IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received April 26, 2019, accepted May 23, 2019, date of publication May 29, 2019, date of current version June 12, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2919800

An Efficient IoV Trajectory Compression Method in Vehicle Terminals Using Width-Direction-Angle

HUIBING ZHANG[1], XIAOLI HU[1], WEIHUA BAI[2], HONGBO ZHANG[1], YA ZHOU[1], AND YUMING LIN[1]

[1]Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin 541004, China
[2]School of Computer Science, Zhaoqing University, Zhaoqing 526061, China

Corresponding authors: Xiaoli Hu (huxiaoli@guet.edu.cn) and Weihua Bai (bandwerbai@gmail.com)

This work was supported in part by the National Natural Science Foundation under Grant 61662013, Grant U1501252, Grant U1711263, Grant U1811264, Grant 61662015, and Grant 61562014, in part by the Guangxi Innovation-Driven Development Project (Science and Technology Major Project) under Grant AA17202024, in part by the Guangxi Natural Science Foundation under Grant 2017GXNSFAA198372 and Grant 2016GXNSFAA380149, in part by the Funds of Guangxi Key Laboratory of Trusted Software Project under Grant kx201511, and in part by the Funds of Graduate Student Innovation Program Guilin University of Electronic Technology under Grant 2017YJCX56 and Grant 2019YCXS045.

ABSTRACT A highly efficient online compression of vehicle trajectories is important to the application of the Internet-of-Vehicles (IoV) system. However, the high time–space complexity of existing vehicle trajectory compression algorithms makes them unable to adapt effectively to actual IoV applications under the condition of a resource-constrained mobile vehicle terminal, bandwidth cost, and latency. This paper addresses these issues by proposing a trajectory compression algorithm using a combination of road width (W), vehicle driving direction (D), and swerve angle (A). The proposed WDA compression algorithm determines whether the vehicle trajectory points are rectilinear within a permitted error range according to W and D and then determines the trajectory points where a vehicle swerves significantly according to A. The algorithm considers the constraints of road shape on vehicle trajectories according to whether the trajectory forms an approximately straight line or a curve indicative of a swerving trajectory. It adopts different compression methods for rectilinear and swerving vehicle trajectories, which allows for highly efficient compression and high similarity between the compressed and original trajectories. In addition, the algorithm preserves direction information regarding D. Testing employing a Microsoft GeoLife dataset demonstrates that the proposed algorithm attains an average compression ratio of 4.03% for vehicle trajectory data, and the average compression time is 1.15 m/each. Relative to the existing compression algorithms, the proposed algorithm reduced compression time and storage space to adapt for the development of the large-scale IoV applications.

INDEX TERMS IoV, online trajectory compression, width-direction-angle.

I. INTRODUCTION

The development of cellular broadband communication technology, such as 4G/5G, has enabled the networking of automobiles as mobile terminals within the Internet of Vehicles (IoV) systems to facilitate the expansion of intelligent services [1]–[3]. According to the forecast report issued jointly by the Global System for Mobile Communications (GSMA) and SBD Automotive, the global IoV market can obtain a compound growth rate of approximately 25%, which is a potentially very large market. Moreover, predictions indicate that the IoV market value will reach 358 billion US dollars by the year 2020. However, the full state monitoring of automobiles, including real-time vehicle trajectory data, is expected to become essential for providing fully intelligent services that greatly enhance the safety and security of users. In this regard, Intel's CEO estimated that the total number of status sensors in an automobile running continuously will generate 4 TB of daily state monitoring data after only five years. Therefore, the scale of monitoring data will increase dramatically with an increasing number of online vehicles, which will critically affect the available IoV system

The associate editor coordinating the review of this manuscript and approving it for publication was Honghao Gao.

resources, such as the transmission network bandwidth, disk storage capacity, and number of disk I/O operations. This issue can be addressed effectively by implementing high efficiency data compression technology to the tradeoff of the computation, bandwidth cost, latency, and particularly for real-time vehicle trajectory data.

Real-time automobile state monitoring data in IoV systems include multiple features, such as small fields and short periods, and therefore involve severe space and time sensitivities. However, classifying and processing these data according to IoV features and application demands will substantially reduce the requirements for network bandwidth, disk I/O, and disk capacity, thereby greatly improving real-time processing capability. As such, the time-variant nature of vehicle trajectory data has performed research regarding relevant high efficiency compression methods one of the key components facilitating full automobile state data management in IoV systems. Existing vehicle trajectory compression methods are classified into three types [4], [5]: line simplification, road network-based compression, and semantic information-preserving compression. However, these compression methods have many disadvantages such as highly complicated time-space relationships, relatively small compression ratios, loss of driving direction information, and severe distortions in the compressed trajectory data relative to the actual trajectories [6]–[8]. Moreover, the application platforms of actual IoV systems, for example ACTIA Smart-Cloud, have not fully considered IoV system resources and the characteristic features of real-time automobile state monitoring data. Therefore, these systems have been unable to adapt to the bottleneck conditions of large-scale IoV data transmission and storage.

Accordingly, the present work considers IoV system resources and real-time automobile state monitoring data features to propose a trajectory data compression algorithm using a combination of road width W, vehicle driving direction D, and swerve angle A (WDA). It adopts different compression methods for rectilinear and swerving vehicle trajectories. This allows for highly efficient compression and high similarity between compressed and original trajectories. In addition, the algorithm preserves direction information. Finally, it gives a method for a dynamic computing compression period. The proposed algorithm optimizes the vehicle terminal data processing logic and simultaneously substantially improves the efficiency of data compression, transmission, and storage.

The remainder of this paper is structured as follows. In Section II, we review existing related compression methods. We then present an overview of the WDA-based direction preserving trajectory compression approach in Section III. Next, we present the detailed compression algorithm that includes two different trajectory compression strategies in Section IV. The results of our experimental study employing a Microsoft GeoLife dataset are reported in Section V. Finally, Section VI concludes this article, and presents some directions for future research.

II. RELATED WORKS

As discussed, existing compression algorithms for trajectory data include line simplification, road network-based compression, and semantic information compression. Most standard trajectory data compression algorithms are classified as line segment simplification types. The basic principle of line simplification compression is to replace actual trajectories with straight line segments between two location points that approximately represent the original trajectory between those two points. Muckell *et al.* [9] presented a new approach that inserts points from a trajectory into a priority queue, and the priority of each point is set to an upper bound that is based on the error introduced by the removal of that point. This technique can be applied to strike a balance between the compression ratio and the deviation between the compressed trajectories and the original trajectories (i.e., the trajectory error). Liu *et al.* [7] and Deng *et al.* [10] proposed an optimized trajectory streaming data compression approach denoted as direction-preserving trajectory simplification (DPTS+) to address the loss of trajectory directions suffered by many compression algorithms. In addition to preserving trajectory directions during compression, the method introduced a new data structure denoted as a bound quadrant system that effectively reduced the compression time of online DPTS+. This study further explored time saving efficiencies by evaluating the feasibility of adopting contemporary general-purpose computing on a graphics processing unit. Nibali and Trajic [11] extended the delta compression approach to facilitate balancing between the compression ratio and the trajectory error. Liu *et al.* [12] proposed the amnesic bounded quadrant system (ABQS) online framework for adapting to resource-constrained environments by automatically adjusting the compression process to available storage resources and then compressing trajectories using trajectory error tolerances established according to the ages of the trajectories. Ke *et al.* [13] proposed a trajectory compression algorithm with interval bounds that guaranteed an error bounded compression result to achieve a better tradeoff between a high compression rate and a short operational time. However, as discussed, line simplification compression can suffer from serious disadvantages. In IoV applications, high vehicle speed or relatively low sampling frequency can result in the loss of key location point information. As a result, semantic deviations or severe distortions in the actual trajectories can occur, such as in the case of compressed vehicle trajectories crossing walls or lakes.

The increased prevalence of accurate data reflecting the structure of road networks has facilitated the development of trajectory compression algorithms based on road network structure. This method abstracts road networks into a directed graph, or digraph, and then maps a sequence of trajectory points onto the corresponding sequence in the digraph. Zhou *et al.* [14] proposed a framework for processing trajectory data, where the framework included three modules: a data distribution module, a data transformation module, and a compression-aware I/O module. The data distribution

module accounted for data locality, load balancing, and scalability, while the data transformation module applied a parallel strategy for facilitating an easily implemented parallelization. Simultaneously, it remained monotonic and improved compression performance significantly. In addition, the I/O performance improved by providing effective decision support in the application of trajectory data compression. Han *et al.* [15] compressed GPS trajectory data using urban road network information, and decomposed trajectories into spatial paths and temporal sequences. Lossless compression was applied to the spatial paths while lossy but error-bounded compression was applied to the temporal sequences in parallel processes. Trajectory compression algorithms based on road network structure generally provide a good compression ratio. However, the actual trajectories can be subjected to rather large distortions when a vehicle overtakes another vehicle, changes lanes, or temporarily swerves during transit. As such, obtaining compressed data in this manner that effectively reflects actual trajectory conditions can be challenging. In addition, the process of abstracting the road network structure into a digraph requires considerable computational capacity. Therefore, the application of these algorithms to resource-constrained mobile vehicle terminals is generally not possible.

The degree to which compressed longitude and latitude trajectory data are discretized tends to be very high, and the semantic information it contains is typically unclear. However, some compression algorithms that preserve semantic information have emerged in recent years. For example, a compression algorithm has been proposed that combines extensive semantic interest data, such as points of interest, road segments, events, road grades, directions, and speeds, to describe trajectory information [16]. While these types of algorithms can preserve rather abundant semantic information in a compressed state, the original longitude and latitude location point data are lost entirely, which negatively affects the query ability of location-based service (LBS) applications [17]. In addition, the generation of semantic information requires an excessive amount of time, such that the real-time performance of these algorithms is not sufficient for application to IoV intensive streaming data compression.

The above discussion amply demonstrates the two principle defects of existing trajectory compression algorithms. First, virtually no efficient means are available for reducing the compression time and compression ratio under the condition of a resource-constrained mobile vehicle terminal. In particular, for IoV applications involving high vehicle speeds and relatively low sampling frequencies, trajectory compression methods with short compression times, a small compression ratio, and low resource consumption are urgently needed. Second, key location points are often lost in the compressed trajectories, which reduces the scope and reliability of the semantic information contained therein. This paper addresses these issues using the proposed WDA-based trajectory compression algorithm.

III. OVERVIEW OF THE WDA-BASED TRAJECTORY COMPRESSION ALGORITHM

A. THE CORE IDEA OF THE WDA ALGORITHM

The effective storage of vehicle trajectory data requires that the volume of trajectory data be reduced by compression to the greatest extent possible, while maintaining an acceptably short compression time and retaining a sufficient amount of the original trajectory information to ensure that the compressed data provides adequate accuracy, reliability and trajectory semantics. In particular, data reflecting the driving directions of vehicles and significant points of interest (POI) (e.g., turning around) should be retained. Moreover, an effective compression algorithm should be independent of road network structure and historical trajectory information to ensure convenient and flexible integration with various application systems. To support these requirements, the proposed WDA compression algorithm determines whether vehicle trajectory location points are rectilinear within a permitted error range according to the principle variables W and D, and then determines the trajectory data points where a vehicle swerves significantly according to the third principle variable A. Here, W can be set dynamically according to specific road classification standards. As such, the proposed algorithm considers the constraints of road shape on vehicle trajectories according to whether the trajectory forms an approximately straight line or a curve indicative of a swerving trajectory. In addition, the algorithm preserves information regarding D. Finally, the WDA compression algorithm adopts different compression methods for rectilinear and swerving vehicle trajectories. This allows for highly efficient compression and high similarity between compressed and original trajectories.

B. WDA-BASED TRAJECTORY COMPRESSION ALGORITHM

Algorithm 1 presents the core functions of the WDA compression algorithm in the form of pseudocode. Here, the algorithm receives the original sampled trajectory data point sequence $p_1, p_2, \ldots, p_n$, where the i-th data point contains coordinate values x_i and y_i reflecting the longitude and latitude of the vehicle, respectively, as well as its corresponding sampling time stamp t_i, i.e., $p_i = (x_i, y_i, t_i)$. The output of the algorithm is a sequence linked list of storage units $U_k(p_j, p_n)$ containing the information of two trajectory data points. The two vehicle trajectory location points that are first collected, i.e., (p_0, p_1), are used to construct the first storage unit U_0 and straight driving line (*DriveLine*). Beginning with p_2, the distance from each point to the straight driving line (*PiLineDistance*) and the swerve angle (*SwerveAngle*) between the current driving direction line (*CurrentDriveDirectionLine*) and the straight driving line are calculated. Then, the algorithm determines whether the current vehicle is driving in an approximately straight line or in swerving state according to *PiLineDistance* and *SwerveAngle*.

Algorithm 1 High-Level Pseudocode for the WDA Compression Algorithm

Input: Original sampled trajectory data points $(p_0, p_1, \ldots, p_n)$

Output: Storage unit $(U_0, U_1, U_2, \ldots)$ representation of the compressed trajectory data points

```
Funciton Compress (p_0, p_1, ..., p_n) begin
    LineDriveRegionWidth ⟵ W;
    DriveLine ⟵ CalculateDLine(p_0, p_1);
    For i ⟵ 2 to n do
        CurrentDriveDirectionLine ⟵
        CalculateCDLine(p_i − 1, p_i);
        PiLineDistance ⟵
        CalculatePLineDisatance(P_i, DriveLine);
        SwerveAngle ⟵ Calculate SwerveAngle
        (CurrentDriveDirectionLine, DriveLine);
        If (PiLineDistance < W/2) and
        (SwerveAngle < 90°) then
            UpdateStorageUnit(U_j, p_i);
        EndIf
        Else
            U_j(p_i, p_{i−1}) ⟵ CreateStorageUnit(p_{i−1}, p_i);
            DriveLine ⟵ CalculateDLine(p_{i−1}, p_i);
            return Storageunits(U_0, U_1, U_2, ...);
        EndElse
    EndFor
```

IV. WDA-BASED TRAJECTORY COMPRESSION STRATEGIES

The compression strategies applied by the WDA compression algorithm for rectilinear and swerving trajectories are discussed in the following subsections.

A. DATA COMPRESSION STRATEGY FOR RECTILINEAR TRAJECTORIES

Due to the constraints associated with location accuracy and road shape, the real-time location data sequence of a vehicle moving a distance ΔL over a relatively short period Δt on an approximately straight road can be regarded as a straight line. In this sense, vehicles typically move along an approximately straight line most of the time. Furthermore, the stored location information for driving along a straight road need only include the coordinate values and the corresponding times of the two end points of ΔL. Therefore, the key issue for effectively realizing the compression of spatial trajectory information is to approximate actual driving trajectories (including any minor nonrectilinear transits such as the overtaking of other vehicles and lane changing) as straight lines. Therefore, an algorithm for determining a vehicle's straight driving line and location information compression is proposed based on [18]–[22]. It determines whether the vehicles are running in the same line based on the maximum road width (W) and vehicle driving direction (D) and can preserve the information of the driving direction.

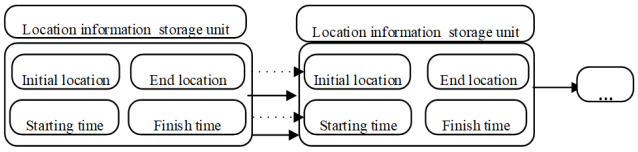

FIGURE 1. Compressed location information storage for approximately rectilinear vehicle trajectories.

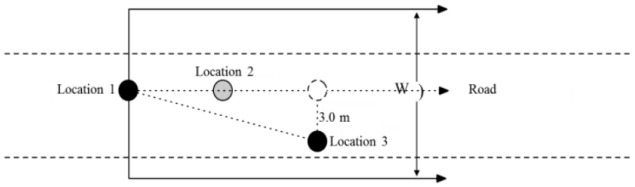

FIGURE 2. Relationship between the central driving line, straight line vehicle driving area, and location data points.[1]

After collecting trajectory location information, the first storage unit shown on the left side of Fig. 1 is constructed by storing the information of the first trajectory data point as the initial location and starting time, and storing the information of the second trajectory data point as the final location and finish time. The coordinate values of these two data points are employed to generate a single straight driving line that includes driving direction D. As shown in Fig. 2, this process begins with the first location data point and passes through the second location data point. This straight line along D is taken as the central line, and the area formed by the central line and width range W is denoted as the straight line vehicle driving area. The trajectory data in this straight line vehicle driving area is defined as follows.

The slope k of the straight line passing from Location 1 to Location 2 is

$$k = \frac{y_2 - y_1}{x_2 - x_1} \tag{1}$$

The intercept b is

$$b = y_2 - k * x_2 \tag{2}$$

The resulting linear equation of the straight trajectory is

$$kx - y + b = 0 \tag{3}$$

When a third point at Location 3 is obtained, the distance l from Location 3 to the central driving line is

$$l = \frac{|kx_1 - y_1 + b|}{\sqrt{k^2 + 1}} \tag{4}$$

If l is less than $W/2$, the data point at Location 3 lies within the straight line vehicle driving area.

[1]The two-dotted lines in the figure are the road edges, the central dotted line is the central driving line formed by location points 1 and 2, and the solid vertical lines are the edges of the straight line vehicle driving area. Here, W ($W = 18$ m) is the width between road edges, which can be set dynamically according to specific road classification standards. when Location 3 deviates from the initial central driving line formed by Locations 1 and 2 by a distance $l < W/2$.

According to the above straight line vehicle driving area definition, a vehicle is assumed to travel along a straight line while all uploaded location information resides in this area. As such, this straight line representation accommodates any minor nonrectilinear transits. In addition, this representation allows for the end location and finish time information to be replaced by the current location and time information. Therefore, the amount of memory space required does not increase while a vehicle is determined to be driving in a straight line because the information of only two location points and their corresponding times are recorded. This process is illustrated by Location 3 in Fig. 2. Here, l is less than $W/2$, and the two straight lines between Locations 1 and 2 and Locations 2 and 3 can be regarded as a single straight line. Therefore, the information at the end location and finish time of the first location information storage unit on the left side in Fig. 1, which was the last set according to Location 2, is updated according to Location 3.

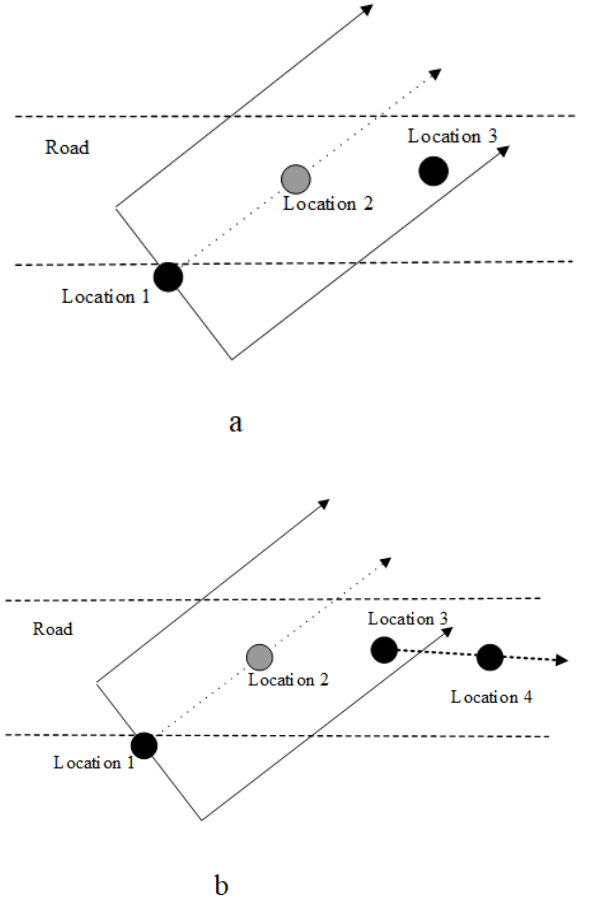

FIGURE 3. Deviant driving conditions, such as when a vehicle turns a corner.

However, deviant conditions can arise, as illustrated in Fig. 3(a). Here, the straight line formed by the vehicle starting location and the second location is not parallel to the road. As such, the generated straight driving line does not reflect the actual vehicle driving direction. Nonetheless, the third location falls within the straight line vehicle driving area, and the end location and finish time information of Location 2 in the location information storage unit are updated as Location 3. However, we note from Fig. 3(b) that Location 4 falls outside of the straight line vehicle driving area. Accordingly, the vehicle is considered as exhibiting a swerving behavior, and a new location information storage unit will be constructed. Then, the end location and finish time information in the previous location information storage unit pertaining to Location 3 are set to the initial location and starting time of the new location information storage unit. Subsequently, the end location and finish time of the new location information storage unit are set according to Location 4. This construction process is illustrated in Fig. 4. This process results in a newly generated straight driving line determined by Locations 3 and 4, as shown in Fig. 3(b), and subsequently collected location information is compressed according to this line.

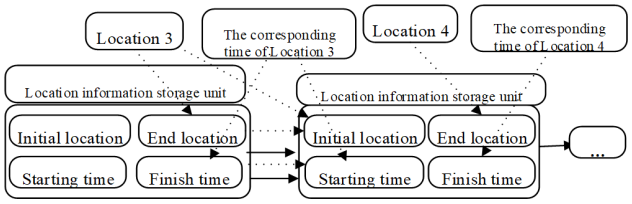

FIGURE 4. Compressed location information storage under the conditions illustrated in Fig. 3.

B. DATA COMPRESSION STRATEGY FOR SWERVING TRAJECTORIES

As demonstrated by the discussion above, the width of the straight line vehicle driving area is twice as large as the maximum road width. Figure 5 presents a condition in which a vehicle conducts a substantial swerving operation and reverses its direction of motion. Under the condition illustrated in the figure, the vehicle continues to reside within the previous straight line vehicle driving area while conducting the swerving action and after reversing its direction. Here, we note that the distance between Location l and each end location first increases and then decreases. The locations from Location i to Location m successively cover Location h. As a result, only Locations l and m are saved in the location

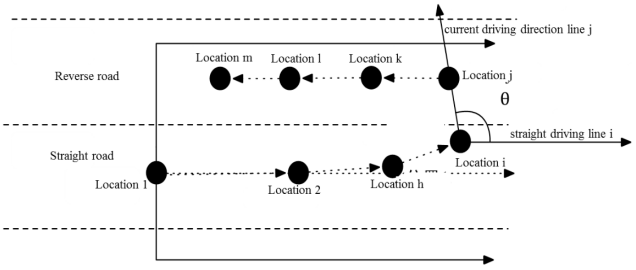

FIGURE 5. Location point relationships under a substantial vehicle swerving trajectory.

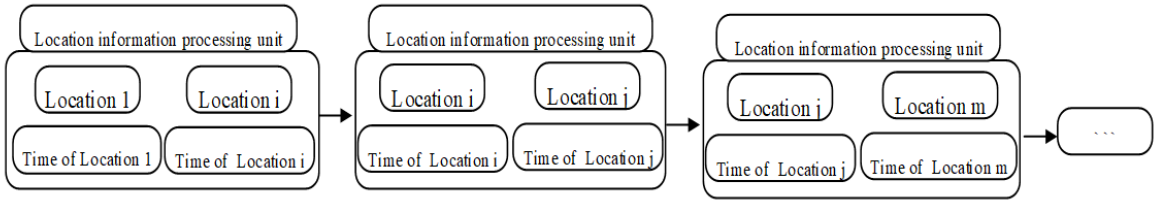

FIGURE 6. Compressed location information storage under the conditions illustrated in Fig. 5.

information storage unit, thereby forming a straight line from Location l to Location m. This trajectory obviously deviates from the actual driving trajectory, which will introduce substantial distortion in the trajectories after compression. The present work addresses this problem by introducing the swerve angle θ to determine when a vehicle conducts a swerving operation.

According to Fig. 5, θ is the angle between the current driving direction line j formed from current location data point (i) and the last location data point (j), and previously defined straight driving line i. As such, θ is defined as

$$\theta = \left| arctan(\frac{k_i - k_j}{(1 + k_j * k_i)}) * \frac{180.0}{\pi} \right| \tag{5}$$

where k_i is the slope of straight driving line i, and k_j is the slope of the current driving direction line j, which is given as

$$k_j = \frac{y_j - y_i}{x_j - x_i} \tag{6}$$

Accordingly, if θ is greater than 90°, the vehicle is deemed to be engaging in swerving behavior, and a new location information storage unit is created. The location storage structure resulting from the conditions illustrated in Fig. 5 is shown in Fig. 6: The end location of the old location storage unit is regarded as the initial location of the new storage unit; the current location is regarded as the end location of the location storage unit; and the current time is regarded as the finish time of the location storage time. Moreover, when the vehicle runs outside the straight vehicle driving line area, the straight driving line is updated automatically. As such, the straight driving line is nearly equivalent to the road direction.

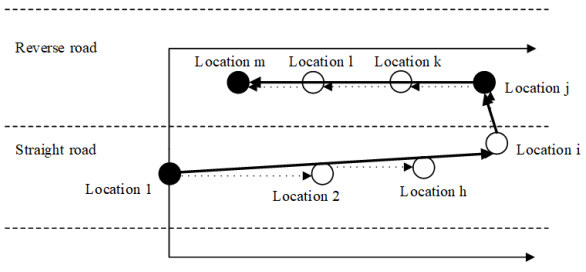

FIGURE 7. Decompressed trajectories obtained for the example illustrated in Fig. 5.

Figure 7 presents the path information (decompressed) of the compressed location information given in Fig. 5. As such, the decompressed trajectories are very similar to the actual trajectories in this example.

C. DYNAMIC COMPUTING COMPRESSION PERIOD

The compression period is the basis of the WDA compression algorithm: it compresses two or more trajectory points into one storage unit if their interval is less than or equal to the maximum compression period and meets compression conditions. Conversely, a new storage unit is created. The maximum compression period can be dynamically computed using the terminal upload frequency and the network delay. The relation of the maximum compression period, new storage unit, terminal upload period and network delay is shown in Fig. 8.

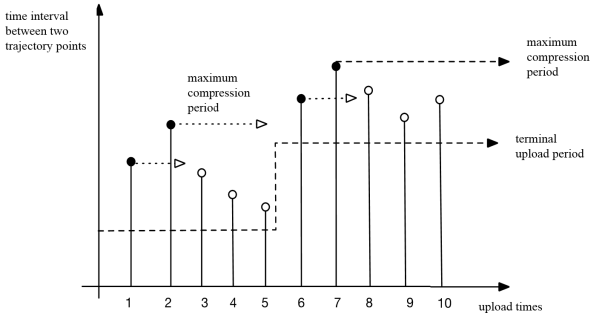

FIGURE 8. The relation of the maximum compression period and terminal upload period.

The length of the line segment ending with a solid or hollow circle denotes the total data upload delay. The length of the line segment above the terminal upload period denotes the network delay. The line segment ending with the solid circle denotes that the total delay exceeds the compression period. Therefore, a new storage unit must be created. The dotted arrow from the solid circle to the right indicates the data upload range that can be covered by the updated maximum compression period. According to their relationship, the maximum compression period can be computed using formula 7.

$$ct_n = \begin{cases} ct_n * \delta + (1 - \delta) * t & t > ct; \\ t & t \leq ct. \end{cases} \tag{7}$$

where ct_n denotes the newly estimated compression period, ct_o denotes the current maximum compression period, t denotes the total delay of current data upload, and δ is

TABLE 1. Compression ratios and compression times of the WDA and TRAJIC compression algorithms for six typical trajectories.[2]

GeoLife Test Set	Number of original trajectory location data points (data volume)	Number of compressed trajectory location data points (data volume)		Compression ratio[1] (%)		Compression/ decompression time[2] (ms)	
		WDA	Trajic	WDA	Trajic	WDA	Trajic
G1	1004(24096)	38	85(2032)	3.8	8.5	0.26	0.57/0.17
G2	2655(63720)	104	191(4575)	3.9	7.2	0.41	1.33/0.40
G3	5157(123768)	198	382(9174)	3.8	7.4	0.62	2.15/0.84
G4	8545(205104)	262	327(7842)	3.1	3.8	0.80	2.95/1.03
G5	12066(289584)	102	923(22147)	0.8	7.6	0.42	3.88/1.43
G6	16613(398712)	1462	641(13871)	8.8	3.9	4.36	5.82/2.00
Average	7673(184164)	361	393(9940)	4.03	6.4	1.15	2.78/0.98

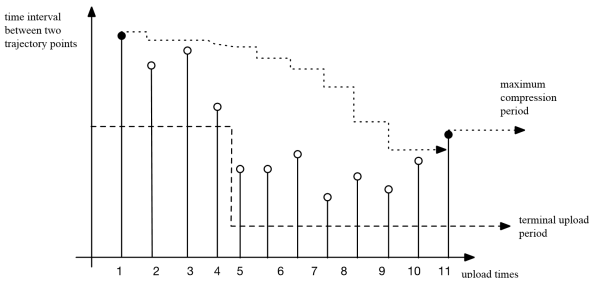

FIGURE 9. The dynamic adjustment process of the maximum compression period.

the smoothing factor, where δ is 7/8. The dynamic adjustment process of the maximum compression period is shown in Fig. 9.

V. EXPERIMENTS

We evaluated the performance of the proposed WDA-based trajectory compression algorithm according to three aspects, including the compression time, compression ratio, and similarity between the compressed and original trajectories. In addition, we compared its performance with that of the Trajic compression algorithm [11]. The Trajic algorithm considers the residual of the coordinate values between the predicted trajectory points and actual trajectory location points, and considers other details, such as residual encoding. Thus, the Trajic algorithm requires a specific decompression algorithm for generating trajectory data from the compressed trajectory data.

We compiled the C++11 code of the WDA and the Trajic compression algorithms using the GNU compiler collection (GCC) version 5.4.0 with the O3 optimization flag. The results were produced on an Intel® Xeon® CPU E5-2680 v3 processor clocked at 2.5 GHz with 2 GB of RAM,

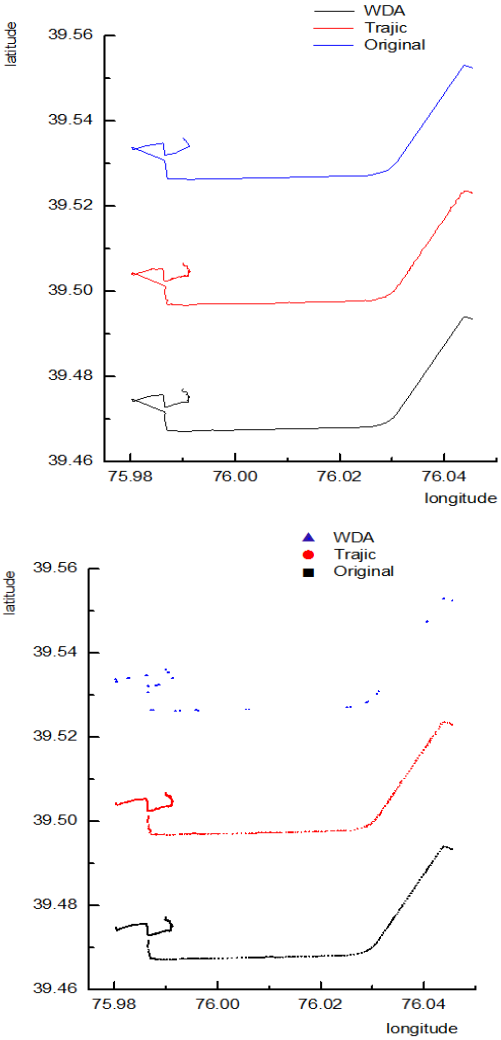

FIGURE 10. Comparison of the original G1 trajectory with the compressed trajectories.

[2]1. The output volume of the WDA algorithm consists of the number of compressed longitude and latitude coordinates of trajectory location data points. The output volume of the Trajic algorithm consists of the number of compressed residuals between two location points. Therefore, the numbers of location data points after and before compression are employed for calculating the compression ratio of the WDA algorithm, while the compression ratio of the Trajic algorithm is calculated as the volume of compressed data divided by the original data volume. 2. The Trajic algorithm requires a specific decompression algorithm for generating trajectory data.

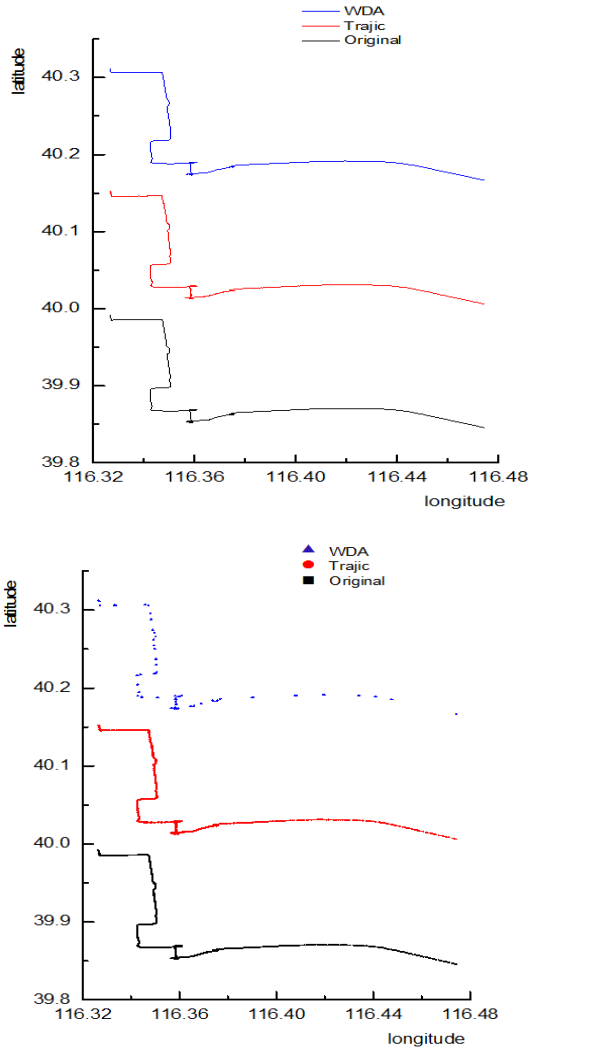

FIGURE 11. Comparison of the original G2 trajectory with the compressed trajectories.

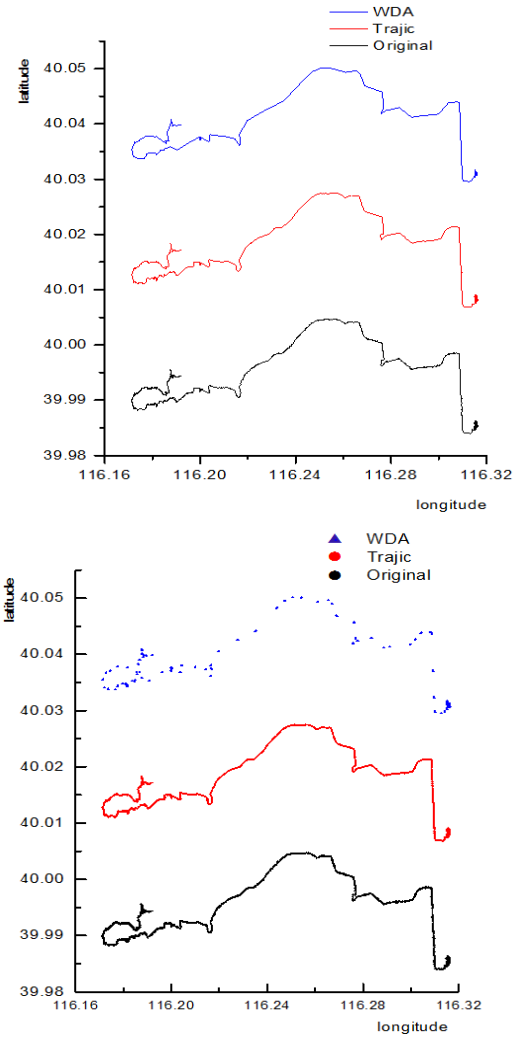

FIGURE 12. Comparison of the original G3 trajectory with the compressed trajectories.

and running Ubuntu Linux. The test data were obtained from the vehicle trajectory data of the Microsoft GeoLife project [23]. The GeoLife data contains 17,621 trajectories with an average sample rate of 1 sample per 3 s taken with a variety of GPS devices. To better test the practicability of the WDA algorithm, the selected test data must have good representativeness in two aspects: trajectory shape and number of trajectory points. According to the standard, six typical vehicle trajectory data were selected for testing and analysis. Based on the features of selected trajectory data, the spatial error of the trajectory data was controlled under 20-25 m, and the temporal error was less than 1-3 s. Figures 10-15 present comparisons of the six original trajectories and location points with their compressed trajectories and location points obtained using the Trajic and WDA compression algorithms. Here, the stacked lines that are offset along the ordinate axis direction of the graphs given on the left-hand sides represent the similarity of the original trajectories and the compressed trajectories. The scatter diagrams given by the graphs on the right-hand sides represent the location points of the original trajectories and those of the compressed trajectories. The performance test results of the WDA and Trajic compression algorithms on these six trajectories are presented in Table 1.

A. COMPRESSION TIME

The results in Table 1 indicate that the WDA algorithm performs obviously better than the Trajic algorithm for both single trajectory and the average compression time. The average compression time of the Trajic algorithm is approximately 2.4 times greater than that of the WDA algorithm (i.e., 2.78/1.15). Moreover, the WDA algorithm has a more obvious advantage from the standpoint of compression/decompression time. The average compression/decompression time of the Trajic algorithm is approximately 3.3 times greater than that of the WDA algorithm. Here, we note that the complex predicting and encoding/decoding technologies of the Trajic algorithm require a considerable amount of computation time. In contrast,

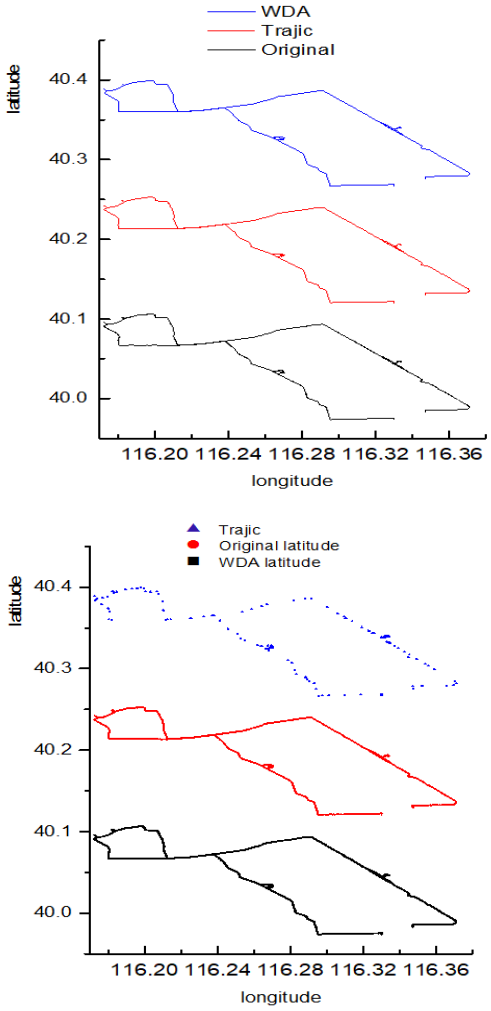

FIGURE 13. Comparison of the original G4 trajectory with the compressed trajectories.

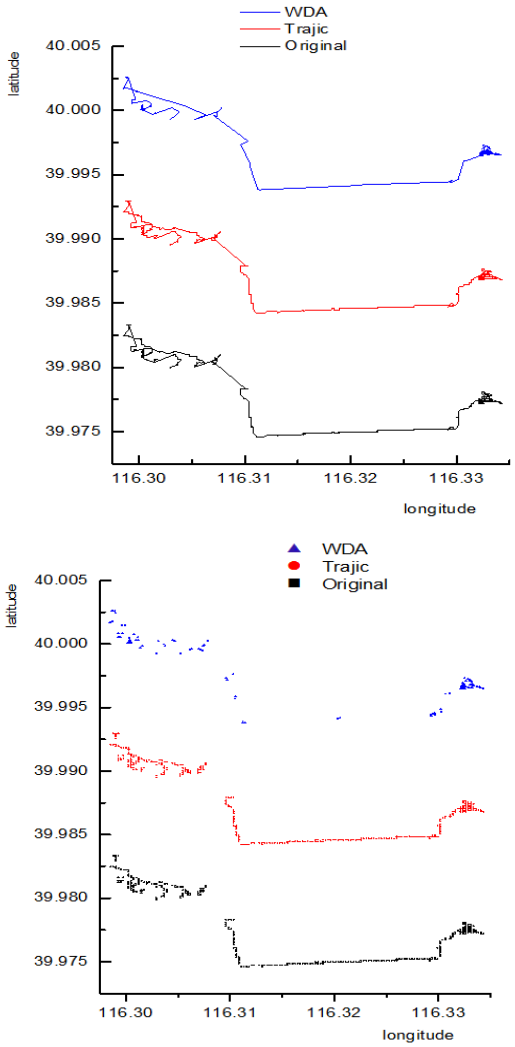

FIGURE 14. Comparison of the original G5 trajectory with the compressed trajectories.

the compression process of the WDA algorithm and its output are both directly related to the actual location data points, which avoids complex encoding/decoding operations. Therefore, the additional time required for calculating the residual of the coordinate values between the predicted trajectory points and actual trajectory location points as well as the time required for encoding the data are why the Trajic algorithm requires much greater compression time than the WDA algorithm.

Figure 16 presents a plot of the compression time required by the WDA and Trajic algorithms as a function of the number of location data points in the six original trajectories. As shown in the figure, the compression times for both the WDA and Trajic algorithms can increase approximately linearly with an increasing number of location data points in the original trajectories, while the compression time for the WDA algorithm increases at a slower rate. We also note from Figs. 10-15 and Table 1 that the complexity of a trajectory has a substantial effect on the compression time required by the WDA algorithm. Naturally, the compression scheme employed by the WDA algorithm causes the compression time to decrease as the trajectory increasingly approaches a straight line. As such, the compression time increases with increasing trajectory complexity. For example, the G5 trajectory represents a fairly straight trajectory for which the WDA algorithm compresses quickly, while the G6 trajectory includes a high proportion of swerving trajectories, and the WDA algorithm requires much greater computational time to evaluate and track the swerving direction information.

B. COMPRESSION RATIO AND THE SIMILARITY BETWEEN COMPRESSED TRAJECTORIES AND ORIGINAL TRAJECTORIES

The compression ratio refers to the ratio of the compressed trajectory location data volume N to the original trajectory location data volume M as a percentage (i.e., N/M Ã- 100%). From Table 1, we note that the average compression ratio of

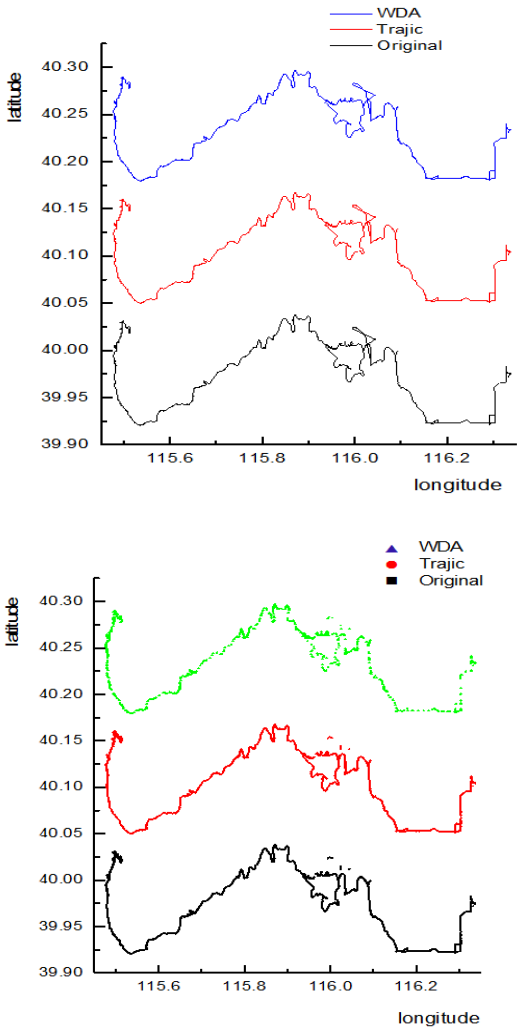

FIGURE 15. Comparison of the original G6 trajectory with the compressed trajectories.

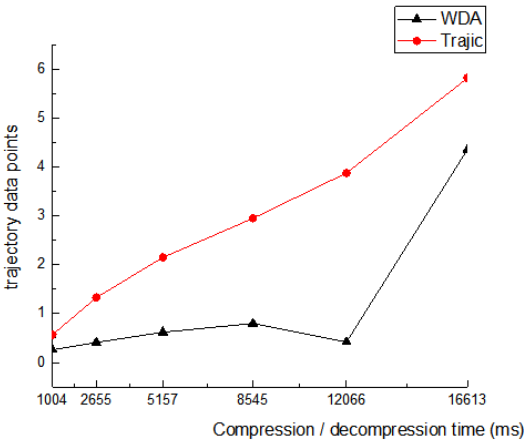

FIGURE 16. Compression time with respect to the volume of trajectory data.

the WDA algorithm is 62.97% that of the Trajic algorithm. Therefore, the compression performance of the WDA algorithm is significantly greater than that of the Trajic algorithm. Moreover, the compression ratios obtained by both algorithms are stable. For rather complex trajectories, particularly for trajectories with continuous sharp swerving or continuous left and right vehicle motions, the compression ratio of the Trajic algorithm is relatively stable.

Regarding the similarity between uncompressed and compressed trajectories, the WDA and Trajic algorithms both satisfy the requirement for low trajectory error. For common trajectories such as G1-G3 shown in Figs. 10-12, the trajectories compressed by the WDA and Trajic algorithms both obtain ideal effects. For rather complex trajectories, such as G5-G6, the trajectory errors of the compressed trajectories are rather low, which demonstrates that the Trajic algorithm has a good prediction function. In addition, we also note for trajectories G5 and G6 shown in Figs. 14 and 15, respectively, the WDA algorithm loses a greater number of significant trajectory points than the Trajic algorithm, leading to distortion in a portion of the compressed trajectory or resulting in a high compression ratio,which occurs for two different reasons. First, the WDA algorithm considers a portion of the trajectories exhibiting abrupt swerving or oscillating motion within a continuous small range as straight lines, which leads to a low compression ratio that is accompanied by trajectory distortion. Second, the WDA algorithm preserves numerous location data points at locations representative of swerving trajectories to guarantee low trajectory error for swerving trajectories, which results in a relatively high compression ratio.

VI. CONCLUSION

The proposed WDA compression algorithm focuses on road width, driving direction, and swerve angle, which are relatively macrotype information. WDA adopts different compression methods for rectilinear and swerving vehicle trajectories, which enables the algorithm to provide a relatively high compression ratio and low compression/decompression time while satisfying the similarity of the compression trajectory. To evaluate the proposed methods, we used a Microsoft GeoLife dataset that contains all types of empirical vehicle trajectories. The selected typical dataset tests show that the proposed algorithm attains an average compression ratio of 4.03% for vehicle trajectory data, and the average compression time is 1.15 m/each. Therefore, the algorithm is well suited for the compression demands of IoV streaming trajectory compression applications with constrained resources, intensive data, and the requirement for near real-time performance.

While the compression ratio and compression time results given in Section V demonstrated that the WDA algorithm presently achieves rather excellent results, there are two problems to be solved. First, the present development of the WDA algorithm does not consider key location points with special semantic significance during the compression process, which may lead to the loss of semantic information. In general, semantic information mainly refers to POI and some special trajectory points. To retain more semantic information,

the more corresponding auxiliary information needs to be added. This information is often difficult to compress. Conversely, if a high compression ratio is required, some semantic information will be lost. Moreover, the compression of semantic information often takes more time and space. Therefore, it needs to find a better balance point. Second, the WDA algorithm requires a preset road width W, which limits its application. Finally, we note that the source code for a working implementation of the WDA compression algorithm may be found at https://github.com/CoreVista/WDA-Path-Compression.git

REFERENCES

[1] Y. Yin, L. Chen, Y. Xu, J. Wan, H. Zhang, and Z. Mai, "QoS prediction for service recommendation with deep feature learning in edge computing environment," in *Mobile Networks and Applications*. New York, NY, USA: Springer, Apr. 2019, pp. 1–11.

[2] S. LaValle, E. Lesser, R. Shockley, M. S. Hopkins, and N. Kruschwitz, "Big data, analytics and the path from insights to value," *MIT Sloan Manage. Rev.*, vol. 52, no. 2, pp. 21–32, Dec. 2011.

[3] H. Gao, W. Huang, X. Yang, Y. Duan, and Y. Yin, "Toward service selection for workflow reconfiguration: An interface-based computing solution," *Future Gener. Comput. Syst.*, vol. 87, pp. 298–311, Oct. 2018.

[4] J. Jiang and X. Wang, "Review on trajectory data compression," *J. East China Normal Univ.*, vol. 5, pp. 61–76, May 2015.

[5] G. Trajcevski, "Compression of spatio-temporal data," in *Proc. IEEE Int. Conf. Mobile Data Manage. (MDM)*, Porto, Portugal, Jun. 2016, pp. 4–7.

[6] N. Meratnia and A. de Rolf, "Spatiotemporal compression techniques for moving point objects," in *Proc. Int. Conf. Extending Database Technol. (EDBT)*. Berlin, Germany: Springer, Mar. 2004, pp. 765–782.

[7] J. Liu, K. Zhao, P. Sommer, S. Shang, B. Kusy, and R. Jurdak, "Bounded quadrant system: Error-bounded trajectory compression on the go," in *Proc. IEEE 31st Int. Conf. Data Eng. (ICDE)*, Seoul, South Korea, Apr. 2015, pp. 987–998.

[8] C. Long, R. C.-W. Wong, and H. V. Jagadish, "Direction-preserving trajectory simplification," *Proc. VLDB Endowment*, vol. 6, no. 10, pp. 949–960, Aug. 2013.

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

[10] Z. Deng, W. Han, L. Wang, R. Ranjan, A. Y. Zomaya, and W. Jie, "An efficient online direction-preserving compression approach for trajectory streaming data," *Future Gener. Comput. Syst.*, vol. 68, pp. 150–162, Mar. 2017.

[11] A. Nibali and Z. He, "Trajic: An effective compression system for trajectory data," *IEEE Trans. Knowl. Data Eng.*, vol. 27, no. 11, pp. 3138–3151, Nov. 2015.

[12] J. Liu, K. Zhao, P. Sommer, S. Shang, B. Kusy, J.-G. Lee, and R. Jurdak, "A novel framework for online amnesic trajectory compression in resource-constrained environments," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 11, pp. 2827–2841, Nov. 2016.

[13] B. Q. Ke, J. Shao, and D. X. Zhang, "An efficient online approach for direction-preserving trajectory smplification with interval bounds," in *Proc. IEEE Int. Conf. Mobile Data Manage. (MDM)*, Kaist, South Korea, May/Jun. 2017, pp. 50–55.

[14] Y. Zhou, Y. Zhang, Y. Ge, Z. Xue, Y. Fu, D. Guo, J. Shao, T. Zhu, X. Wang, and J. Li, "An efficient data processing framework for mining the massive trajectory of moving objects," *Comput. Environ. Urban Syst.*, vol. 61, pp. 129–140, Jan. 2017.

[15] Y. Han, W. Sun, and B. Zheng, "COMPRESS: A comprehensive framework of trajectory compression in road networks," *ACM Trans. Database Syst.*, vol. 42, no. 2, Jun. 2017, Art. no. 11.

[16] K. F. Richter, F. Schmid, and P. Laube, "Semantic trajectory compression: Representing urban movement in a nutshell," *J. Spatial Inf. Sci.*, vol. 2012, no. 4, pp. 3–30, Jun. 2012.

[17] Y. Yin, W. Xu, Y. Xu, H. Li, and L. Yu, "Collaborative QoS prediction for mobile service with data filtering and SlopeOne model," *Mobile Inf. Syst.*, vol. 2017, no. 3, Jun. 2017, Art. no. 7356213.

[18] H. Su, K. Zheng, K. Zeng, J. Huang, and X. Zhou, "STMaker: A system to make sense of trajectory data," in *Proc. VLDB Endowmentm*, vol. 7, no. 13, pp. 1701–1704, Aug. 2014.

[19] G. Zhu and B. Wang, "Switched optimization algorithm of piecewise contractive path," *J. Comput. Appl.*, vol. 32, no. 6, pp. 1700–1703, Jun. 2013.

[20] R. Seidel and M. Sharir, "Top-down analysis of path compression," *SIAM J. Comput*, vol. 34, no. 3, pp. 515–525, Jun. 2006.

[21] Z.-Y. Cai and L. Li, "Research of spatiotemporal indexing mechanism based on snapshot-increment," *Acta Geod. Cartogr. Sin.*, vol. 34, no. 3, pp. 257–261, Jun. 2005.

[22] P. Q. Jin, X. Zhang, and L. H. Yue, "NBR-tree: A novel spatio-temporal index for urban traffic networks," *Geom. Inf. Sci. Wuhan Univ.*, vol. 35, no. 2, pp. 147–151, Feb. 2010.

[23] Microsoft. *GeoLife GPS Trajectories*. Accessed: Aug. 9, 2012. [Online]. Available: http://research.microsoft.com/en-us/downloads/b16d359dd164-469e-9fd4-daa38f2b2e13

HUIBING ZHANG was born in Nanyang, China, in 1976. He received the B.S. and M.S. degrees in computer science and technology from the Guilin University of Electronic Technology, Guilin, China, in 2007, and the Ph.D. degree in computer science and technology from the Beijing University of Technology, Beijing, China, in 2012. From 2012 to 2016, he was a Lecturer with the Guilin University of Electronic Technology, where he has been an Associate Professor with the Guangxi Key Laboratory of Trusted Software, since 2016. His research interests include trust evaluation and management in the Internet of Things, and social computing.

XIAOLI HU was born in Suqian, China, in 1978. She is currently a Lecturer with the Guilin University of Electronic Technology. Her research interests are social networking and trust computing.

WEIHUA BAI received the M.Sc. degree from the School of Computer Science, South China Normal University, in 2006, and the Ph.D. degree from the School of Computer Science and Engineering, South China University of Technology, China, in 2017. He is currently an Associate Professor with Zhaoqing University. His research interests include cloud computing, parallel scheduling, and software architectures. He is a member of the China Computer Federation.

HONGBO ZHANG, photograph and biography not available at the time of publication.

YA ZHOU was born in 1966. She received the B.S. degree in computer application from the Guilin University of Electronic Technology, Guilin, China, and the M.S. degree in computer software theory from Fudan University, Shanghai, China. She is a CCF Senior Member (09834S), a School Demonstration Professor, and a Famous Teacher in Guangxi. Her current research interests include massive data management and analysis, and peer-to-peer networks technology.

YUMING LIN was born in 1978. He received the B.S. and M.S. degrees in computer science and technology from the Guilin University of Electronic Technology, Guilin, China, and the Ph.D. degree in computer application from East China Normal University, Shanghai, China. He is currently an Associate Professor of computer sciences with the Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology. His current research interests include opinion mining, knowledge graph, and massive data management.