	*	***

Table 3 shows the comparison between some methods found in the literature and the proposed method in this study. In the table, more stars indicate a better performance in terms of the corresponding assessment. As shown in Table 3, a trace of the methods presented in the literature can also be used online, similar to the proposed method.

Based on the test results, while operating on identical datasets and hardware, the proposed method exhibited a runtime improvement of up to 49% and achieved an RMSE value  $2e+7$  times better than the RDP algorithm.

The novelty of this study lies in its capability to perform data reduction-both online, at the moment it is obtained, and at high speed with a low error rate. Only a limited number of the methods in the literature can perform reduction processes online, as in this study. However, these methods generally have higher error rates and/or slower runtimes. Therefore, in cases where data needs to be processed in the future, the possibility of having a trajectory different from the original trajectory exists, leading to potentially incorrect results. Moreover, when methods that work online are applied on mobile devices with relatively lower bandwidth, battery life, and processing capacity, the high runtime cost of data reduction may not be efficient. The proposed method aims to overcome these challenges.

As a result of the tests and literature review, it was observed that the method produced better outcomes compared to the existing methods in terms of dissimilarity rate, processing time, and visual distortion. For example, the visual difference between the output obtained from the  $n$ th point method is significantly greater when compared to the results produced by the proposed method, as the former method exhibits low-performance results in terms of turn detection [44, 45]. In addition, some methods in the literature work offline but exhibit a low visual difference. However, when evaluating their effectiveness based on success rates in shape distortions and displacement against the proposed method, they demonstrate inferior performance. For instance, the widely used Douglas–Peucker method and its derivatives yield subpar results in terms of shape distortion and displacement. On the contrary, the method proposed in this study not only addresses the shortcomings of the previous method but also outperforms in terms of runtime and has the capability to function online.

Consequently, we have proposed an approach that exhibits significantly faster runtime and remarkably lower error rates while maintaining a reasonable reduction rate. Furthermore, it is believed that existing methods in the literature seldom operate online, underscoring the significance of our developed method in addressing this notable gap in the literature. The results obtained in this study indicate that the proposed method is well-suited for running efficiently on large datasets. In future research, efforts are aimed at making

the proposed method compatible with parallel programming to ensure higher performance in processing both online and offline data.

**Funding** Open access funding provided by the Scientific and Technological Research Council of Türkiye (TÜBİTAK).

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

1. Shan-shi, Z.; Xiao-gong, H.; Li, L.; Feng, H.; Cheng-pan, T.; Jun-yang, P.: Status of satellite orbit determination and time synchronization technology for global navigation satellite systems. *Chin. Astron. Astrophys.* **43**(4), 479–492 (2019). <https://doi.org/10.1016/j.chinastron.2019.11.003>
2. Current and Planned Global and Regional Navigation Satellite Systems and Satellite-based Augmentations Systems. United Nations Office for Outer Space Affairs. Accessed 22 Oct 2022 (2010)
3. Sturm, J.: GPS: Global Positioning System. Let's Explore Science (Hardcover). Rourke Publishing, Vero Beach (2001)
4. Zheng, Y.; Xie, X.; Ma, W.: GeoLife: a collaborative social networking service among user, location and trajectory. *IEEE Data Eng. Bull.* **33**(2), 32–40 (2010)
5. 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. WWW '09, pp. 791–800. Association for Computing Machinery, New York (2009). <https://doi.org/10.1145/1526709.1526816>
6. 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. UbiComp '08, pp. 312–321. Association for Computing Machinery, New York (2008). <https://doi.org/10.1145/1409635.1409677>
7. Zheng, Y.: Trajectory data mining: an overview. *ACM Trans. Intell. Syst. Technol.* (2015). <https://doi.org/10.1145/2743025>
8. Muckell, J.; Hwang, J.H.; Lawson, C.T.; Ravi, S.S.: Algorithms for compressing GPS trajectory data: an empirical evaluation. In: Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems. GIS '10, pp. 402–405. Association for Computing Machinery, New York (2010). <https://doi.org/10.1145/1869790.1869847>
9. Zhao, L.; Shi, G.: A method for simplifying ship trajectory based on improved Douglas–Peucker algorithm. *Ocean Eng.* **166**, 37–46 (2018). <https://doi.org/10.1016/j.oceaneng.2018.08.005>
10. Potamias, M.; Patroumpas, K.; Sellis, T.: Sampling trajectory streams with spatiotemporal criteria. In: 18th International Conference on Scientific and Statistical Database Management (SSDBM'06), pp. 275–284 (2006). <https://doi.org/10.1109/SSDBM.2006.45>



11. Sandu Popa, I.; Zeitouni, K.; Oria, V.; Kharrat, A.: Spatio-temporal compression of trajectories in road networks. *GeoInformatica* **19**(1), 117–145 (2015). <https://doi.org/10.1007/s10707-014-0208-4>
12. Long, C.; Wong, R.C.-W.; Jagadish, H.V.: Direction-preserving trajectory simplification. *Proc. VLDB Endow.* **6**(10), 949–960 (2013). <https://doi.org/10.14778/2536206.2536221>
13. Gasparetto, A.; Boscaroli, P.; Lanzutti, A.; Vidoni, R.: Path planning and trajectory planning algorithms: a general overview. In: *Motion and Operation Planning of Robotic Systems. Mechanisms and Machine Science*, pp. 3–27. Springer, Cham (2015)
14. Lin, C.-Y.; Hung, C.-C.; Lei, P.-R.: A velocity-preserving trajectory simplification approach. In: *2016 Conference on Technologies and Applications of Artificial Intelligence (TAAI)*, pp. 58–65 (2016). <https://doi.org/10.1109/TAAI.2016.7880172>
15. Meng, Q.; Yu, X.; Yao, C.; Li, X.; Li, P.; Zhao, X.: Improvement of OPW-TR algorithm for compressing GPS trajectory data. *J. Inf. Process. Syst.* **13**(3), 533–545 (2017)
16. Barbeau, S.: GPSTest. <https://github.com/barbeau/gptest>
17. Spinsanti, L.; Berlingerio, M.; Pappalardo, L.: Mobility and geosocial networks. In: Renso, C., Spaccapietra, S., Zimányi, E. (eds.) *Mobility Data: Modeling, Management, and Understanding*, pp. 315–333. Cambridge University Press, Cambridge (2013)
18. Zhang, S.-K.; Liu, Z.-J.; Cai, Y.; Wu, Z.-L.; Shi, G.-Y.: AIS trajectories simplification and threshold determination. *J. Navig.* **69**(4), 729–744 (2016). <https://doi.org/10.1017/S0373463315000831>
19. Fernández-García, N.L.; Del-Moral Martínez, L.; Carmona-Poyato, A.; Madrid-Cuevas, F.J.; Medina-Carnicer, R.: A new thresholding approach for automatic generation of polygonal approximations. *J. Vis. Commun. Image Represent.* **35**, 155–168 (2016). <https://doi.org/10.1016/j.jvcir.2015.12.013>
20. Liu, B.; Liu, X.; Li, D.; Shi, Y.; Fernandez, G.; Wang, Y.: A vector line simplification algorithm based on the Douglas-Peucker algorithm, monotonic chains and dichotomy. *ISPRS Int. J. Geo Inf.* **9**(4), 251 (2020). <https://doi.org/10.3390/ijgi9040251>
21. Tasnim, S.; Caldas, J.; Pissinou, N.; Iyengar, S.S.; Ding, Z.: Semantic-aware clustering-based approach of trajectory data stream mining. In: *2018 International Conference on Computing, Networking and Communications (ICNC)*, pp. 88–92 (2018). <https://doi.org/10.1109/ICCNC.2018.8390371>
22. Liu, J.; Zhao, K.; Sommer, P.; Shang, S.; Kusy, B.; Lee, J.-G.; Jurdak, R.: A novel framework for online amnesic trajectory compression in resource-constrained environments. *IEEE Trans. Knowl. Data Eng.* **28**(11), 2827–2841 (2016). <https://doi.org/10.48550/arXiv.1605.02337>
23. Nibali, A.; He, Z.: Trajic: an effective compression system for trajectory data. *IEEE Trans. Knowl. Data Eng.* **27**(11), 3138–3151 (2015). <https://doi.org/10.1109/TKDE.2015.2436932>
24. Wu, S.-T.; Marquez, M.R.G.: A non-self-intersection Douglas-Peucker algorithm. In: *16th Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI 2003)*, pp. 60–66 (2003). <https://doi.org/10.1109/SIBGRA.2003.1240992>
25. Zhang, H.; Hu, X.; Bai, W.; Zhang, H.; Zhou, Y.; Lin, Y.: An efficient IoV trajectory compression method in vehicle terminals using width-direction-angle. *IEEE Access* **7**, 71447–71458 (2019). <https://doi.org/10.1109/ACCESS.2019.2919800>
26. Byon, S.; Kwon, E.; Jung, E.-S.; Lee, Y.-T.: An implementation of location trajectory data reduction. In: *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 1270–1272 (2018). <https://doi.org/10.1109/ICTC.2018.8539709>
27. Cheung, K.L.; Shi, W.; Shea, Y.K.G.: Modelling GML for the spatial data presentation and editing on the web and mobile devices. In: *Greater China GIS Conference; Conference Date 1 Jan 2004* (2004)
28. Long, C.; Wong, R.C.-W.; Jagadish, H.V.: Trajectory simplification: on minimizing the direction-based error. *Proc. VLDB Endow.* **8**(1), 49–60 (2014). <https://doi.org/10.14778/2735461.2735466>
29. Deng, Z.; Han, W.; Wang, L.; Ranjan, R.; Zomaya, A.Y.; Jie, W.: An efficient online direction-preserving compression approach for trajectory streaming data. *Future Gener. Comput. Syst.* **68**, 150–162 (2017). <https://doi.org/10.1016/j.future.2016.09.019>
30. Katsikouli, P.; Sarkar, R.; Gao, J.: Persistence based online signal and trajectory simplification for mobile devices. In: *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. SIGSPATIAL '14*, pp. 371–380. Association for Computing Machinery, New York (2014). <https://doi.org/10.1145/2666310.2666388>
31. Qian, H.; Lu, Y.: Simplifying GPS trajectory data with enhanced spatial-temporal constraints. *ISPRS Int. J. Geo Inf.* (2017). <https://doi.org/10.3390/ijgi6110329>
32. Chen, C.; Ding, Y.; Xie, X.; Zhang, S.; Wang, Z.; Feng, L.: Trajcompressor: an online map-matching-based trajectory compression framework leveraging vehicle heading direction and change. *IEEE Trans. Intell. Transp. Syst.* **21**(5), 2012–2028 (2020). <https://doi.org/10.1109/TITS.2019.2910591>
33. Xu, M.; Wu, J.; Liu, M.; Xiao, Y.; Wang, H.; Hu, D.: Discovery of critical nodes in road networks through mining from vehicle trajectories. *IEEE Trans. Intell. Transp. Syst.* **20**(2), 583–593 (2019). <https://doi.org/10.1109/TITS.2018.2817282>
34. Iiyama, S.; Oda, T.; Hirota, M.: SESA: fast trajectory compression method using sub-trajectories segmented by stay areas. In: *Advanced Data Mining and Applications: 18th International Conference, ADMA 2022, Brisbane, QLD, Australia, 28–30 November, 2022, Proceedings, Part I*, pp. 187–198. Springer, Berlin (2022). [https://doi.org/10.1007/978-3-031-22064-7\\_15](https://doi.org/10.1007/978-3-031-22064-7_15)
35. Al Jawarneh, I.M.; Foschini, L.; Bellavista, P.: Polygon simplification for the efficient approximate analytics of georeferenced big data. *Sensors* (2023). <https://doi.org/10.3390/s23198178>
36. Sasaki, I.; Arikawa, M.; Lu, M.; Sato, R.: Mobile collaborative heatmapting to infer self-guided walking tourists' preferences for geomeia. *ISPRS Int. J. Geo Inf.* (2023). <https://doi.org/10.3390/ijgi12070283>
37. Xiong, W.; Wang, X.; Li, H.: Efficient large-scale GPS trajectory compression on spark: a pipeline-based approach. *Electronics* (2023). <https://doi.org/10.3390/electronics12173569>
38. Zhao, H.; Chen, W.; Zhou, S.; Zheng, F.; Liu, Y.-H.: Localization and motion planning of industrial tractor-trailers vehicles. *IEEE Trans. Control Syst. Technol.* (2023). <https://doi.org/10.1109/tcst.2023.3275497>
39. McMaster, R.B.: Automated line generalization. *Cartogr. Int. J. Geogr. Inf. Geovisualization* **24**(2), 74–111 (1987). <https://doi.org/10.3138/3535-7609-781G-4L20>
40. Douglas, D.H.; Peucker, T.K.: Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartogr. Int. J. Geogr. Inf. Geovisualization* **10**(2), 112–122 (1973). <https://doi.org/10.3138/FM57-6770-U75U-7727>
41. De Donno, M.; Tange, K.; Dragoni, N.: Foundations and evolution of modern computing paradigms: cloud, IoT, edge, and fog. *IEEE Access* **7**, 150936–150948 (2019). <https://doi.org/10.1109/ACCESS.2019.2947652>



42. Fitri, N.A.; Emba, R.Z.; Mufid, M.R.; Fiyanto, A.; Wajib, W.; Shofyan, A.: Kediri city tourism object application using firebase realtime database technology. In: Proceedings of the International Conference on Applied Science and Technology on Social Science 2021 (iCAST-SS 2021). Atlantis Press, Paris (2022). <https://doi.org/10.2991/assehr.k.220301.147>
43. Khawas, C.; Shah, P.: Application of firebase in android app development-a study. *Int. J. Comput. Appl.* **179**(46), 49–53 (2018). <https://doi.org/10.5120/ijca2018917200>
44. Shi, W.; Cheung, C.: Performance evaluation of line simplification algorithms for vector generalization. *Cartogr. J.* **43**(1), 27–44 (2006). <https://doi.org/10.1179/000870406X93490>
45. Chen, L.; Özsü, M.T.; Oria, V.: Robust and fast similarity search for moving object trajectories. In: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data. SIGMOD '05, pp. 491–502. Association for Computing Machinery, New York (2005). <https://doi.org/10.1145/1066157.1066213>

