electronics

MDPI

Review

Trajectory Data Preprocessing: Methods and Models

Peiyu Li [1,2], Zhao Tian [3], Yanfang Yang [4,5] and Yusong Lin [3,6,*]

[1] School of Computer and Artificial Intelligence, Zhengzhou University, Zhengzhou 450001, China; lpy@haust.edu.cn
[2] Office of Information Construction and Management, Henan University of Science and Technology, Luoyang 471023, China
[3] School of Cyber Science and Engineering, Zhengzhou University, Zhengzhou 450002, China; tianzhao@zzu.edu.cn
[4] Key Laboratory of Transport Industry of Big Data Application Technologies for Comprehensive Transport, Beijing Jiao Tong University, Beijing 100044, China; yangyf@motcats.ac.cn
[5] China Academy of Transportation Sciences, Beijing 100029, China
[6] Collaborative Innovation Center for Internet Healthcare, Zhengzhou University, Zhengzhou 450000, China
* Correspondence: yslin@ha.edu.cn

Abstract

Trajectory data from GPS and sensors are increasingly available, necessitating effective preprocessing techniques for data mining. To systematically review the methods and models for trajectory data preprocessing, we conducted a systematic literature search in IEEE Xplore, Association for Computing Machinery Digital Library (ACM DL), Scopus, Web of Science, and Transport Research International Documentation published over the past several decades, using keywords related to trajectory data preprocessing. The studies were screened and selected based on predefined inclusion and exclusion criteria. We included 138 studies, summarizing techniques in data cleaning, compression, segmentation, and map matching. Key algorithms and their performance are compared. This review synthesizes current preprocessing methods and identifies future research directions, including real-time processing, semantic labeling, and privacy protection.

Keywords: trajectory data preprocessing; data cleaning; trajectory segmentation; trajectory compression; map matching

Received: 31 August 2025
Revised: 20 November 2025
Accepted: 23 November 2025
Published: 28 November 2025

Citation: Li, P.; Tian, Z.; Yang, Y.; Lin, Y. Trajectory Data Preprocessing: Methods and Models. *Electronics* **2025**, *14*, 4694. https://doi.org/10.3390/electronics14234694

1. Introduction

Trajectory data has gained significant importance across various domains, including smart transportation, urban planning, and location-based services. Advances in technologies such as satellite navigation systems, geographic information systems (GISs), sensor networks, wireless communication, and the mobile Internet have led to the generation of large volumes of spatiotemporal trajectory data. These massive datasets provide a solid foundation for applications in intelligent transportation, smart logistics, and academic research. However, raw trajectory data often contains errors, noise, and inconsistencies, which complicate analysis and hinder the extraction of meaningful insights. Therefore, effective trajectory data preprocessing is essential for ensuring the accuracy and reliability of analytical outcomes [1].

As the cornerstone of trajectory mining, trajectory preprocessing plays a critical role in converting raw data into a structured format suitable for in-depth analysis and mining [2–4]. The preprocessing pipeline typically includes data cleaning, compression, segmentation, and map matching [5,6]. Trajectory data frequently includes inaccuracies, outliers, and

redundancies that can distort interpretation. Data cleaning, the initial step, involves detecting and removing outliers, correcting inconsistencies, and imputing missing values. This process enhances the data quality and supports trustworthy subsequent analysis. Data compression techniques are applied to reduce storage and computational demands while preserving essential information. By minimizing the data volume without substantial information loss, compression facilitates efficient transmission and analysis—especially vital when handling large-scale trajectory datasets. Data segmentation partitions trajectories into meaningful segments according to criteria such as time intervals, distance thresholds, or semantic labels. This division allows for a focused examination of specific segments, aiding in the identification of movement patterns, behaviors, and events—such as transport modes, trip duration, origins, and destinations. Common segmentation methods include clustering, threshold-based approaches, and probabilistic models. Map matching aligns trajectory points with an underlying road network or digital map. This process corrects spatial inaccuracies caused by GPS drift, multipath effects, or signal loss. Accurate map matching is crucial for reliable visualization and advanced analytical tasks, including route planning, traffic analysis, and location-based services.

Although there are many different approaches to trajectory preprocessing [7–12], there is no comprehensive review to summarize the entire technical system. To the best of our knowledge, this article is the first comprehensive overview of trajectory preprocessing techniques. This paper covers the entire scope of trajectory preprocessing, with a focus on methods and techniques. Existing surveys on trajectory data preprocessing have provided good summaries and classifications of existing methods. However, given the significant advancements made in this field over the past decade, these papers may not cover the latest technological developments. Therefore, the previous classifications of these methods may no longer accurately reflect the current state of the field and need to be to be re-summarized and updated. There are also some papers that focus on specific subfields of trajectory processing, such as surveys on trajectory compression methods or map matching techniques, but they are not comprehensive enough. Therefore, we believe that this article is a novel contribution and holds significant importance for the development of higher-level applications in trajectory management and mining. Additionally, this paper discusses the challenges and open research questions in trajectory data preprocessing and provides insights into potential future directions in this field. Table 1 compares and contrasts previous surveys with our own, highlighting the new areas covered in our work.

Table 1. Summary of contributions relative to previous trajectory data preprocessing surveys.

Survey	Data Cleaning	Data Compression	Data Segmentation	Map Matching	Datasets
(Lee and Krumm, 2011) [13]	√	√			
(Amigo et al., 2021) [14]		√	√		√
(Sun et al., 2016) [15]		√			
(Muckell et al., 2014) [16]		√			
(Chao et al., 2020) [17]				√	
(Huang et al., 2021) [18]				√	
(Hashemi and Karimi, 2014) [19]				√	
(Sousa et al., 2020) [20]	√	√			
This Survey	√	√	√	√	√

The remainder of this paper is structured as follows: Section 2 details our systematic review methodology. Section 3 provides an in-depth discussion of core preprocessing techniques, covering data cleaning (e.g., outlier detection and imputation), compression (e.g.,

line simplification and semantic compression), segmentation (supervised, unsupervised, and semi-supervised), and map matching (geometric, topological, probabilistic, and advanced methods). Section 4 summarizes public datasets, Section 5 explores future research directions, and Section 6 concludes the study.

2. Overview of Research Methods

2.1. PRISMA Declaration

This system evaluation strictly follows PRISMA 2020 (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) Guide. The research protocol has not been registered.

2.2. Retrieval Strategy

Our system searched the following electronic databases:

1. IEEE Xplore;
2. ACM Digital Library;
3. Scopus;
4. Web of Science;
5. Transport Research International Documentation.

The search time range is from 1 January 1973 to 31 August 2025. The search keywords include the following: trajectory data, GPS trajectory, preprocessing, data cleaning, compression, segmentation, and map matching.

2.3. Inclusion and Exclusion Criteria

The inclusion criteria are as follows: peer-reviewed papers published in English; focusing on trajectory data preprocessing methods; and provides an algorithm description or evaluation.

The exclusion criteria are as follows: Non-English; only abstract; non-trajectory data (such as images and text); and only involves applications rather than preprocessing methods.

2.4. Literature Screening and Data Extraction

Two authors independently conducted title/abstract screening, full-text evaluation, and data extraction. Disagreements can be resolved through discussion or a third author's decision. The extracted information includes the following: author, year, method type, algorithm name, performance indicators, etc.

2.5. Literature Screening Process

This systematic review was conducted in accordance with the PRISMA 2020 statement. The checklist has been completed, and the results of the literature search and the study selection procedure are detailed in the PRISMA flow diagram (Figure 1).

2.6. Substantiation of Comprehensive Coverage

To substantiate our claim of being a comprehensive review, we employed a systematic and reproducible search protocol. The search strings used were of the following form: ("trajectory" OR "GPS") AND ("preprocessing" OR "cleaning" OR "compression" OR "segmentation" OR "map matching"). This ensured broad coverage across all key subtopics. Furthermore, Table 1 has been designed as a quantitative comparison to prior surveys. It demonstrates that, while previous works often focus on one or two aspects of the preprocessing pipeline (e.g., compression alone, or map matching alone), this review is, to the best of our knowledge, the first to systematically integrate and compare methods across all four foundational pillars: cleaning, compression, segmentation, and map matching.

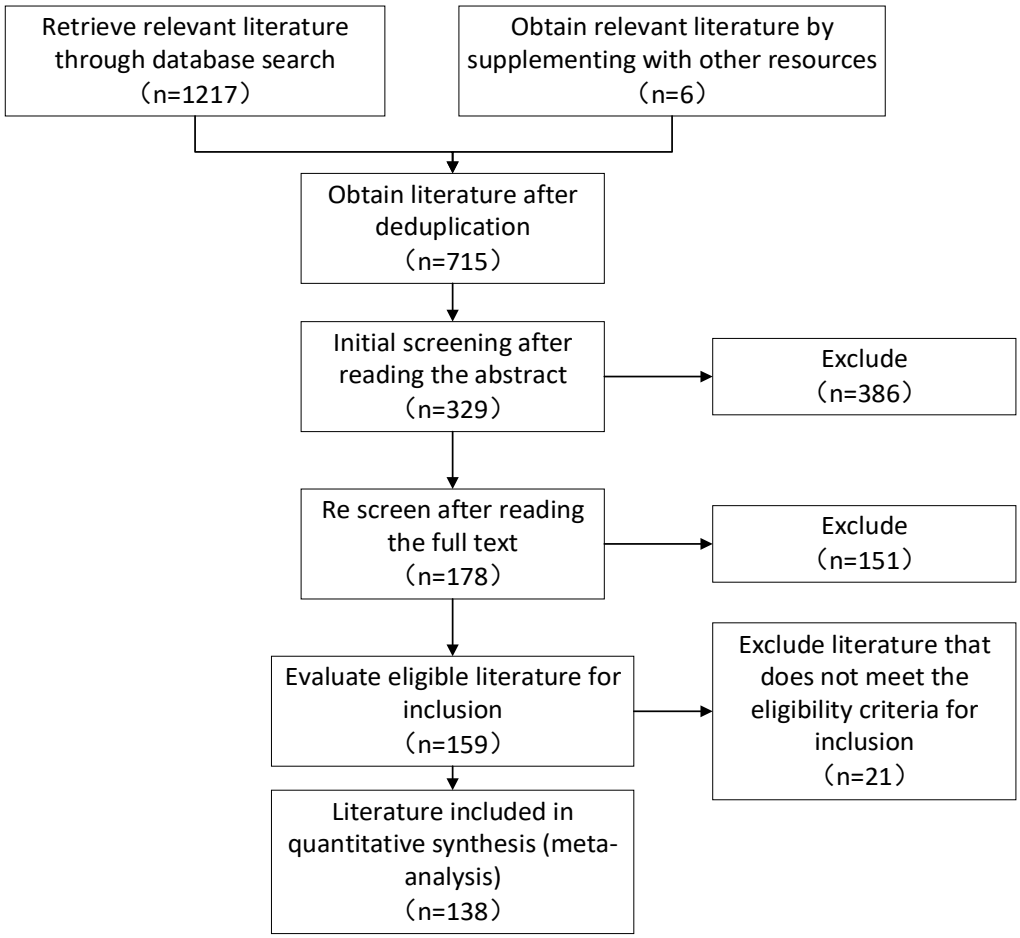


Figure 1. PRISMA flow diagram.

3. Trajectory Data Preprocessing Methods

Trajectory data contains rich spatio-temporal feature information, from which a large number of valuable patterns and knowledge can be extracted for intelligent transportation services. However, at the same time, the existence of poor data quality and some unique features in trajectory data greatly increases the difficulty of trajectory mining tasks. Therefore, it is necessary to use some algorithms to preprocess the trajectories in order to improve the quality of trajectory data, as is shown in Figure 2. Trajectory data preprocessing is a crucial step in trajectory data mining because it can have a significant impact on the quality and accuracy of subsequent analysis. It involves data cleaning, trajectory compression, segmentation, interpolation, and map matching.

3.1. Data Cleaning

Different types of trajectory data are inaccurate due to sampling accuracy, data format, sensor noise, and record point offset. The main purposes of data cleaning are as follows: These steps are typically applied in a logical sequence (1–4) to progressively refine data quality, though the order may be adjusted based on specific application needs:

1. Removing outliers: Outliers are data points that are significantly different from the rest of the data. They can be caused by errors in the GPS tracking device or environmental factors. Removing these outliers can improve the accuracy of the data.
2. Interpolating missing data: Sometimes, GPS tracking devices may lose signal or malfunction, resulting in gaps in the data. Interpolation generally estimates missing data points based on neighboring data points.

3. Smoothing data: Smoothing involves averaging out small variations in the data to remove noise. This can be carried out using a moving average or a low-pass filter.
4. Aligning data: If the GPS device and the data processing software are not synchronized, there may be time delays or offsets in the data. Aligning the data involves adjusting the timestamps to ensure that they are accurate.

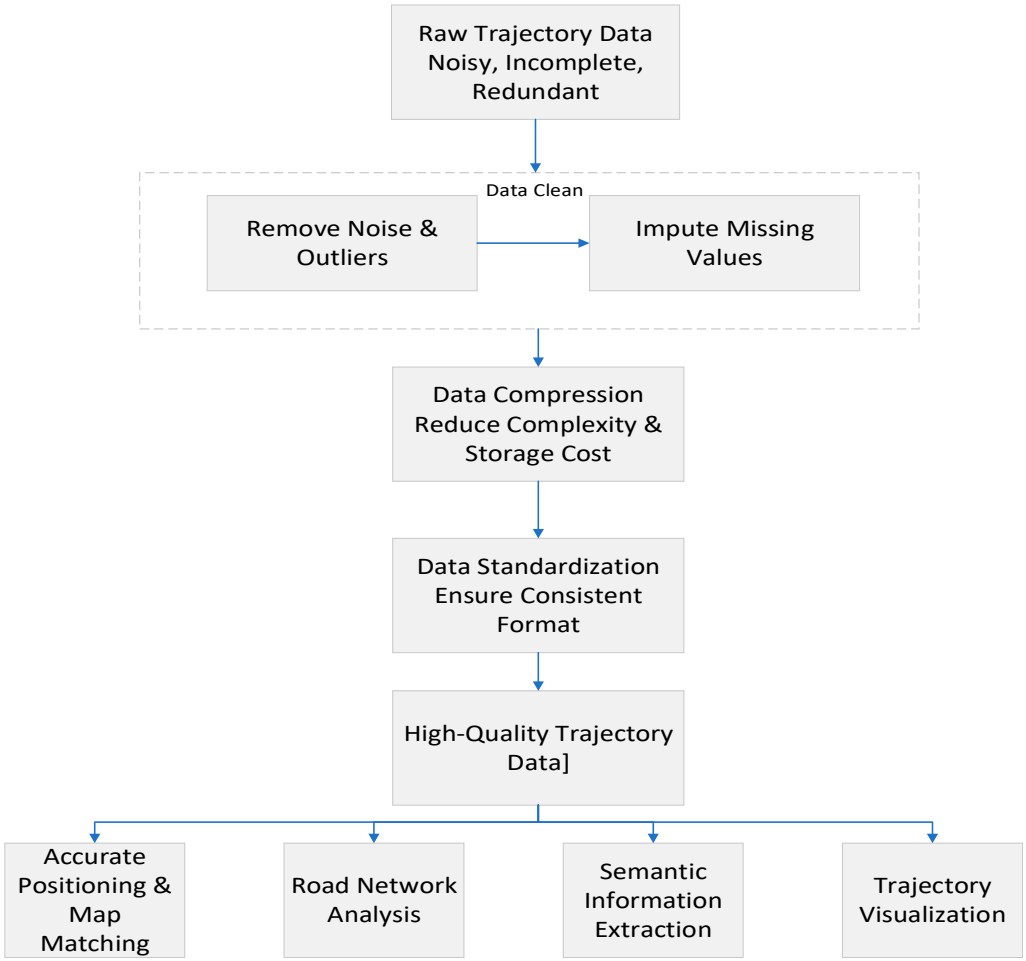


Figure 2. The significance of trajectory preprocessing.

3.1.1. Removing Outliers and Duplicates

Abnormal data can greatly affect the accuracy and computational efficiency of trajectory compression and mining. Therefore, it is necessary to employ methods to eliminate outliers before trajectory compression. The main methods for eliminating outliers are mainly filtering such as median or mean filtering [21–25], particle filtering [26], and machine-learning-based method [27,28].

Mean and median filters are fundamental techniques for smoothing trajectory data and removing isolated outliers. The mean filter replaces each point with the average value of its neighbors within a surrounding window. While simple, this method is sensitive to extreme outliers, which can skew the average and cause abnormal fluctuations in the processed trajectory. The median filter, which selects the middle value in the sorted window, offers stronger robustness against such extreme outliers [29]. In dense trajectories with infrequent noise, both filters are effective. However, their local and memoryless nature means they cannot utilize the motion context; consequently, they often fail to produce satisfactory results when dealing with consecutive noisy points or when the trajectory undergoes legitimate rapid changes.

The Kalman filter (KF) is employed when the trajectory cleaning task must account for the underlying dynamics of the moving object, a scenario where simple filters are inadequate. Unlike mean and median filters, the KF is a recursive estimator that combines a dynamic model (predicting the object's state, e.g., position and velocity) with a measurement model (incorporating noisy GPS observations) [13]. This structure allows it to smooth the trajectory based not only on current and past measurements but also on predictable motion patterns, making it particularly advantageous for handling continuous noisy points and providing optimal estimates in linear Gaussian systems. However, a key limitation of the standard KF is its assumption of linear motion; it struggles with the highly nonlinear dynamics typical of urban vehicle trajectories, such as sharp turns or stops.

Unlike the Kalman filter, particle filters do not require the dynamic model to be linear. As a result, particle filters have been widely applied in nonlinear and non-Gaussian systems. Particle filters approximate the actual probability distribution based on measurements by adjusting the weights of each particle and the positions of sampling points. During the prediction phase, each particle is propagated according to the state transition equation to obtain the predicted particles. Then, calculate the importance weights of the predicted particles and perform the normalization and importance sampling process. This makes it highly effective for complex motion patterns, such as those constrained to a road network or involving maneuvering [30].

Machine-learning-based method: These methods often employ algorithms such as regression and classification to process and clean data. These algorithms and techniques automatically clean and process data by learning patterns, features, and rules from the data. Zhang et al. proposed a vehicle trajectory data cleaning method using Support Vector Machines (SVM) and decision trees [31]. Li et al. proposed a data-driven trajectory cleaning method that extracts main paths from a large number of historical trajectories and derives a smooth path skeleton as anchor points. Safe zones are set around the skeleton anchor points to detect and clean abnormal data based on the reference of the safe zones [32].

3.1.2. Data Interpolating

Data interpolating refers to the process of estimating missing or irregularly sampled data points to create a complete and uniformly sampled trajectory. This step is crucial to ensure the accuracy and continuity of the trajectory data for further analysis and processing. Currently, research on data interpolation for trajectory data cleaning involves various techniques and methods. One commonly used approach is geometric-based interpolation, such as linear interpolation [33], spline interpolation [34], and polynomial interpolation [35]. These methods estimate the missing data points by fitting a curve or a line through the available data points. Another approach is trajectory interpolation based on spatio-temporal features. Algorithms in this category utilize techniques such as time series analysis and Kalman filtering.

Several research studies have extensively explored and compared these data interpolating techniques in the context of trajectory data cleaning. For instance, Markovsky and Dörfler proposed a data-driven dynamic interpolation and optimal approximation method [36]. Guo et al. proposed a time-based kinematic interpolation method that estimates interpolated positions and velocities by establishing the acceleration function of a moving object within one cycle. Based on the kinematic equations, the velocity function and position function are derived, allowing for the estimation of interpolated positions and velocities [37]. These studies provide valuable insights into the strengths and limitations of different data interpolating techniques for trajectory data cleaning, aiding researchers and practitioners in selecting appropriate methods for their specific application scenarios.

3.1.3. Data Smoothing

Data smoothing techniques aim to remove discontinuity and irregularity from trajectory data, transforming discrete trajectory segments and sampling points into a continuous trajectory sequence to represent the underlying trajectory more smoothly. The main goal of data smoothing, on the other hand, is to improve the visual appearance of the trajectory, making it smoother and more continuous for further analysis and applications. In terms of data smoothing techniques, one commonly used method is the moving average filter. This technique effectively eliminates high-frequency noise and reduces the impact of outliers. Other smoothing techniques include low-pass filtering [38], Savitzky–Golay filtering [39], Gaussian smoothing [40], and exponential smoothing [41]. The Kalman filter is widely used for trajectory data smoothing, as it estimates the true trajectory based on a dynamic state model and noise measurements.

3.1.4. Data Aligning

Data alignment technology focuses on unifying trajectory data from different sources or formats for further analysis and processing. Currently, there are several main techniques used for trajectory data alignment: time alignment, spatial alignment, and format alignment.

Time alignment involves adjusting the timestamps of different trajectory data to a unified time reference system. Common methods include timestamp synchronization and time calibration. One of the most challenging problems in such techniques is how to achieve online time series alignment, especially when there are overlapping regions and temporary stops in the trajectories [42,43]. Therefore, it is necessary to use incremental computing and real-time updating algorithms, as well as machine learning and deep learning techniques such as Bayesian estimation, to further improve the effectiveness and performance of online alignment.

Spatial alignment focuses on adjusting the spatial coordinates of different trajectory data to a common coordinate system. Common techniques include coordinate transformation and projection transformation. Common spatial alignment methods include nearest neighbor alignment, feature matching alignment, grid alignment, and geometric transformation alignment. The most challenging problem in spatial alignment technology is how to achieve large-scale alignment of multi-source data. This is because different trajectory data exhibit inconsistencies in spatial distribution, such as offsets or displacements, and the processing of massive data also imposes certain requirements on the efficiency of algorithms. Chen et al. proposed a large-scale parallel sub-trajectory alignment method based on parallel processing, which achieved efficiency and scalability of data alignment algorithms in large-scale trajectory processing platforms [44].

Format alignment involves converting trajectory data of different formats into a unified data format. This may include renaming fields, unit conversion, and data type conversion. In format alignment, one of the important problems that such algorithms need to solve is how to correctly understand the semantics of the data using artificial intelligence algorithms and find the corresponding relationship of the correct fields.

3.2. Trajectory Compression

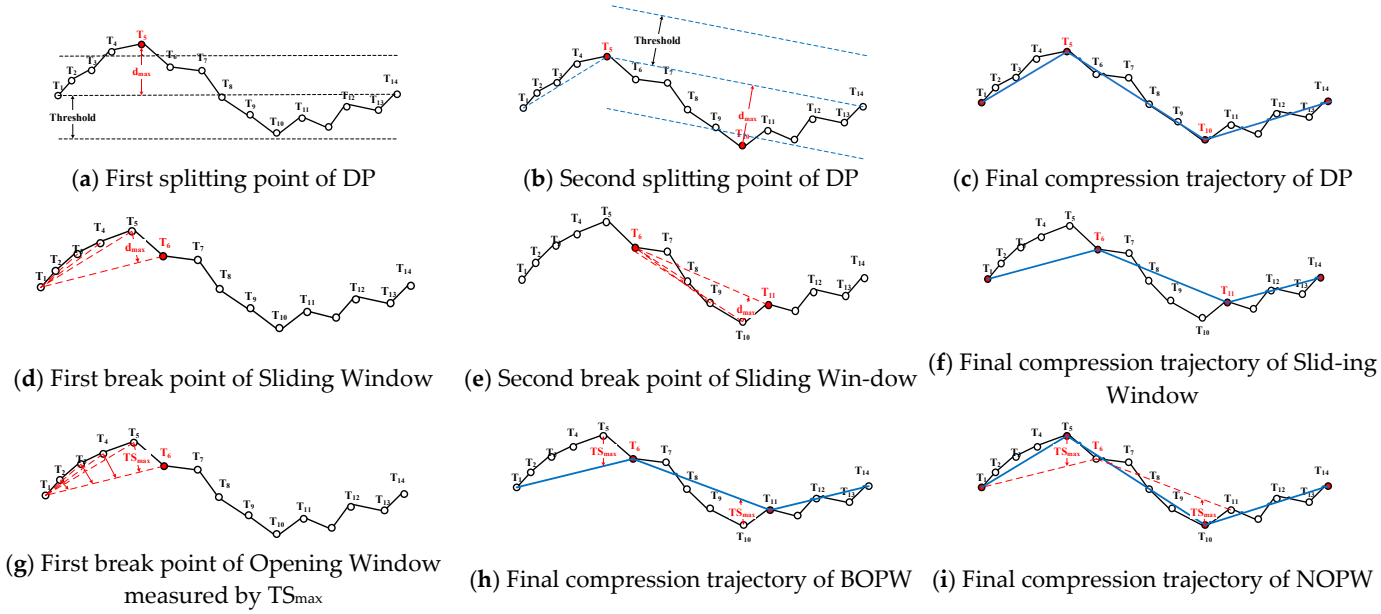
Large-scale trajectory data occupies significant storage capacity, and the presence of substantial redundant information also considerably undermines the efficiency of trajectory mining. The primary goal of compressing trajectory data is to identify effective and precise approaches that reduce both storage and computational demands while maintaining analytical integrity. Commonly employed trajectory compression techniques can be broadly

grouped into three types: compression through line generalization, compression under road network constraints, and compression based on semantics [14,15,45].

3.2.1. Line-Simplification-Based Trajectory Compression

Line-simplification-based trajectory compression is the most commonly used compression method. The advantages of this type of algorithm are as follows: (1) the minimal data constraints, (2) the simplicity and easy implementation, and (3) the lightweight deployment. Therefore, it is suitable for trajectory compression scenarios with limited resources. It treats the trajectory as a sequence of line segments composed of key feature points and uses line simplification methods to compress the trajectory. Line-simplification-based trajectory compression can be divided into offline and online compression:

1. Offline compression is a static compression based on global trajectory characteristics, which compresses the entire trajectory data after it has been read. A classic example of this approach is the Douglas–Peucker algorithm [46]. The algorithm uses the segmented and simplified lines to replace the original trajectory (as shown in Figure 3a–c). This splitting process repeats recursively for each new segment until all deviations fall below the threshold or only endpoints remain. The final simplified trajectory consists of key points such as T1, T5, T10, and T14. The Douglas–Peucker algorithm greatly recognizes the data information of special nodes and has a better performance in both compression rate and accuracy. However, the disadvantage of this algorithm is the high time complexity.



(**a**) First splitting point of DP (**b**) Second splitting point of DP (**c**) Final compression trajectory of DP

(**d**) First break point of Sliding Window (**e**) Second break point of Sliding Win-dow (**f**) Final compression trajectory of Slid-ing Window

(**g**) First break point of Opening Window measured by TS_{max} (**h**) Final compression trajectory of BOPW (**i**) Final compression trajectory of NOPW

Figure 3. Trajectory compression algorithm based on line segment simplification. (**a**–**c**) Douglas–Peucker (DP) Algorithm (Offline): (**a**) First Split: The algorithm begins by connecting the first (T1) and last (T14) points of the original trajectory. It then finds the intermediate point (T5) with the maximum perpendicular distance to the segment T1–T14. (**b**) Recursive Splitting: If the distance exceeds a predefined threshold, T5 is retained as a key point. The trajectory is split into two segments (T1–T5 and T5–T14), and the process repeats recursively for each new segment. (**c**) Final Result: The recursion continues until all points deviate less than the threshold. The final compressed trajectory consists of the key points {T1, T5, T10, and T14}, which preserve the original shape. (**d**–**f**) Sliding Window Algorithm (Online): (**d**) Window Expansion: A sliding window starts from T1 and expands one point at a time (T1–T2, T1–T3, . . .). For each window, the perpendicular distance of all intermediate points to the segment connecting the start and end points is calculated. (**e**) Triggering Compression: When a point (T6) causes the error to exceed the threshold, the sub-trajectory from the start (T1) to the previous

point (T5) is approximated by the segment T1–T6. Point T6 becomes the new start point. (**f**) Final Result: The process repeats, yielding the compressed trajectory {T1, T6, T11, and T14}. (**g**–**i**) Opening Window Algorithm (Online): (**g**) Time-Synchronous Distance: Similar to Sliding Window, but uses Time-Synchronous (TS) distance (considering both spatial and temporal gaps) instead of perpendicular distance. The first point (T6) that exceeds the TS threshold is found. (**h**) BOPW Result: The Bottom-Up Opening Window (BOPW) variant compresses the trajectory to {T1, T6, T11, and T14}. (**i**) NOPW Result: The Normal Opening Window (NOPW) variant, more sensitive to turning points, produces a more accurate compression {T1, T5, T10, and T14}. The blue line is the final compressed path, the red dashed line is an auxiliary line used to calculate the maximum distance.

Hershberger improved the Douglas–Peucker algorithm by introducing the concept of the path hull data structure [47]. Two arrays are used to store the two convex hulls of the subchain: the left stores CH (Vi. . .Vm), and the right stores CH (Vm. . .. Vj). Each time, it is only necessary to search for the trajectory points within the convex hull. By searching for the trajectory points within the convex hull each time, Meratniaand and de By denote a top-down time-ratio algorithm (TD-TR) [48]; this algorithm considers another important extra dimension of trajectories, time. And a more accurate time-ratio distance-measuring technique is used to replace the previous perpendicular distance. The author also applied the time-ratio distance-measuring technique to online compression algorithms in this paper and proposed Opening Window, which will be described in next part. Cheng Long et al. proposed direction-preserving trajectory simplification (DPTS) [49]. DPTS constructs a graph from the original trajectory and identifies a shortest path to produce a compressed trajectory with minimal directional error. Lately, Bashir et al. proposed two compression methods based on urban road networks [50]. They created a compressed lookup lexicon using intelligent mining paradigm to store information about dynamically selected points of interest (POI). This lexicon efficiently computes key patterns using the relative geographic positions of spatial vertices relative to the origin in Euclidean space within a POI. Their method attains linear-time compression by retaining only trajectory data near POIs, rendering it highly suitable for real-time and mission-critical applications.

2. Online compression is a dynamic compression based on local trajectory features, which compresses trajectory data in real time. The most representative algorithm for online compression is Sliding Window [51] and Opening Window [52]. The main idea of the Sliding Window algorithm is to compress trajectories through increasing the size of the window until the error for the potential segment is greater than the threshold. A sliding window with a unit of 1 is defined from the start point of the trajectory, and the size of the window is gradually increased until there exists a trajectory point within the window whose distance to the approximated line segment is greater than the error threshold. Then, the trajectory within the previous window is compressed, as shown in Figure 3d–f. Due to the Sliding Window algorithm's inability to look ahead and lack of a global view like offline algorithms, the compression accuracy is somewhat lacking.

The Opening Window algorithm is a modification of the Sliding Window algorithm [48]. It replaces the vertical Euclidean distance with time-synchronous distance (TS), which takes into account the temporal dimension. This concept is depicted in Figure 3g. In order to improve the accuracy of trajectory compression, Meratnia proposed two algorithms, BOPW and NOPW, as shown in Figure 3h,i. The experimental results have demonstrated that the NOPW algorithm outperforms the BOPW algorithm in terms of trajectory compression accuracy when dealing with trajectories that have unique turning points. As a result, the NOPW algorithm not only achieves real-time compression but also guarantees superior accuracy.

Potamias et al. proposed the STTrace algorithm using the velocity and direction of the two closest sampled points in the approximate trajectory as the reference, and a joint safe area is constructed with a threshold [52]. Muckell et al. proposed the SQUISH [53] and SQUISH-E [16] algorithms. These algorithms initialize a priority queue with a size of k. Then, they add trajectory points to the queue, and, once it reaches its maximum capacity, the point with the smallest error will be removed, and the priority of each remaining point will be recalculated. SQUISH-E, as an enhancement of SQUISH, aims to improve the data compression rates. Liu et al. introduced the Bounded Quadrant System (BQS) algorithm [54]. This algorithm uses the convex hull optimization technique and bounded quadrant method to compress trajectory. Li et al. proposed an error-bounded line simplification algorithm, called ROPW, for trajectory compression [55]. This algorithm is based on the ideas of the Sliding Window algorithm and the Opening Window algorithm, using the Perpendicular Euclidean Distance as the error measurement criterion. They also proposed a reverse Opening Window algorithm based on the backward traversal of the trajectory, and added strategies for jump and acceleration, greatly improving the efficiency of the algorithm. Recently, several scholars have conducted performance evaluations on various line-simplification-based compression algorithms [56,57].

3.2.2. Road Network Constrained Trajectory Compression

Road network constrained trajectory compression is to map the trajectory to the road path based on the road network structure and then compress those map matching points. These methods can significantly reduce the issues caused by sampling errors and generate trajectory representations that are more in line with natural semantics. The nonmaterialized algorithm is one of the earliest compression algorithms based on the road network structure [58]. It converts the mapped trajectory data to a sequence of vertices, then constructs a Snapping Configuration Graph (SCG) to find a road-snapped trajectory under tolerance ε. Note that this compression algorithm provides the temporal information separately from the spatial information. Lerin et al. improved the nonmaterialized algorithm by adopting the shortest path algorithm (SPA) and following path algorithm (FPA) to encode a trajectory path formed by a sequence of links [59]. The shortest path is calculated by the Dijkstra algorithm. If the trajectory is consistent with the shortest path, only the first link and the last link are retained. The main objective is to minimize the number of links in the trajectory encode and reduce the storage requirements.

Kelleris et al. proposed the Map-Matched Trajectory Compression (MMTC) algorithm [60,61]. This method replaces sub-paths by the shortest path with the minimum information loss, so as to make the approximate trajectory and the original trajectory have a high similarity. A cost-optimization solution is calculated by the Minimal Description Length (MDL) model to optimize the objective function composed of the compression ratio and similarity. MMTC is a lossy compression method that includes two types of algorithms: offline algorithm and online algorithm. The offline algorithm is more time-consuming, while the online algorithm has a better running efficiency, but the compression ratio is not as good as the offline algorithm.

Song et al. proposed the PRESS (Paralleled Road-Network-Based Trajectory Compression) compression system [62], which decomposes the original trajectory into a spatial sequence and a temporal sequence for compression. For spatial compression, a lossless algorithm called Hybrid Spatial Compression (HSC) was proposed. It first uses the shortest path to achieve the first-level compression and then decomposes the trajectory into FST sequences. The Huffman code is used to encode the FST, where, the more frequent the FST is, the shorter the corresponding code is, achieving the second-level compression. Note that the PRESS framework has no limit on road matching distance, so there is no upper bound

on the matching distance between the final trajectory points and road segments. In addition, PRESS is a spatial lossless compression algorithm, which means its compression ratio is limited and it does not support online compression. The COMPRESS (Comprehensive Paralleled Road Network Based Trajectory Compression) system is a further development of the PRESS system, which proposes different strengths of lossless spatial compression algorithms to meet different application requirements [63]. Koide et al. proposed a trajectory compression and pattern-matching algorithm called CiNCT, which is more suitable for sparse path data, based on COMPRESS [64]. Zhao proposed Compressing Spatial-temporal Trajectories by Pattern Mining (CLEAN) [65]. CLEAN integrates spatial and temporal compression at the same time. CLEAN designs a frequent spatial compression and encodes the long trajectories to shorter paths to reduce space costs. Finally, on top of the space compression algorithm, a time-domain compression algorithm with a bounded error was designed, which reduces both the space and time costs.

With the application of trajectory compression in the field of mobility, more and more research has focused on distributed online compression based on map matching [66,67]. Among them, a representative work is proposed by Chen et al., who introduced an online trajectory compression framework called TrajCompressor in a mobile environment. As a result, it improved the performance of trajectory compression [68]. To enhance the computational performance of map matching in TrajCompressor, Chen et al. leverage the theory of edge computing and offload the computation tasks of map matching and trajectory compression to nearby drivers' smartphones, in order to meet the requirements of computational capability, lightweight design, and low latency for online trajectory compression [69].

3.2.3. Semantic Trajectory Compression

Semantic Trajectory Compression is a compression method that reconstructs trajectories into a brief semantic text using meaningful states and events, such as street intersections, points of interest, and public transport links. It extends the concepts of road-network-constrained compression in moving object databases and technologies used in wayfinding assistance. The compressed semantic trajectories are easier to read, but the compressed network query is not supported because of the losing of the original longitude and latitude. Schmid and Richter proposed a Semantic Trajectory Compression algorithm (STC) with the major advantage of being human-readable and having a significantly improved compression ratio [70,71]. However, embedding trajectories in the semantic geographic context will increase the time of compression and decompression. In addition, STC does not support the construction of a spatial index and subsequent query which is important to trajectory management. And the LBS application supported by the STC is also limited because it lacks information on the original longitude and latitude. Although STC achieves a high trajectory compression rate, the algorithm lacks the detection of trajectory motion characteristics. Therefore, when there are situations like U-turns or back-and-forth movements in the trajectory, it cannot identify turning behaviors in the compressed trajectory. To address this issue, Feng et al. proposed an enhanced semantic trajectory compression based on road semantics and motion characteristics (EHSTC) [72].

Su et al. proposed the STMaker system using a partition-and-summarization approach to summarize the individual trajectory [73]. STMaker uses routing features and moving features to describe the road and motion behavior of a moving object, which involves the grade of road, road width, direction, speed limit, number of stay points, and number of U-turns. In the period of the trajectory partition, the trajectory is divided into several sub-tracks according to the previously extracted features. Each sub-trajectory has similar features inside, but the characteristics between sub-tracks are quite different. In each

segment, the most representative features are selected to describe the driving process of this segment. In the summarization stage, the characteristics of each segment and each segment are combined to describe the driving process of the trajectory. Since the output of STMaker's framework is a summary of the original trajectory rather than a transformation (such as semantic trajectory), the data volume is significantly reduced, which makes it easier to store and communicate. Despite the smaller data volume, the information conveyed by STMaker is strategically concentrated in the most "interesting" parts of the trajectory, which is more meaningful to humans. Subsequently, Su conducted a further comparative analysis on the collective behavior of historical trajectories on the same route based on STMaker, selected the most interesting features, and generated brief textual descriptions based on these features [74].

Semantic-based trajectory compression improves the readability of trajectories. However, when the compression ratio is high, the compression process may result in the loss of important semantic features from the original trajectory. To address this issue, Liu et al. proposed an enhanced semantic trajectory compression method called Stop-enhanced Trajectory Semantic Compression (STSS). This method divides the trajectory into motion segments and stop segments by extracting stop features. It establishes the relationship between the stop segments and motion segments thresholds through function fitting. By doing so, it reduces information loss during the compression process at high compression rates, thus enhancing the usability of the trajectory [75]. In addition, there are also some novel approaches to achieve trajectory compression, for example, trajectory compression based on vehicle motion pattern recognition [76], trajectory compression based on finite element method [77], and compression methods based on historical trajectory references [78,79]. The overview of trajectory compression methods is shown in Table 2.

Table 2. Overview of trajectory compression methods.

Classification		Algorithm	Time Complexity	Error Metric	Article
Compression based on line segment simplification	Offline Compression	Douglas–Peucker	$O\left(N^2\right)$	Perpendicular Euclidean distance	(Douglas and Peucker, 1973) [46]
		Path Hull	O(NlogN)	Perpendicular Euclidean distance	(Hershberger and Snoeyink, 1992) [47]
		TD-TR	$O\left(N^2\right)$	Time-ratio distance	(Meratnia and de By, 2004) [48]
		DPTS	O(N)	Euclidean distance, direction, speed	(Long et al., 2013) [49]
		CTEV	O(N)	Euclidean distance	(Bashir et al., 2022) [50]
	Online Compression	Sliding Window	$O\left(N^2\right)$	Perpendicular Euclidean distance	(Keogh et al., 2001) [51]
		Opening Window	$O\left(N^2\right)$	Time-synchronous distance	(Meratnia and de By, 2004) [48]
		STTrace	$O\left(N^2\right)$	Time-synchronous distance, direction, speed	(Potamias et al., 2006) [52]
		SQUISH	$O(Nlog(\beta))$	Time-synchronous distance, direction, speed	(Muckell et al., 2011) [53]
		SQUISH-E	$O(Nlog(\frac{N}{\lambda}))$	Time-synchronous distance, direction, speed	(Muckell et al., 2014) [16]
		BQS	O(N)	Euclidean distance	(Liu et al., 2015) [54]
		ROPW	O(N)	Perpendicular Euclidean distance	(Li et al., 2021) [55]

Table 2. *Cont.*

Classification	Algorithm	Time Complexity	Error Metric	Article
Road-network-constrained compression	Nonmaterialized	$O(NM^2)$	Perpendicular Euclidean distance	(Cao and Wolfson, 2005) [58]
	Shortest Path	$O(N^2)$	Euclidean distance	(Lerin et al., 2012) [59]
	MMTC	$O(N^2 \log N)$	A weighted average of network distance and time distance	(Kellaris et al., 2009, 2013) [60,61]
	PRESS	$O(N)$	Time synchronized network distance, network synchronized time difference	(Song et al., 2014) [62]
	COMPRESS	$O(N)$	Time synchronized network distance, network synchronized time difference	(Han et al., 2017) [63]
	CiNCT	$O(B)$	Bit-wise rank value	(Koide et al., 2018) [64]
	CLEAN	$O(N^2 + N\log(N))$	Time synchronized network distance, network synchronized time difference	(Zhao et al., 2020) [65]
	TrajCompressor	$O(M)$	Perpendicular Euclidean distance	(Chen et al., 2019) [68]
	VTracer	$O(M)$	Perpendicular Euclidean distance	(Chen et al., 2019) [69]
Semantic Compression	STC	$O(N^2)$	Average spatio-temporal distance	(Richter et al., 2012; Schmid et al., 2009) [70,71]
	EHSTC	$O(N^2)$	Perpendicular Euclidean distance	(Feng et al., 2013) [72]
	STMaker	$O(N^2)$	N/A	(Su et al., 2014) [73]
	STSS	$O(N^2)$	Homomorphic distance	(Liu et al., 2021) [75]
	SATC	$O(N^2)$	Synchronous Euclidean distance	(Ta et al., 2016) [80]
	ROCE	$O(N)$	Point-to-segment Euclidean distance	(Yin et al., 2022) [81]

N represents the number of trajectory points, β represents the buffer size, λ represents target compression ratio, M represents the number of line segments or edges in a trajectory, B represents a bit vector that controls the size of the internal blocks.

Line simplification algorithms (e.g., Douglas–Peucker, and Opening Window) generally offer a lower computational complexity and are well-suited for resource-constrained environments or applications requiring rapid online processing, such as the real-time tracking of vehicle fleets in urban transport. However, their primary limitation lies in ignoring underlying network constraints, making them less accurate for map-based applications.

Road-network-constrained methods (e.g., MMTC and PRESS) inherently produce semantically meaningful paths aligned with the road network, making them ideal for navigation systems and transportation analytics. The trade-off is their higher computational cost due to map-matching and path search operations, and their dependency on the availability and accuracy of digital road networks.

Semantic compression techniques (e.g., STC and STMaker) achieve the highest compression ratios and generate human-readable outputs, which are valuable for trajectory summarization and mobility pattern analysis. Their drawbacks include the loss of original

coordinate information, which hinders certain spatial queries, and increased computational complexity during the semantic annotation phase.

In practical applications, the choice of algorithm is heavily influenced by the target domain. Urban transport systems often prioritize online compression with road-network awareness (e.g., TrajCompressor). In contrast, UAV or robotics trajectories, which may not follow a predefined network, can benefit more from direction-preserving or speed-aware line simplification (e.g., DPTS and STTrace). For large-scale human mobility analysis, semantic compression offers an excellent balance between storage reduction and interpretability.

3.3. Trajectory Segmentation

One of the key steps for trajectory preprocessing is segmentation, where a raw trajectory is divided into several meaningful segments or phases. If trajectory segmentation is not performed, the trajectory recorded in the time sequence will become an infinitely extended trajectory line, which is not conducive to subsequent analysis and mining. There are three types of existing trajectory segmentation methods: supervised trajectory segmentation, unsupervised trajectory segmentation, and semi-supervised trajectory segmentation.

3.3.1. Supervised Trajectory Segmentation

In supervised trajectory segmentation, the goal is to segment a trajectory based on prior knowledge or labeled data which relies on human experience or subjective criteria, such as time and speed threshold, acceleration, direction, distance, similarity, stop point, and point of interest. In addition, it uses labels available in the training data as input. The labeled data consists of pre-segmented trajectories, where each segment is labeled with a specific activity or behavior. Threshold setting is an important metric in supervised trajectory segmentation, such as time threshold, speed threshold, distance threshold, error threshold, etc. Stay Point Detection (SPD) algorithm is a simple and easy-to-use algorithm for identifying stop points in mobile trajectories [82]. This algorithm identifies stop points by setting a time threshold and a distance threshold, and the trajectory between two stop points is a segment. The implementation of this algorithm is based on the following two assumptions: (1) when people stay in one place for a period of time, their movement speed will decrease; and (2) the movement trajectory between adjacent stop points should be relatively continuous—that is, the distance and time between them should be close enough. WS-II defines an error threshold [83]. The sequence error value is generated by calculating the deviation between the generated points and the actual points in a sliding window, and the extracted sequence error values are used to create a training dataset. Then, a binary classifier is used to classify each error signal sample into segment points and non-segment points. Finally, the trajectory segmentation points are determined by a voting mechanism.

Supervised segmentation may achieve high accuracy but requires labeled data, which can be time-consuming and expensive to obtain. Supervised segmentation methods with targeted criteria settings can achieve the desired effects in specific problems. In addition, in supervised segmentation methods, the criteria threshold is pre-set, and different thresholds can result in significant differences in the accuracy and efficiency of segmentation, which can also lead to poor transferability and compatibility of segmentation methods. If it is set too large, it may lead to missed judgments, while, if it is set too small, it may lead to misjudgments. Therefore, it is necessary to adjust and optimize the threshold according to the specific dataset and application scenario.

3.3.2. Unsupervised Trajectory Segmentation

In unsupervised trajectory segmentation, the goal is to segment a trajectory based on the characteristics of the raw data, without any prior knowledge or labeling. Unsu-

pervised segmentation algorithms mainly include clustering-based segmentation, cost function segmentation, interpolation segmentation, and semantic segmentation. Clustering algorithms are commonly used for unsupervised trajectory segmentation. The algorithm cluster points in the trajectory that are similar to each other in terms of some criterion or attribute. Cost-function-based trajectory segmentation calculates the distance or similarity between trajectories by defining a cost function and divides the trajectories into different segments based on the minimum value of the cost function. Interpolation-based trajectory segmentation is more suitable for denser trajectory data and divides trajectories into different segments using methods such as linear interpolation and random-walk interpolation. In comparison, unsupervised segmentation does not require prior knowledge or labeled data. It can segment the trajectory according to the designed loss function and other methods, which is more universal than the supervised segmentation method:

1. Clustering-based segmentation. In earlier research, unsupervised trajectory segmentation algorithms were mostly based on clustering, such as clustering based on distance, speed, acceleration, density, direction, etc. Among them, TRACLUS is one of the representative algorithms for unsupervised trajectory segmentation methods, which is a distance-based clustering algorithm [84]. It proposed three distance properties: perpendicular distance, parallel distance, and angle distance. Approximate trajectory partitioning is achieved by using the Minimum Description Length (MDL) to partition trajectories. SMoT is a time-based clustering algorithm, which extracts stop points and move points from a trajectory based on the stop time in the trajectory, and then divides a trajectory into multiple stop and move segments [85]. CB-SMoT is a speed-based clustering algorithm [86], and its most significant difference from the SMoT algorithm is that CB-SMoT adds speed as a segmentation criterion on the basis of SMoT, which not only considers the spatial and temporal relationships between trajectory points, but also takes into account the influence of speed. Therefore, the CB-SMoT algorithm can identify stop points and move points more accurately. DB-SMoT is a direction-based clustering algorithm [87]. It clusters trajectories by computing the magnitude of the direction changes. SMoT, CB-SMoT, and DB-SMoT do not require any labels or prior knowledge to perform clustering operations. Time, speed, and direction are only the metrics they use to calculate clustering, not labels for clustering. Leiva et al. proposed a well-known Warped K-Means model to achieve unsupervised trajectory segmentation through sequential clustering algorithm [88]. By minimizing the criterion function with the Sum of Quadratic Error (SQE) which incorporates rigid temporal constraints in the trajectory segmentation step, WKM achieves a more efficient and accurate trajectory segmentation. As is known, K-means algorithm has the advantages of low complexity and high robustness that can quickly converge to local minima. However, it requires the number of priori clusters k to be an input parameter, which limits its application and development in multiple scenarios. Buchin et al. presented an algorithmic framework that can efficiently segment trajectories according to feature combination analysis [89,90]. This framework uses a greedy strategy for segmentation to obtain an optimal solution for monotone criteria, which is similar to the principle of clustering algorithms. It ensures that the trajectory is divided into as few segments as possible and maximizes the length of each individual piece. In addition, they proposed two routines, TEST and FURTHEST, to segment three-dimensional trajectories, respectively. In univariate attribute criteria, it segments trajectories based on the criteria of location, velocity, and heading. In combinations of attribute criteria, they present two different ways of combining criteria, namely, Boolean combinations and linear combinations. In addition, three more complex criteria have been added on the basis of the univariate attribute criteria, which are

curvature, sinuosity, and curviness. The proposed method can improve the robustness of the trajectory segmentation algorithm.

2. Cost-function-based segmentation. The cost function is commonly used in unsupervised trajectory segmentation, mainly to calculate the distance, correlation, and similarity of trajectories. Yan et al. proposed an unsupervised trajectory construction platform, namely, SeTraStream [91]. This method uses a sliding window strategy to achieve online trajectory segmentation. Firstly, the feature vectors of the new incoming batch of raw trajectories are extracted to form the corresponding matrix, and the new batch of trajectories is buffered into a segmentation queue, waiting for segmentation processing. Secondly, the earliest trajectory that completes segmentation is dequeued and a candidate division point is placed at the end of the original batch as shown in Figure 4. Then, the new batch of trajectory Wr is compared with the previous batch of trajectory matrix Wl to calculate the correlation between these two feature vectors with an RV-coefficient function which is a generalization of the correlation coefficient for matrix data. This process is called short-term changes seeking. If the short-term changes do not trigger the threshold of the segmentation algorithm, the comparison window is doubled and long-term changes are sought to find a division point. If no long-term changes are detected, segmentation stops until the next new batch of trajectories is buffered into the queue and the trajectory segmentation process is restarted.

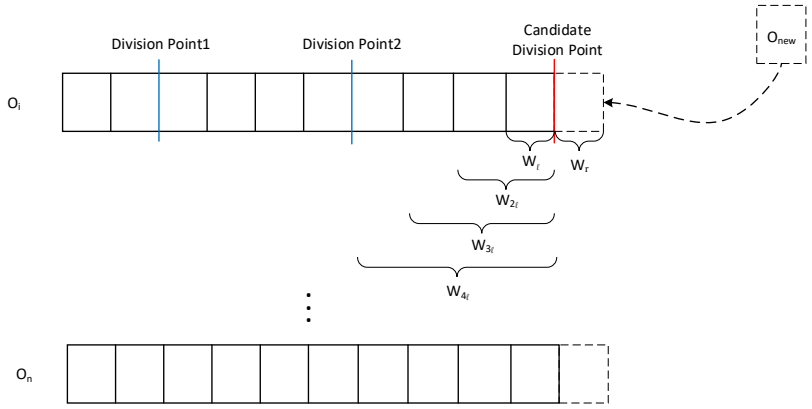


Figure 4. SeTraStream segmentation method.

Júnior et al. proposed a Greedy Randomized Adaptive Search Procedure for Unsupervised Trajectory Segmentation (GRASP-UTS) [92]. The core idea is to achieve a higher homogeneity within segments by applying the Minimum Description Length (MDL) principle. Note that the biggest advantage of GRASP-UTS is that it no longer limits the features of trajectory segmentation, so it can overlay any features on the original trajectory to meet the trajectory segmentation needs of different applications. Xu and Dong introduced an algorithm named TS-TF, which leverages multiple motion attributes for effective trajectory simplification [93]. The process begins with trajectory segmentation, where the Pearson correlation coefficient is applied to assess similarity in movement features—including longitude, latitude, speed, acceleration, and heading—between consecutive points. This helps detect sensitive variations and identify potential segmentation locations. Subsequently, a merging phase is conducted to refine the segmentation result by optimizing the minimum description length (MDL) criterion, thereby preventing over-segmentation and enhancing the overall coherence of the compressed trajectory.

3. Interpolation-based segmentation. Trajectory interpolation is one of the commonly used methods for trajectory segmentation. Etemad defined an octal window (Sow),

which is a sub-trajectory with seven trajectory points, and proposed a trajectory segmentation algorithm based on interpolation called Octal Window Segmentation (OWS) [94]. In octal window, new estimated trajectory points can be created using interpolation techniques, and an error signal can be created by comparing the actual position and estimated position of the moving object. By evaluating the error signal, we can determine whether there has been a heterogeneous change in the trajectory of the moving object. The advantage of this algorithm is that it can adapt to different application scenarios by selecting different interpolation algorithms, which can improve the flexibility of trajectory segmentation. After OWS, Etemad extended Sow to a sliding window, which modified the window size to a configurable parameter with a minimum value of 7, and named the algorithm Sliding Window Segmentation Algorithm (SWS) [95]. The author analyzed the segmentation performance of the SWS algorithm under different window sizes. The generation of error signals and trajectory segmentation are realized through different interpolation techniques, such as linear interpolation, cubic interpolation, random walk interpolation, and kinematic interpolation. At the same time, the robustness of different interpolation techniques is also analyzed. Highlighted that SWS can endure more noise to some extend by having the forward and backward extrapolation mechanisms. Therefore, the algorithm has a high robustness to noise.

4. Semantic-based segmentation. Apart from speed, acceleration, direction, and trajectory similarity, the metric for trajectory segmentation also includes transportation semantic and mode. Inspired by the deep semantic image segmentation in computer vision applications, deep semantic segmentation has been applied to trajectory segmentation. By extracting the motion feature sequence information of GPS raw trajectories, such as geographic location, traffic patterns, user behavior, trajectory relationships, and other feature vectors, the trajectory is segmented and assigned corresponding semantics for each segment. One of the representative methods is Convolutional Neural Network (CNN) schemes for trajectory segmentation, firstly, with an extraction of continuous, overlapping, and equidistant trajectory segments for segmentation, and extracting features such as position, velocity, and acceleration for each segment. Then, transportation modes are recognized through a CNN framework. This method has high segmentation efficiency, but, in cases where traffic patterns are more complex, the uniformly fixed segmentation method will lead to a decrease in accuracy. To address this issue, a Bayesian Temporal Convolutional Network (BTCN) is proposed for unsupervised, uncertainty-aware GPS trajectory segmentation [96]. The BTCN algorithm can capture the uncertainty of different traffic patterns in the trajectory segmentation process. Specifically, this method uses a convolutional neural network to map GPS trajectory points into a one-dimensional space and uses a variational autoencoder to capture the latent distribution between trajectory points, improving the representation ability of trajectory points. Then, Bayesian inference is used to estimate the posterior distribution of trajectory points, and Monte Carlo dropout sampling is used for variational inference. This method can more accurately capture the uncertainty in the trajectory segmentation process, improving the accuracy of segmentation. However, this method requires a large amount of computing resources and time and is not suitable for online trajectory segmentation applications.

3.3.3. Semi-Supervised Trajectory Segmentation

In semi-supervised trajectory segmentation, it combined both supervised and unsupervised methods. Some trajectories are labeled by an expert, while others are segmented using an unsupervised algorithm. Both supervised and unsupervised methods

have their own advantages and disadvantages. Supervised methods heavily rely on domain experts' professional knowledge, making it difficult to obtain high-quality labeled datasets. Unsupervised algorithms can achieve automatic trajectory segmentation, but have a lack of semantic relevance. Therefore, some scholars have proposed a semi-supervised trajectory segmentation method that combines the advantages of both supervised and unsupervised methods. Semi-supervised segmentation can achieve a higher accuracy than unsupervised segmentation while requiring less labeled data than supervised segmentation.

Junior et al. introduced the RGRASP-SemTS algorithm [97], which uses a reactive greedy randomized search strategy for semi-supervised semantic trajectory segmentation. The method employs a limited amount of labeled data to guide the segmentation of unlabeled trajectories. It identifies optimal segmentation points by adjusting segment boundaries and uses unsupervised criteria such as the minimum description length (MDL) to assess internal segment consistency. Simultaneously, supervised cues including semantic landmarks help measure the similarity across labeled segments, leading to locally optimal segmentation. Dabiri et al. developed a semi-supervised convolutional autoencoder (SECA) for trajectory segmentation [98,99]. The process first splits the trajectory uniformly, then merges segments sharing the same traffic pattern via online detection. Subsequent change point detection differentiates patterns and triggers re-segmentation. The outcome is a set of segments, each representing a distinct traffic pattern, which serves as input for traffic pattern recognition.

In conclusion, the choice of trajectory segmentation method depends on the specific application and available resources. Supervised segmentation may be useful when labeled data is available and when the goal is to achieve high accuracy. Unsupervised segmentation may be more appropriate when there is no prior knowledge or labeling available and when the goal is to discover unknown patterns or structures in the data. Semi-supervised segmentation may be useful when there is a limited amount of labeled data available, and when the goal is to balance accuracy and efficiency. The overview of map segmentation methods is shown in Table 3.

Supervised methods (e.g., SPD, WS-II) can achieve a high accuracy for specific, well-defined tasks like stop-point detection. Their major limitation is the reliance on labeled data, which is costly to acquire. They are best suited for applications where clear rules or thresholds exist, such as segmenting commuter trips based on dwell times.

Unsupervised methods (e.g., TRACLUS, GRASP-UTS, and Warped K-Means) offer greater flexibility for discovering unknown patterns without pre-labeled data, making them applicable to exploratory data analysis in new domains like UAV flight path analysis or animal movement studies. Their challenge lies in parameter tuning and the potential lack of semantic meaning in the resulting segments.

Semi-supervised approaches (e.g., RGRASP-SemTS, and SECA) strike a balance, leveraging small amounts of labeled data to guide the segmentation of large unlabeled datasets. This makes them highly practical for real-world scenarios where some expert knowledge is available but full manual labeling is infeasible, such as in large-scale transportation mode identification.

The segmentation choice depends on data availability and the analysis goal. Robotics and autonomous systems, which often operate in structured environments with clear state changes, may effectively use cost-function or interpolation-based segmentation. For urban computing, semi-supervised methods are increasingly popular due to their ability to incorporate domain knowledge (e.g., known landmarks) while scaling to city-wide datasets.

Table 3. An overview of map segmentation methods.

Classification	Method	Time Complexity	Segment Metric	Article
Supervised trajectory segmentation	SPD	O(N)	Time and distant threshold	(Zheng et al., 2011) [82]
	WS-II	O(N)	Error threshold	(Etemad et al., 2020) [83]
Unsupervised trajectory segmentation	TRACLUS	O(NlogN)	Distance	(Lee et al., 2007) [84]
	SMoT	O(NlogC)	Time	(Alvares et al., 2007) [85]
	CB-SMoT	O(NlogC)	Speed	(Palma et al., 2008) [86]
	DB-SMoT	O(NlogC)	Direction	(Rocha et al., 2010) [87]
	Warped K-Means	O(Nd)	Criterion function	(Leiva and Vidal, 2013) [88]
	Greedy segmentation	O(NlogN)	Location, heading, speed, velocity, curvature, sinuosity, and curviness.	(Buchin et al., 2011; Buchin et al., 2010) [89,90]
	SeTraStream	O(N)	Correlation of features	(Yan et al., 2011) [91]
	GRASP-UTS	O(mN)	Homogeneity of features	(Soares Júnior et al., 2015) [92]
	TS-MF	O(N)	Similarity of multiple motion features	(Xu and Dong, 2022) [93]
	OWS	O(N)	Error signal	(Etemad et al., 2019) [94]
	SWS	O(N)	Error signal	(Etemad et al., 2021) [95]
	BTCN	O(mN)	Transportation mode	(Markos et al., 2021) [96]
Semi-supervised trajectory segmentation	RGRASP-SemTS	O(mN)	Homogeneity of features	(Junior et al., 2018) [97]
	SECA	O(N)	Homogeneity of features	(Dabiri et al., 2019) [98]

N represents the number of trajectory points, C represents candidate stop, d represents the sample vector dimension, and m represents the total number of iterations.

3.4. Map Matching

Due to transmission errors, sampling errors, low-frequency sampling, and device malfunctions of trajectory collection equipment, there is a problem of deviation between the original trajectory data and the true path data, and even the phenomenon of a large distance gap between two consecutive sampling points, which exacerbates the difficulty of path reconstruction and seriously affects the effectiveness of subsequent data mining and analysis. Therefore, in the trajectory data preprocessing stage, map matching technology is necessary for relocating the original trajectory points, correcting the sampling data to the actual road, and obtaining a trajectory consistent with the actual driving path. The accuracy of map matching is closely related to the accuracy of device positioning, sampling rate, and the complexity of the road network topology. Early map matching methods mainly include geometric-based matching methods and topology-based matching methods [17,18]. These methods have simple ideas, which are easy to implement, and focus on high sampling rates and online matching. However, they are easily affected by positioning errors and sampling frequency. Therefore, they are more suitable for scenarios with a high sampling frequency and small positioning errors. Recently, map matching algorithms have focused more on challenging scenarios such as low sampling frequency and high positioning errors, such as WIFI or mobile signal positioning. In order to solve the problems brought by new application scenarios, various probability-based matching methods and advanced matching methods have been proposed [19,100,101]. Table 4 summarizes the classification of these map matching techniques.

Table 4. An overview of map matching techniques.

Classification	Method	Time Complexity	Article
Geometric-based model	PTP, PTC, CTC	$O(N)$	(Bernstein and Kornhauser, 1996) [102]
	PTP, PTC, CTC	$O(N)$	(White et al., 2000) [103]
	RRF	$O(N)$	(Taylor et al., 2001) [104]
	Global Map-Matching	$O\left(MNlog^2MN\right)$ for Fréchet dist, $O(MNlogMN)$ for weak Fréchet dist	(Brakatsoulas et al., 2005) [105]
Topology-based model	MPM	$O(S)$	(Alt et al., 2003) [106]
	MM	$O\left(N^2\right)$	(Quddus et al., 2003) [107]
	GeoTrackMapper	$O(NlogN)$	(Chawathe, 2007) [108]
	ATMM	$O\left(N^3\right)$	(Zhao et al., 2018a) [109]
	HFTMM	$O\left(N^2\right)$	(Yu et al., 2022) [110]
Probability-statistics-based model	MHT-MM	N/A	(Pyo et al., 2001) [111]
	Adaptation MHT-MM	N/A	(Marchal et al., 2005; Schuessler and Axhausen, 2009) [112,113]
	MDP-MM	$O\left(Nk^2MlogM\right)$	(Chen et al., 2014) [114]
	ST-CRF	$O(NSlogS)$	(Liu et al., 2016) [115]
	MCM	$O(MlogM)$	(Li et al., 2023) [116]
	PMHT-MM	N/A	(Wang et al., 2023) [117]
Advanced model	HMM	$O\left(NM^2\right)$	(Newson and Krumm, 2009) [118]
	OHMM	$O\left(NM^2\right)$	(Goh et al., 2012) [119]
	FMM	$O(MNlogN)$	(Yang and Gidofalvi, 2018) [120]
	OM2	$O\left(NM^2\right)$	(Xie et al., 2020) [121]
	INC-RB	$O\left(NM^2logM\right)$	(Luo et al., 2020) [122]
	ST-Matching	$O\left(Nk^2MlogM\right)$	(Lou et al., 2009) [123]
	IVMM	$O\left(Nk^2MlogM\right)$	(Yuan et al., 2010) [124]
	STP-IWC	$O(N)$	(Teng and Wang, 2019) [125]
	AMM	$O\left(Nk^2\right)$	(Hu et al., 2023) [126]
	HRIS	$O\left(NkM^2\right)$	(Zheng et al., 2012) [127]
	HMM + RCM	$O\left(NkM^2\right)$	(Jagadeesh and Srikanthan, 2017) [128]
	DeepMM	N/A	(Zhao et al., 2019) [129]
	TMM	N/A	(Jin et al., 2022) [130]
	L2MM	N/A	(Jiang et al., 2023a) [131]

N represents the number of GPS points in the trajectory, k represents the average number of candidate points per GPS point, M represents the total number of edges or intersections in the road network, and S represents the number of segments.

3.4.1. Geometric-Based Methods

This method is based on geometric principles and analyzes the relationship between point to point, point to segment, and segment to segment. By using features such as distance, angle, and shape similarity, the trajectory points are matched with map data optimally. The geometry-based model algorithm is relatively simple to implement, but

the disadvantage is that it does not consider the correlation between the collected points and the connectivity between roads, resulting in a low matching accuracy and poor model stability. This method is suitable for road sections with a high sampling frequency and dense sampling points, but performs poorly in sparsely sampled road sections.

A representative model in this field is the point-to-point, point-to-curve, and curve-to-curve matching method proposed by Bernstein et al. [102]. Subsequently, White et al. expanded this class of algorithms and conducted in-depth analysis and road tests [103]. This algorithm matches the sampling points to the nearest road network node, finds the arcs connected to these nodes, and calculates the projection distance from the sampling points to the candidate road segments, then selects the nearest road segment as the matching segment. Therefore, this method is highly sensitive to the spatial road network data generation method, and the more data points there are of the map shape, the easier the matching becomes. Taylor et al. proposed a segment-to-segment method using the road reduction filter (RRF) algorithm [104]. For each trajectory sampling point, a candidate point set is generated on the road network, and, then, the road segments generated by these candidate points are compared with the road segment trajectory formed by the sampling signal, selecting the closest segment as the matching result. The most representative model in geometric-based matching methods is the global matching algorithm based on the Fréchet distance proposed by Brakatsoulas et al. [105]. This algorithm first finds a potential matching path that minimizes the sum of Fréchet distances between sample points and the path. Unlike the Euclidean distance or Hausdorff distance, the Fréchet distance is a spatial–temporal curve similarity distance. Therefore, this method is more suitable for comparing the similarity between two curves and has a higher matching accuracy.

3.4.2. Topology-Based Method

Topology-based matching methods use the multiple features of the road network topology to constrain the potential matching set of sample points and calculate the topological relationship between sample points and adjacent road segments. This method places emphasis on the road network structure and connectivity, and incorporates an analysis of similarity between the historical sample data and road topology. As a result, it improves both the matching efficiency and accuracy, particularly in complex geographical environments such as intersections and roundabouts. However, the algorithm requires accurate road network data and historical sample data, making it susceptible to the influence of data sparsity and sampling errors.

In this type of research, Alt et al. proposed a simple topological matching method, which uses information such as the vehicle history data, relationship between trajectories, and road topology features to constrain the candidate matching of sample points [106]. Although this algorithm improves the matching efficiency, it is susceptible to collection noise and data sparsity, and may lead to incorrect matching in complex road conditions. Quddus et al. introduced a weighted topological matching approach that integrates trajectory direction, GPS point proximity to roads, and correlation metrics [107]. The method assigns corresponding weights to these features and identifies the road segment with the highest aggregated weight as the matched segment. This strategy aims to minimize input requirements, streamline computational steps, and support efficient matching.

In order to improve the generality of map-matching algorithms, Chawathe designed an enhanced model that can be applied to various segmented map-matching methods [108]. This model implements three different levels of map matching: Point Match, Local Match, and Segmented Path Match. Based on these algorithms, the author established a map-matching system called the GeoTrackMapper system. This system dynamically compares the features of the next sample point and adjusts the previous path-matching results based

on the comparison results. Test results have shown that segmented map-matching methods have a higher matching accuracy for long paths compared to Point Match and Local Match. Zhao et al. developed an enhanced topological map-matching algorithm grounded in the Dempster–Shafer theory [109]. This framework addresses the inherent uncertainty and unpredictability in map matching by computing belief probabilities for all candidate points associated with each GPS observation.

Topological map-matching algorithms are particularly suitable for high-frequency sampling scenarios. Among these, Yu et al. introduced an enhanced High-Frequency Trajectory Map Matching (HFTMM) algorithm [110]. This approach first partitions the trajectory into segments and then performs map matching on each segment by incorporating road network topological features. The method significantly enhances the accuracy and efficiency of matching.

3.4.3. Probability-Statistics-Based Model

The model based on probabilistic statistics utilizes multiple features of the sampled points, such as the position, direction, velocity, etc., to calculate the matching degree between the trajectory data and candidate points on the map using probabilistic statistical theory. The most representative method is based on confidence intervals and multiple hypothesis techniques. Based on multiple hypothesis techniques, multiple road segments are first selected within the confidence interval and added to the candidate road segment set. Then, a score is calculated for each candidate road segment, and the road segment with the highest score is selected as the final match. The advantage of this matching method is that, even if there are errors in the information of a sampled point, it will not have a significant impact on the subsequent matching. Therefore, in complex road environments, the overall performance of probabilistic algorithms is superior to topological algorithms. In addition, the model can provide a measure of uncertainty for the localization result through the calculation of confidence intervals, which helps users evaluate the reliability of the localization result. A wider confidence interval indicates a higher uncertainty in the matching result. However, probabilistic model methods require complex probability calculations, which have a higher computational complexity. The derivation of model algorithms is also difficult, making them less easy to understand. Therefore, such map-matching algorithms are slower in processing large-scale data and more challenging to implement.

Pyo et al. proposed the multiple hypothesis technique for online map matching using a GPS device and a dead reckoning (DR) device [111]. They calculated the probability of each hypothesis using likelihood functions, generated pseudo-measurements around GPS locations based on surrounding roads, and restructured the multiple hypothesis tracking (MHT) into a single target problem. Hypothesis pruning was then performed on hypotheses with probabilities below a threshold to reduce their number. To mitigate the impact of deviations between GPS/DR sensor outputs and pseudo-measurements on algorithm performance in map matching, a biased Kalman filter was used to estimate the biases, leading to improved matching accuracy. However, Pyo et al.'s work primarily focused on the accuracy of localization rather than algorithm speed, resulting in suboptimal performance when matching large-scale data.

In contrast, Marchal et al. and Schuessler and Axhausen emphasized the computational performance of the algorithm, ensuring that it could rapidly process large amounts of data within reasonable errors and even achieve real-time trajectory visualization on the map [112,113]. The core idea of this algorithm is to establish an initial set of paths through topological search and always maintain a set of candidate paths as GPS feedback data. When encountering intersections, each subsequent link is added as a new link at the end of the routing. A scoring function is established to continuously update the matching scores

of candidate paths with newly linked segments. Therefore, achieving a balance between the matching accuracy and computational speed is the key challenge that such algorithms need to address.

For low-sampling-rate trajectories, the MDP-MM algorithm employs a multiple hypothesis technique (MHT) to track several candidate routes for each GPS point during matching [114]. The optimal route is selected based on the lowest evaluation score, with spatial proximity serving as the main criterion. However, the number of candidate routes can grow exponentially, making MHT computationally expensive for large-scale applications. To mitigate this, MDP-MM retains only the non-dominated route at each candidate location, thereby pruning inferior options and significantly enhancing algorithmic efficiency. Liu et al. have proposed a novel spatial-temporal conditional random field (ST-CRF) method for low-frequency GPS map-matching, which outperforms existing approaches in performance and robustness while solving the label-bias problem [115].

To improve the robustness of map matching, Li et al. developed the Multiple Candidate Matching (MCM) algorithm [116]. This approach identifies the longest common subsequence between the actual trajectory and possible matched routes. In contrast to earlier MHT-based methods, MCM eliminates the need for historical data to score candidate paths or pretrained probability models. Instead, it retains plausible historical matching candidates and uses the structural continuity between roads and trajectories to constrain computation. By pruning infeasible paths, the algorithm achieves a balance between matching accuracy and operational efficiency.

Wang et al. were the first to apply the MHT technique in the field of airborne INS platform positioning [117]. They proposed the PMHT-MM framework to assist INS in eliminating position errors caused by long-term inertial navigation biases and drift. The PMHT-MM algorithm operates in a batch processing mode and seeks to maximize the posterior probability density function $\rho(X \mid Z)$ and the maximum expected iteration by collecting the measurement set Z over multiple sampling periods T.

In addition to confidence intervals and the probability calculations of the multiple hypothesis technique, there are other probability statistical techniques that can be applied in map-matching models. For example, Bayesian inference methods can be used to calculate the probability distribution of multiple candidate points to determine the best matching result [132]. Markov Chain Monte Carlo (MCMC) methods can provide more comprehensive positioning information by estimating the posterior probability distribution of vehicle positions [133]. Based on Conditional Random Field (CRF) methods, it is possible to model the relationship between observation points, candidate points, emission probability, and transition probability. The model parameters can be optimized using methods such as maximum likelihood estimation, and map matching can be performed through inference and decoding. Additionally, particle-filter-based map-matching algorithms are also utilized [134,135]. These methods can further enhance the accuracy and robustness of map-matching models.

3.4.4. Advanced Model

With the development of map-matching algorithms, the consideration of feature factors has gradually diversified. In particular, addressing the issues of low sampling frequency and high fault tolerance in early algorithms has become the focus of optimization. In order to adapt to new challenges, various advanced map-matching models have been proposed, including HMM models, maximum weight models, deep learning models, and local path inference models. Currently, advanced models are the mainstream in map-matching models and a key research direction for the future. Advanced map-matching algorithms take into account more comprehensive trajectory information, including the

speed, direction, angle, road network topology, sampling frequency, and noise in historical data. Therefore, these advanced map-matching algorithms generally have a higher accuracy, but their common drawback is the higher algorithm complexity. We have categorized and summarized representative map algorithms proposed in the past two decades and found that HMM models are the most widely used method in advanced models, followed by maximum-weight-based models and deep-learning-based models:

1. The HMM model performs well in sequential modeling and road network connectivity, which is why it is widely used. In the HMM-based map-matching algorithm, the observation sequence represents the trajectory sample points, and the state sequence represents the potential matching path points. The key of this algorithm is to transform the map-matching problem into the decoding problem of HMM, to find the state sequence with the maximum joint probability, which means selecting the maximum probability value and its corresponding previous state at each time step, and, finally, obtaining the optimal state sequence corresponding to the maximum joint probability value. The most representative research in this field is the algorithm proposed by Newson et al. [118]. They assumed that the absolute difference between the Euclidean distance of adjacent sampling points and the path distance between two matched points follows an exponential distribution, and designed the transition probability based on this assumption. The smaller the difference, the higher the transition probability. This method uses the Viterbi algorithm to calculate the sequence of vehicle travel paths.

The OHMM algorithm is an online map-matching algorithm that uses the variable sliding window method (VSW) to ensure the accuracy of online matching [119]. This algorithm assumes that the measurement error of GPS sampling points follows a normal distribution. On the other hand, Yang et al. assumed that the distance between observation points and candidate matching points follows a Gaussian distribution and proposed the FMM algorithm [120]. This algorithm adds an Upper Bounded Origin Destination Table (UBODT) in the preprocessing stage, which transforms the search for the optimal path into a hash search process for the shortest path. In the matching stage, the path matching is achieved through two steps: candidate search and optimal path inference. Additionally, a penalty mechanism is introduced to reduce the weight of long paths, addressing the issue of backward movements in the HMM algorithm.

OM2 is an offline map-matching algorithm that consists of three steps: preprocessing, map-matching, and post-processing [121]. These steps are responsible for trajectory simplification, offline matching, and accurate intersection mapping, respectively. This algorithm adopts the same assumptions for observation probability, transition probability, and probability distribution as Newson's algorithm. INC-RB is an online map-matching algorithm. Unlike OHMM, this algorithm assumes that the measurement error of GPS sampling points follows a Gaussian distribution [122].

2. The maximum-weight-based map-matching algorithm converts the map-matching problem into a problem of minimizing distance weights. By establishing a scoring system, this system evaluates each candidate path based on multiple features and selects the optimal path with the highest score. Compared to the HMM model, the advantage of this method is its higher flexibility in handling complex paths. However, a drawback of weight-based map-matching algorithms is that they require higher data quality for the collected point data. Therefore, data preprocessing is needed to improve the data quality.

The ST-Matching algorithm has the same observation probability and initial probability as the HMM algorithm, but the calculation of the state transition probability and

weight is different [123]. The calculation of state transition probabilities in the ST-Matching algorithm includes a measure of the difference between the velocity vector and the average velocity vector, which improves the matching accuracy. However, the weight calculation in this algorithm uses a simple summation method, ignoring the mutual influence between candidate points. Once a node is misclassified, it will lead to continuous errors in subsequent nodes.

To address the above issues, Yuan et al. proposed an Interactive Voting-based Map Matching (IVMM) algorithm that considers the relationships between all sampling points to find the optimal path [124]. Firstly, a distance weight matrix is defined for each sampling point to evaluate the mutual influence between all candidate points, where larger distances correspond to smaller weights. Then, an optimal route passing through each candidate point is determined. Finally, the final matching route is selected through voting among these routes.

Teng and Wang proposed a real-time vehicle map-matching algorithm called STP-IWC, which improves positioning accuracy and reduces time lag by integrating spatio-temporal proximity and an improved weighted circle [125]. This algorithm first develops an STP method, which reduces the positioning time by dynamically and adaptively refining the optimal candidate matching road, and then identifies the best matching road to improve positioning accuracy by enhancing angle similarity and introducing new weighted values using the improved IWC method.

In scenarios with uneven sampling errors, the AMM algorithm proposes an adaptive map-matching algorithm [126]. It establishes a synchronous evaluation model between the sampling points and candidate points to automatically adjust the calibration observation data and filter out low-quality sampling points based on different measurement errors, thereby improving the matching accuracy.

3. Deep-learning-based map-matching algorithms have become popular in recent years. The map-matching algorithms based on the HMM model or weight-based methods do not take into account the potential value of historical trajectories, including the historical trajectories of the same vehicle and the trajectories of other vehicles passing through similar road segments. With the rise of deep learning algorithms, researchers have approached the problem from a data-driven perspective. By using deep learning methods, the large-scale trajectory data can be maximally utilized in the map-matching process. This approach reduces the impact of sampling frequency and trajectory noise on the matching results, greatly improving the accuracy of map matching. However, a drawback of these algorithms is that they require a large amount of labeled data to train the model parameters, and obtaining labeled training data is not always easy.

Initially, the introduction of these algorithms simply added information from historical trajectories, such as the HRIS algorithm [127]. The HRIS algorithm is proposed to infer possible routes for low-sampling-rate trajectories by leveraging information from historical trajectories. With the development of deep learning models, more and more improved models have been applied to map matching. Jagadeesh and Srikanthan extended the widely used HMM matching method and supplemented it with a route choice model based on a multinomial logit model [128]. This algorithm uses a multinomial logit path choice model to re-evaluate the partial paths generated by the online map-matching method based on HMM. Zhao et al. proposed a DMM algorithm, assuming that the spatial noise of the sampling points follows a Gaussian distribution [129]. The algorithm uses embedding techniques, an attention-enhanced sequence-to-sequence model, and trajectory data augmentation to improve the accuracy of map matching.

To tackle data sparsity and limited labeled data in training, Jin et al. introduced the TMM model, which performs map matching efficiently with minimal supervision [130]. The

model operates under the assumption that spatial noise in each coordinate follows a zero-mean Gaussian distribution. It synthesizes trajectory data using road network information and available labeled trajectories. These generated trajectories are used to pretrain a deep learning model, which is later fine-tuned with real labeled data. A Transformer architecture is then applied to further enhance matching accuracy. By capturing both internal correlations among GPS points and external relationships between input and output trajectories, the TMM algorithm achieves strong matching performance.

Similarly, Jiang et al. proposed two representation enhancement methods from the perspective of data augmentation to learn high-quality representations of low-quality data [131]. This algorithm has a good robustness to low sampling rates, uneven sampling rates, and noise, and performs well with a small amount of training data.

Geometric and topological methods are computationally efficient and perform well in high-frequency sampling scenarios with open, well-defined road networks, making them suitable for real-time applications in urban transport. However, their accuracy drops significantly with decreasing sampling rates or in complex road environments like intersections and overpasses.

Probability-based models (e.g., HMM and MHT) significantly improve robustness to noise and low-sampling-rate data, establishing themselves as the de facto standard for matching vehicle GPS data in city environments. The primary trade-off is their higher computational complexity, especially for the offline, global versions.

Advanced models, particularly those based on deep learning, show great promise in handling extremely challenging conditions, such as noisy data from mobile phones or complex urban canyons, by learning matching patterns directly from large-scale historical trajectories. Their current limitations include a dependency on large, diverse training datasets and reduced interpretability compared to probabilistic models.

The application context is critical: High-frequency logistics tracking can be effectively handled by fast topological algorithms. In contrast, UAV trajectory matching over rural or unstructured areas may require more robust probabilistic methods that can handle larger positional errors. The emerging deep-learning-based methods are particularly suited for large-scale mobility service platforms that possess vast amounts of historical data and require a high matching accuracy across diverse conditions.

3.5. Interplay and Boundaries Between Preprocessing Tasks

While we have categorized methods into cleaning, compression, segmentation, and map matching for clarity, the boundaries between these tasks are often fluid and synergistic in practice. It is crucial to distinguish between their core, mutually exclusive functions and their overlapping applications.

Core Functions: The primary objective of compression is to reduce data volume, whereas segmentation aims to partition data into meaningful units. Map matching's core role is spatial correction.

Boundary Cases and Overlaps: A key boundary case is Road-Network Constrained Compression (Section 3.2.2), which explicitly integrates map matching as a foundational step. Similarly, Semantic Trajectory Compression (Section 3.2.3) and Semantic Segmentation (Section 3.3.2) both rely on extracting meaningful features (e.g., stops and modes) from the trajectory, blurring the line between compression and segmentation. Understanding these interconnections is vital for designing effective, multi-stage preprocessing pipelines.

4. Public Dataset

Currently, there is a wide range of publicly available traffic trajectory datasets, obtained from diverse sources such as GPS devices, mobile applications, and transportation agencies.

However, the quality and consistency of these datasets can vary due to factors like data collection methodologies, sensor accuracy, and data processing techniques. Furthermore, the recorded fields in these datasets may differ, including information such as latitude, longitude, timestamp, speed, and heading. Additionally, the time of data collection can vary, ranging from real-time streaming data to historical data spanning months or years. In order to present each dataset more clearly, Table 5 compares the geolocation, data objects, classification, content, format, data sources, and production years of representative publicly available datasets in various traffic trajectory domains. It also provides download links for the datasets.

Table 5. Openly available datasets.

Dataset	Download Address	Object	Geography	Classification	Field	Format	Source	Year
GeoLife	https://www.microsoft.com/en-us/download/details.aspx?id=52367 (accessed on 22 November 2025)	Vehicle, Pedestrian	China, USA, Republic of Korea, Japan	Urban activity trajectory data	Position coordinates, time, transportation mode, etc.	plt	Sensors, phones	2007–2012
T-Drive	https://www.microsoft.com/en-us/research/publication/t-drive-trajectory-data-sample/ (accessed on 22 November 2025)	Vehicle	Beijing, China	Taxi trajectory data	Position coordinates, time, etc.	txt	Sensors	2011
NYC-Taxi	https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page (accessed on 22 November 2025)	Vehicle	New York, NY, USA	Taxi trajectory data	Position coordinates, time, etc.	parquet	Sensors	2009–present
Mirror-Traffic	http://www.scenarios.cn (accessed on 22 November 2025)	Vehicle, Pedestrian	China	Vehicle trajectory data on highway ramps, straight roads, intersections, etc.	Mileage, time, lane, etc.	video	Camera, UAV	2019–2020
NGSIM	https://catalog.data.gov/dataset/next-generation-simulation-ngsim-vehicle-trajectories (accessed on 22 November 2025)	Vehicle	USA	Highway vehicle trajectory dataset	Position coordinates, time, etc.	txt	Sensors	2004
HighD	https://www.highd-dataset.com/ (accessed on 22 November 2025)	Vehicle	Cologne, Germany	Highway vehicle trajectory dataset	Position coordinates, velocity, acceleration, lane, etc.	csv	UAV	2017–2018
MAGIC	https://magic.tongji.edu.cn/kycg/MAGICsjj.htm (accessed on 22 November 2025)	Vehicle	Shanghai, China	Urban expressway vehicle trajectory data	Position coordinates, velocity, acceleration, lane, etc.	csv	UAV	2020
BikeShare	https://open.toronto.ca/dataset/bike-share-toronto-ridership-data/ (accessed on 22 November 2025)	Shared bike	Toronto, UK	Shared bicycle ride data	Origin–destination, time, etc.	csv	APP	2017–2020
OpenSky Network	https://opensky-network.org/ (accessed on 22 November 2025)	aircraft	Global	Aircraft trajectory data	Origin–destination, time, etc.	csv	ADS-B	2012–present

A Framework for Comparative Evaluation and Benchmarking

To guide future research in algorithm selection and development, we propose a standardized framework for the comparative evaluation of trajectory preprocessing methods.

1. Datasets: Utilizing diverse public datasets to cover various scenarios (e.g., GeoLife for urban pedestrian and vehicle trajectories, T-Drive for taxi data, HighD for highway driving, and UAV datasets for free-moving objects).
2. Unified Metrics: Adopting a core set of metrics for each task. For example, evaluating compression algorithms should always report the Compression Ratio and Synchronous Euclidean Distance (SED). Map-matching algorithms should be compared on Accuracy and Running Time under different conditions.
3. Testing Dimensions: Crucially, methods must be evaluated across dimensions as follows:
    (1) Varying Sampling Rates: From high-frequency (1 s) to low-frequency (5 min).
    (2) Induced Noise Levels: Adding synthetic Gaussian noise or outliers to test robustness.
    (3) Road-Network Density: Comparing performance in sparse rural networks versus dense urban grids.

5. Discussion

In the era of big data, preprocessing massive trajectory data remains a complex and systematic challenge. As identified throughout this survey, existing methods in cleaning, compression, segmentation, and map matching each exhibit specific limitations that hinder their scalability, accuracy, and applicability in real-world scenarios. Below, we synthesize these challenges and propose future research directions that directly address the gaps identified in earlier sections.

(1) Efficient Computation for Large-Scale Trajectory Processing

As noted in Sections 3.2 and 3.4, many trajectories' compression [20,136] and map-matching algorithms suffer from high computational complexity, making them impractical for real-time or city-scale applications. To overcome these limitations, future work should investigate distributed and parallel computing frameworks such as Spark, Flink, and GPU-accelerated platforms. These technologies can help scale trajectory compression (Section 3.2.1) and global map-matching algorithms (Section 3.4.4) to support real-time streaming applications in intelligent transportation and location-based services.

Therefore, future researchers can explore technologies such as parallel computing, distributed computing, and high-performance computing to improve the computational capacity of trajectory data preprocessing. The computational demands of preprocessing massive-scale trajectory data (e.g., from city-wide vehicle fleets) necessitate parallel and distributed computing. We believe that adopting a distributed processing architecture can effectively improve the scalability of trajectory preprocessing computational performance. Among them, MapReduce, as a model used in the Hadoop distributed processing architecture, offers significant advantages in handling large datasets.

In the context of trajectory data preprocessing, Hadoop's distributed system architecture has facilitated the transition from traditional trajectory data processing to big data platforms. There have been significant developments in trajectory data processing on the Hadoop platform, particularly in map matching, data cleaning and data clustering. Various works of literature has explored the improvement of traditional clustering methods using the Hadoop distributed architecture.

Apart from the MapReduce model in Hadoop, there are other distributed processing architectures similar to MapReduce that can be used for trajectory data preprocessing. Two notable examples are the Spark platform and the Flink architecture. Spark is a general-purpose parallel computing framework similar to Hadoop MapReduce, utilizing in-memory distributed datasets, making it particularly well-suited for optimizing iterative

workloads, which is especially important for trajectory data pattern mining. Flink's unified stream and batch architecture has become the new standard for real-time stream processing, particularly suitable for millisecond-level response scenarios such as intelligent transportation. Stream processing is becoming increasingly important in the field of trajectory big data processing, especially for handling real-time trajectory data that cannot be stored in advance, such as intelligent transportation and location-based service recommendations.

Furthermore, GPU-based parallel architectures have also been utilized for trajectory data processing. Researchers have explored the use of GPU-based distributed architectures for compression, map matching, and other applications. Additionally, GPU-based parallel architectures have shown promise in accelerating trajectory data indexing, clustering, and pattern mining, such as using the Fréchet distance for sub-trajectory clustering. Future work should focus on establishing reproducible benchmarking frameworks to quantify the performance of these parallel paradigms under realistic data loads (e.g., city-wide V2X settings). The development of a 'data-rate–latency–resource' design map that outlines achievable performance targets for different preprocessing modules remains a critical and open research objective for the community.

(2) Deep-Learning-based Preprocessing

Current trajectory segmentation and map-matching methods often rely on handcrafted features and strong assumptions—such as Gaussian noise in HMMs or fixed thresholds in segmentation—which limit their adaptability across diverse environments (Sections 3.3 and 3.4).

Deep learning approaches, such as spatio-temporal transformers, graph neural networks, and semi-supervised autoencoders, can learn robust representations directly from data, reducing the dependence on heuristic models. These are particularly promising for low-sampling-rate trajectories (Section 3.4.3) and complex urban networks where traditional models degrade. Future work should also focus on interpretable deep learning to enhance model transparency in critical applications such as traffic management and urban planning.

Transformers and graph neural networks are reshaping the paradigm of trajectory processing, breaking through the sequence modeling limitations of traditional RNNs and CNNs. Spatio-temporal Transformers capture long-range dependencies through self-attention mechanisms, while Graphormer directly models the topology of mobile networks. However, several significant challenges remain to be addressed in future research, such as data representation, multimodal fusion, limited labeled data, and interpretability.

The temporal sequential nature of trajectory data necessitates a suitable representation format for deep learning models. In future work, there is a need to design effective input representations to capture the temporal and spatial characteristics of trajectories. In addition, trajectory datasets can be extensive and complex, requiring significant computational resources to train deep learning models. Developing efficient algorithms and architectures that can handle the scale and complexity of trajectory data is another challenge.

More critically, labeled trajectory data for training deep learning models may be scarce or expensive to obtain. Developing semi-supervised or unsupervised learning techniques in trajectory data preprocessing will be an important research direction in the future. Furthermore, deep learning models often lack interpretability, making it difficult to understand the reasoning behind their predictions. Developing new techniques to enhance the interpretability of deep learning models for trajectory data preprocessing has significant practical application value.

With the advancement in the utilization of large-scale models, such as those used in natural language processing, for semantic description of trajectories, by leveraging these models, key keywords and semantic information can be extracted from trajectory

data. Resolving semantic conflicts and modeling trajectories can then be combined to obtain a topic-based distribution of keywords that describe user locations. Generating trajectory descriptions using text generation techniques will be a crucial research direction in this context.

The shift towards deep learning requires concrete input-representation paradigms and strategies to address the "black-box" problem. For example, the input-representation paradigms are as follows:

Spatio-temporal Tokenization: A trajectory is treated as a sequence of tokens, where each token is an embedding of a spatio-temporal point [latitude, longitude, and timestamp]. This sequence can be fed directly into Transformer models.

Road-Graph Embeddings: For map matching and network-constrained tasks, the road network is preprocessed using a Graph Neural Network (GNN) to generate node/edge embeddings. Trajectory points are then contextualized with these graph embeddings.

Semantic Embedding Tiers: Beyond raw coordinates, input feature vectors are enriched with derived semantic features such as [speed, acceleration, heading change, and stop_likelihood], forming a multi-tiered embedding that captures motion context.

(3) Semantic Trajectory Tagging with Context Awareness

While semantic compression and segmentation methods (Sections 3.2.3 and 3.3.2) improve interpretability, they often fail to capture dynamic contextual factors such as user activity, transportation mode transitions, or environmental conditions.

Future systems should incorporate multi-source data fusion and temporal context modeling—using techniques from NLP and computer vision—to infer richer semantics and support more accurate trajectory summarization and querying [137]. This is essential for applications such as the personalized travel recommendation and human mobility analysis, where semantic ambiguity remains a major barrier (Section 3.3.3). In future studies, semantic annotation still faces the following challenges:

Label ambiguity and variability: Trajectories can exhibit ambiguity and variability in terms of their semantic interpretation. For example, a trajectory segment can represent both walking and running, depending on the user's speed. Similarly, a trajectory passing through a shopping mall can correspond to various activities like browsing, purchasing, or window shopping. Dealing with such ambiguity and variability poses a challenge in accurately labeling trajectories.

Contextual information integration: Trajectory semantics often rely on contextual information such as geographical features, temporal patterns, and environmental factors. Integrating this contextual information with trajectory data is essential for accurate semantic labeling. However, capturing and effectively utilizing these contextual factors is a complex task that requires advanced data fusion and integration techniques.

Labeling scalability and efficiency: As the volume and velocity of trajectory data increase, scalable and efficient methods for semantic labeling become crucial. Traditional manual labeling approaches are not feasible for large-scale datasets. Therefore, there is a need to develop automated and scalable techniques that can handle massive amounts of trajectory data in real time or near real time.

Transferability and generalization: Trajectory semantic labeling models trained on one dataset or geographical region may not generalize well to different datasets or regions due to variations in user behavior, transportation infrastructure, and cultural factors. Developing transferable and generalizable labeling models is essential for practical applications that operate across diverse datasets and locations.

(4) Trajectory Data Privacy in the Era of Cloud and AI

As trajectory data are increasingly used in cloud-based and AI-driven services, privacy risks escalate. Existing methods seldom address the trade-off between data utility and privacy, especially when dealing with sensitive locations (Section 3.1) or published trajectories (Section 3.2).

Future research should develop privacy-aware learning techniques, federated learning frameworks, and lightweight cryptographic schemes that allow for useful analysis without exposing raw user data [138]. Special attention should be paid to trajectory data publishing in cloud environments, where privacy guarantees must be maintained without compromising analytical value.

Data privacy protection in trajectory data preprocessing involves several aspects. Firstly, one manifestation is the development of privacy evaluation systems and personalized protection mechanisms. This includes assessing the privacy protection level and ensuring performance quality in protecting trajectory data. Additionally, providing personalized privacy protection modes based on different semantic environments of trajectory data is crucial. Secondly, the challenge lies in encryption techniques for handling sensitive location information. With the presence of multiple mobile terminals and frequent location updates, the performance of encryption can be affected. Optimizing encryption techniques to ensure the efficient and secure protection of sensitive location data is a key challenge. Another aspect is the need to balance the protection of sensitive and non-sensitive data, and find the right balance between protecting sensitive data and maintaining the quality of published data. It is important to consider the semantic features of trajectory data and the backgrounds of potential attackers to effectively handle both types of data. Another future research is the improvement of privacy protection in the context of cloud data services. As location data is often outsourced to cloud service providers, ensuring the privacy and integrity of user data becomes crucial. Encryption techniques specific to cloud service providers and techniques for verifying the integrity of retrieval results are areas of interest.

Overall, the manifestation of data privacy protection in trajectory data preprocessing involves evaluation systems, personalized protection, encryption techniques, and handling sensitive and non-sensitive data. The challenges include performance optimization, balancing protection and data quality, and addressing security issues in data mining and cloud services. Future research focuses on privacy protection in trajectory data mining and enhancing privacy protection in the context of cloud services.

6. Conclusions

In this paper, we have presented a review of papers on the topic of trajectory data preprocessing published over the past several decades. Based on this, we provide a comprehensive summary of the main techniques and algorithms used in trajectory data preprocessing in recent years, focusing on trajectory data cleaning, data compression, data segmentation, and map matching. The aim of this article is to assist researchers in quickly and comprehensively understanding the key technologies involved in trajectory data preprocessing. Furthermore, we explore the emerging trends and propose potential research topics in the field of trajectory preprocessing. Specifically, we highlight the research prospects and key technological challenges in trajectory preprocessing, such as efficient computation, deep-learning-based preprocessing, semantic trajectory tagging, and data privacy protection. This provides insights and solutions for future research in this field. These proposed discussions and analyses aim to attract more researchers to delve into the field of trajectory preprocessing.

Author Contributions: P.L.: conceptualization, methodology, validation, investigation, data curation, formal analysis, writing—original draft, and writing—review and editing. Z.T.: conceptualization, methodology, investigation, data curation, writing—review and editing, supervision, poject adminis-

tration, and funding acquisition. Y.Y.: conceptualization, methodology, investigation, data curation, and writing—review and editing. Y.L.: conceptualization, methodology, investigation, data curation, and writing—review and editing, supervision, project administration, and funding acquisition. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Research Team Cultivation Project of Zhengzhou University, Open Foundation of Key Laboratory of Transport Industry of Big Data Application Technologies for Comprehensive Transport (2022B1201), and the Key Scientific Research Project of Colleges and Universities in Henan Province (24A520045).

Data Availability Statement: No new data were created or analyzed in this study.

Conflicts of Interest: All authors declare that this research was carried out without any commercial or financial relationships that could be perceived as potential conflicts of interest. They also confirm that they have no competing interests and take full responsibility for the integrity of the work presented.

Abbreviations

TD-TR	Top-Down Time-Ratio
DPTS	Direction Preserving Trajectory Simplification
CTEV	Compression Technique Enhanced Version
POI	Points Of Interest
TS	Time-Synchronous
BOPW	Before Opening Window
NOPW	Normal Opening Window
STTrace	Sampling Trajectory Threshold Race
SQUISH	Spatial Quality Simplification Heuristic
SQUISH-E	Enhanced Spatial Quality Simplification Heuristic
BQS	Bounded Quadrant System
ROPW	Reverse Order Processing Window
SCG	Snapping Configuration Graph
FPA	Following Path Algorithm
MMTC	Map-Matched Trajectory Compression
MDL	Minimal Description Length
PRESS	Paralleled Road Network Based Trajectory Compression
COMPRESS	Comprehensive Paralleled Road Network Based Trajectory Compression
CiNCT	Compressed-Index for Network Constrained Trajectories
VTracer	Vehicle Tracing
STC	Semantic Trajectory Compression
EHSTC	Enhanced Semantic Trajectory Compression
SATC	Semantic-Aware Trajectory Compression
GR-B	Gpstrajectory Compression
STSS	Semantics Based Trajectory Segmentation Simplification
ROCE	Region Based Online Trajectory Compression with Error Bounded
SPD	Stay Point Detection
WS-II	Wise Sliding Window Segmentation
TRACLUS	Trajectory Clustering
SMoT	Simplification Method of Single Stop Segment of Trajectory
CB-SMoT	Clustering-Based Simplification Method of Single Stop Segment of Trajectory
DB- SMoT	Direction-Based Simplification Method of Single Stop Segment of Trajectory
SQE	Sum Of Quadratic Error
WKM	Warped K-Means Model
SeTraStream	Semantic-aware trajectory construction over streaming movement
GRASP-UTS	Greedy Randomized Adaptive Search Procedure for Unsupervised Trajectory Segmentation

TS-TF	Trajectory Segmentation Based on Multiple Motion Features
OWS	Octal Window Segmentation
SWS	Sliding Window Segmentation
CNN	Convolutional Neural Network
BTCN	Bayesian Temporal Convolutional Network
RGRASP-SemTS	Reactive Greedy Randomized Search Strategy for Semi-Supervised Semantic Trajectory Segmentation
SECA	Semi-Supervised Convolutional Autoencoder
MM	Map Matching
MHT	Multiple Hypothesis Tracking
ATMM	Advanced Topological Map Matching
HFTMM	High-Frequency Trajectory Map Matching
MHT-MM	Map-Matching Method Using the Multiple Hypothesis Technique
MDP-MM	Map-Matching Algorithm for Large-Scale Low-Frequency Floating Car Data
ST-CRF	Spatial And Temporal Conditional Random Field
MCM	Multiple Candidate Matching
PMHT-MM	Probabilistic Multiple Hypotheses Tracking Map Matching
MCMC	Markov Chain Monte Carlo
HMM	Hidden Markov Map Matching
OHMM	Online Map-Matching Based on Hidden Markov Model
FMM	Fast Map Matching, An Algorithm Integrating Hidden Markov Model
OM2	Off-Line Map-Matching
INC-RB	Incremental Route Inference Algorithm With Rollback
ST-Matching	Spatial Temporal Map-Matching
IVMM	Interactive Voting-Based Map Matching
STP-IWC	Spatio-Temporal Proximity and Improved Weighted Circle
AMM	Adaptive Online Map Matching
HRIS	History-Based Route InferenceSystem
DMM	Deep-Learning-Based Map-Matching
TMM	Transformer-Based Map-Matching
L2MM	Learning To Map Matching with Deep Models

References

1. Wang, S.; Li, L.; Ma, W.; Chen, X. Trajectory analysis for on-demand services: A survey focusing on spatial-temporal demand and supply patterns. *Transp. Res. Part C Emerg. Technol.* **2019**, *108*, 74–99. [CrossRef]
2. Xiong, W.; Wang, X.; Li, H. Efficient large-scale GPS trajectory compression on spark: A pipeline-based approach. *Electronics* **2023**, *12*, 3569. [CrossRef]
3. Guo, P. Optimized Unsupervised Semantic Trajectory Mining for Personalized Tourism Recommendations. *Informatica* **2025**, *49*. [CrossRef]
4. Zhang, P. Distributed Computing and Unsupervised Deep Learning for Analyzing Human Travel Behaviors Using Big Trajectory Data. University of Maryland, College Park. 2025. Available online: https://www.proquest.com/dissertations-theses/distributed-computing-unsupervised-deep-learning/docview/3250258686/se-2 (accessed on 22 November 2025).
5. Sheng, H.; Wang, T.; Luo, Y.; Liang, H. A review of trajectory data preprocessing and mining technology research. In Proceedings of the 2024 4th International Conference on Big Data, Artificial Intelligence and Risk Management, Shanghai, China, 19–21 January 2024; pp. 45–50. [CrossRef]
6. Chen, J.; Zhang, H.; Li, W.; Shibasaki, R. Spatio-temporal data preprocessing technologies. In *Big Data and Mobility as a Service*; Elsevier: Amsterdam, The Netherlands, 2022; pp. 25–75. [CrossRef]
7. Feng, Z.; Zhu, Y. A survey on trajectory data mining: Techniques and applications. *IEEE Access* **2016**, *4*, 2056–2067. [CrossRef]
8. Mazimpaka, J.D.; Timpf, S. Trajectory data mining: A review of methods and applications. *J. Spat. Inf. Sci.* **2016**, *13*, 61–99. [CrossRef]
9. Ribeiro de Almeida, D.; de Souza Baptista, C.; Gomes de Andrade, F.; Soares, A. A survey on big data for trajectory analytics. *ISPRS Int. J. Geo Inf.* **2020**, *9*, 88. [CrossRef]
10. Wang, D.; Miwa, T.; Morikawa, T. Big trajectory data mining: A survey of methods, applications, and services. *Sensors* **2020**, *20*, 4571. [CrossRef] [PubMed]

11. Wang, S.; Bao, Z.; Culpepper, J.S.; Cong, G. A survey on trajectory data management, analytics, and learning. *ACM Comput. Surv.* **2021**, *54*, 1–36. [CrossRef]
12. Zheng, Y. Trajectory data mining: An overview. *ACM Trans. Intell. Syst. Technol.* **2015**, *6*, 1–41. [CrossRef]
13. Lee, W.-C.; Krumm, J. Trajectory preprocessing. In *Computing with Spatial Trajectories*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 3–33. [CrossRef]
14. Amigo, D.; Sánchez Pedroche, D.; García, J.; Molina, J.M. Review and classification of trajectory summarisation algorithms: From compression to segmentation. *Int. J. Distrib. Sens. Netw.* **2021**, *17*, 15501477211050729. [CrossRef]
15. Sun, P.; Xia, S.; Yuan, G.; Li, D. An overview of moving object trajectory compression algorithms. *Math. Probl. Eng.* **2016**, *5*, 1–13. [CrossRef]
16. Muckell, J.; Olsen, P.W.; Hwang, J.-H.; Lawson, C.T.; Ravi, S. Compression of trajectory data: A comprehensive evaluation and new approach. *GeoInformatica* **2014**, *18*, 435–460. [CrossRef]
17. Chao, P.; Xu, Y.; Hua, W.; Zhou, X. A survey on map-matching algorithms. In Proceedings of the Databases Theory and Applications: 31st Australasian Database Conference, ADC 2020, Melbourne, VIC, Australia, 3–7 February 2020; pp. 121–133. [CrossRef]
18. Huang, Z.; Qiao, S.; Han, N.; Yuan, C.A.; Song, X.; Xiao, Y. Survey on vehicle map matching techniques. *CAAI Trans. Intell. Technol.* **2021**, *6*, 55–71. [CrossRef]
19. Hashemi, M.; Karimi, H.A. A critical review of real-time map-matching algorithms: Current issues and future directions. *Comput. Environ. Urban Syst.* **2014**, *48*, 153–165. [CrossRef]
20. Sousa, R.S.D.; Boukerche, A.; Loureiro, A.A. Vehicle trajectory similarity: Models, methods, and applications. *ACM Comput. Surv.* **2020**, *53*, 1–32. [CrossRef]
21. Li, H.; Ma, D.; Yan, Z.; Fu, J.; Zeng, M.; Bao, W. Algorithm of Vehicle's Data Cleaning and Monitoring. In *Journal of Physics: Conference Series*; IOP Publishing: Bristol, UK, 2021; Volume 1828, p. 012052. [CrossRef]
22. Xia, X.; Meng, Z.; Han, X.; Li, H.; Tsukiji, T.; Xu, R.; Zheng, Z.; Ma, J. An automated driving systems data acquisition and analytics platform. *Transp. Res. Part C Emerg. Technol.* **2023**, *151*, 104120. [CrossRef]
23. Zhu, S.; Yue, H.; Suzuki, T.; Kim, I.; Yu, L.; Lan, Q. A UWB/INS Trajectory Tracking System Application in a Cycling Safety Study. *Sensors* **2023**, *23*, 3629. [CrossRef]
24. Khodarahmi, M.; Maihami, V. A review on Kalman filter models. *Arch. Comput. Methods Eng.* **2023**, *30*, 727–747. [CrossRef]
25. Liu, Y.; Yang, Z. Trajectory Smoothing Algorithm Based on Kalman Filter. In Proceedings of the 2023 7th International Conference on Machine Vision and Information Technology (CMVIT), Guangzhou, China, 24–26 February 2023; pp. 52–56. [CrossRef]
26. Yuan, G.; Zhu, M.; Qiao, S.; Wang, Z.; Zhang, L. Sparse high-noise GPS trajectory data compression and recovery based on compressed sensing. *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.* **2018**, *101*, 811–821. [CrossRef]
27. Feng, D.; Zhang, H.; Song, X. Noise filter method for mobile trajectory data. In *Handbook of Mobility Data Mining*; Elsevier: Amsterdam, The Netherlands, 2023; pp. 35–50. [CrossRef]
28. Wu, R.; Luo, G.; Shao, J.; Tian, L.; Peng, C. Location prediction on trajectory data: A review. *Big Data Min. Anal.* **2018**, *1*, 108–127. [CrossRef]
29. Marczak, F.; Buisson, C. New filtering method for trajectory measurement errors and its comparison with existing methods. *Transp. Res. Rec.* **2012**, *2315*, 35–46. [CrossRef]
30. Vaibhav, M.; Rahul, K. Trajectory prediction and tracking using a multi-behaviour social particle filter. *Appl. Intell.* **2022**, *52*, 7158–7200. [CrossRef]
31. Zhang, J.; Sun, Y. An Automatic Data Cleaning Method for GPS Trajectory Data on Didi Chuxing GAIA Open Dataset Using Machine Learning Algorithms. In Proceedings of the 2019 6th International Conference on Systems and Informatics (ICSAI), Shanghai, China, 2–4 November 2019; pp. 1522–1526. [CrossRef]
32. Li, L.; Chen, X.; Liu, Q.; Bao, Z. A data-driven approach for GPS trajectory data cleaning. In Proceedings of the Database Systems for Advanced Applications: 25th International Conference, DASFAA 2020, Jeju, Republic of Korea, 24–27 September 2020; Springer: Cham, Switzerland, 2020; pp. 3–19. [CrossRef]
33. Xie, Y.; Xu, F.; Wang, Q.; Han, W. Data Construction Method of Unmanned Underwater Vehicle Test Scene Based on Linear Interpolation. In Proceedings of the 2021 IEEE International Conference on Unmanned Systems (ICUS), Beijing, China, 15–17 October 2021; pp. 260–265. [CrossRef]
34. Early, J.J.; Sykulski, A.M. Smoothing and interpolating noisy GPS data with smoothing splines. *J. Atmos. Ocean. Technol.* **2020**, *37*, 449–465. [CrossRef]
35. Ambrósio, J.; Antunes, P.; Pombo, J. On the requirements of interpolating polynomials for path motion constraints. In *Interdisciplinary Applications of Kinematics, Proceedings of the International Conference, Lima, Peru, 9–11 September 2014*; Springer: Cham, Switzerland; pp. 179–197. [CrossRef]
36. Markovsky, I.; Dörfler, F. Data-driven dynamic interpolation and approximation. *Automatica* **2022**, *135*, 110008. [CrossRef]

37. Guo, S.; Mou, J.; Chen, L.; Chen, P. Improved kinematic interpolation for AIS trajectory reconstruction. *Ocean. Eng.* **2021**, *234*, 109256. [CrossRef]
38. Venthuruthiyil, S.P.; Chunchu, M. Vehicle path reconstruction using Recursively Ensembled Low-pass filter (RELP) and adaptive tri-cubic kernel smoother. *Transp. Res. Part C Emerg. Technol.* **2020**, *120*, 102847. [CrossRef]
39. Zhao, J.; Yang, X.; Zhang, C. Vehicle trajectory reconstruction for intersections: An integrated wavelet transform and Savitzky-Golay filter approach. *Transp. A Transp. Sci.* **2023**, *20*, 2163207. [CrossRef]
40. Aftab, W.; Mihaylova, L. A learning Gaussian process approach for maneuvering target tracking and smoothing. *IEEE Trans. Aerosp. Electron. Syst.* **2020**, *57*, 278–292. [CrossRef]
41. Li, L.; Pagnucco, M.; Song, Y. Graph-based spatial transformer with memory replay for multi-future pedestrian trajectory prediction. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 18–22 June 2023; Vancouver, BC, Canada; IEEE: New York, NY, USA, 2022; pp. 2231–2241. Available online: https://openaccess.thecvf.com/content/CVPR2022/html/Li_Graph-Based_Spatial_Transformer_With_Memory_Replay_for_Multi-Future_Pedestrian_Trajectory_CVPR_2022_paper.html (accessed on 22 November 2025).
42. Lasota, P.A.; Shah, J.A. Bayesian estimator for partial trajectory alignment. In Proceedings of the Robotics: Science and Systems, Freiburg, Germany, 22–26 June 2019. [CrossRef]
43. Taylor, J.; Zhou, X.; Rouphail, N.M.; Porter, R.J. Method for investigating intradriver heterogeneity using vehicle trajectory data: A dynamic time warping approach. *Transp. Res. Part B Methodol.* **2015**, *73*, 59–80. [CrossRef]
44. Chen, L.; Shang, S.; Feng, S.; Kalnis, P. Parallel subtrajectory alignment over massive-scale trajectory data. In Proceedings of the International Joint Conferences on Artificial Intelligence Organization, Montreal, QC, Canada, 19–27 August 2021. [CrossRef]
45. Reyes Zambrano, G. GPS trajectory compression algorithm. In Proceedings of the Computer and Communication Engineering: First International Conference, ICCCE 2018, Guayaquil, Ecuador, 25–27 October 2018; Springer: Berlin/Heidelberg, Germany, 2019; pp. 57–69. [CrossRef]
46. 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* **1973**, *10*, 112–122. [CrossRef]
47. Hershberger, J.E.; Snoeyink, J. *Speeding Up the Douglas-Peucker Line-Simplification Algorithm*; University of British Columbia: Vancouver, BC, Canada, 1992.
48. Meratnia, N.; de By, R.A. Spatiotemporal compression techniques for moving point objects. In Proceedings of the Advances in Database Technology-EDBT 2004: 9th International Conference on Extending Database Technology, Heraklion, Crete, Greece, 14–18 March 2004; Springer: Berlin/Heidelberg, Germany, 2004; pp. 765–782. [CrossRef]
49. Long, C.; Wong, R.C.-W.; Jagadish, H. Direction-preserving trajectory simplification. *Proc. VLDB Endow.* **2013**, *6*, 949–960. [CrossRef]
50. Bashir, M.; Ashraf, J.; Habib, A.; Muzammil, M. An intelligent linear time trajectory data compression framework for smart planning of sustainable metropolitan cities. *Trans. Emerg. Telecommun. Technol.* **2022**, *33*, e3886. [CrossRef]
51. Keogh, E.; Chu, S.; Hart, D.; Pazzani, M. An online algorithm for segmenting time series. In Proceedings of the 2001 IEEE international conference on data mining, San Jose, CA, USA, 29 November–2 December 2001; pp. 289–296. [CrossRef]
52. Potamias, M.; Patroumpas, K.; Sellis, T. Sampling trajectory streams with spatiotemporal criteria. In Proceedings of the 18th International Conference on Scientific and Statistical Database Management (SSDBM'06), Vienna, Austria, 3–5 July 2006; pp. 275–284. [CrossRef]
53. Muckell, J.; Hwang, J.-H.; Patil, V.; Lawson, C.T.; Ping, F.; Ravi, S. SQUISH: An online approach for GPS trajectory compression. In Proceedings of the 2nd International Conference on Computing for Geospatial Research & Applications, Washington, DC, USA, 23–25 May 2011; pp. 1–8. [CrossRef]
54. Liu, J.; Zhao, K.; Sommer, P.; Shang, S.; Kusy, B.; Jurdak, R. Bounded quadrant system: Error-bounded trajectory compression on the go. In Proceedings of the 2015 IEEE 31st International Conference on Data Engineering, Seoul, Republic of Korea, 13–17 April 2015; pp. 987–998. [CrossRef]
55. Li, S.; Zhang, K.; Yin, H.; Yin, D.; Zu, H.; Gao, H. ROPW: An online trajectory compression algorithm. In *Database Systems for Advanced Applications, Proceedings of the DASFAA 2021 International Workshops: BDQM, GDMA, MLDLDSA, MobiSocial, and MUST, Taipei, Taiwan, 11–14 April 2021*; Springer: Cham, Switzerland, 2021; pp. 16–28. [CrossRef]
56. Lin, X.; Ma, S.; Jiang, J.; Hou, Y.; Wo, T. Error bounded line simplification algorithms for trajectory compression: An experimental evaluation. *ACM Trans. Database Syst.* **2021**, *46*, 1–44. [CrossRef]
57. Makris, A.; Silva, C.L.d.; Bogorny, V.; Alvares, L.O.; Macedo, J.A.; Tserpes, K. Evaluating the effect of compressing algorithms for trajectory similarity and classification problems. *GeoInformatica* **2021**, *25*, 679–711. [CrossRef]
58. Cao, H.; Wolfson, O. Nonmaterialized motion information in transport networks. In Proceedings of the Database Theory-ICDT 2005: 10th International Conference, Edinburgh, UK, 5–7 January 2005; pp. 173–188. [CrossRef]

59. Lerin, P.M.; Yamamoto, D.; Takahashi, N. Encoding travel traces by using road networks and routing algorithms. In *Intelligent Interactive Multimedia: Systems and Services, Proceedings of the 5th International Conference on Intelligent Interactive Multimedia Systems and Services (IIMSS 2012), Gifu, Japan, 25–27 July 2012*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 233–243. [CrossRef]
60. Kellaris, G.; Pelekis, N.; Theodoridis, Y. Trajectory compression under network constraints. In Proceedings of the Advances in Spatial and Temporal Databases: 11th International Symposium, SSTD 2009, Aalborg, Denmark, 8–10 July 2009; Springer: Berlin/Heidelberg, Germany, 2009; pp. 392–398. [CrossRef]
61. Kellaris, G.; Pelekis, N.; Theodoridis, Y. Map-matched trajectory compression. *J. Syst. Softw.* **2013**, *86*, 1566–1579. [CrossRef]
62. Song, R.; Sun, W.; Zheng, B.; Zheng, Y. PRESS: A novel framework of trajectory compression in road networks. *arXiv* **2014**, arXiv:1402.1546. [CrossRef]
63. Han, Y.; Sun, W.; Zheng, B. COMPRESS: A comprehensive framework of trajectory compression in road networks. *ACM Trans. Database Syst.* **2017**, *42*, 1–49. [CrossRef]
64. Koide, S.; Tadokoro, Y.; Xiao, C.; Ishikawa, Y. CiNCT: Compression and retrieval for massive vehicular trajectories via relative movement labeling. In Proceedings of the 2018 IEEE 34th International Conference on Data Engineering (ICDE), Paris, France, 16–19 April 2018; pp. 1097–1108. [CrossRef]
65. Zhao, P.; Zhao, Q.; Zhang, C.; Su, G.; Zhang, Q.; Rao, W. CLEAN: Frequent pattern-based trajectory compression and computation on road networks. *China Commun.* **2020**, *17*, 119–136. [CrossRef]
66. Chen, Q.; Cao, J.; Xia, Y. Physics-enhanced pca for data compression in edge devices. *IEEE Trans. Green Commun. Netw.* **2022**, *6*, 1624–1634. [CrossRef]
67. Li, T.; Chen, L.; Jensen, C.S.; Pedersen, T.B. TRACE: Real-time compression of streaming trajectories in road networks. *Proc. VLDB Endow.* **2021**, *14*, 1175–1187. [CrossRef]
68. 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.* **2019**, *21*, 2012–2028. [CrossRef]
69. Chen, C.; Ding, Y.; Wang, Z.; Zhao, J.; Guo, B.; Zhang, D. VTracer: When online vehicle trajectory compression meets mobile edge computing. *IEEE Syst. J.* **2019**, *14*, 1635–1646. [CrossRef]
70. Schmid, F.; Richter, K.-F.; Laube, P. Semantic trajectory compression. In Proceedings of the Advances in Spatial and Temporal Databases: 11th International Symposium, SSTD 2009, Aalborg, Denmark, 8–10 July 2009; Springer: Berlin/Heidelberg, Germany, 2009; pp. 411–416. [CrossRef]
71. Richter, K.-F.; Schmid, F.; Laube, P. Semantic trajectory compression: Representing urban movement in a nutshell. *J. Spat. Inf. Sci.* **2012**, *4*, 3–30. [CrossRef]
72. Feng, S.; Xu, J.; Xu, M.; Zheng, N.; Zhang, X. EHSTC: An enhanced method for semantic trajectory compression. In Proceedings of the 4th ACM SIGSPATIAL International Workshop on GeoStreaming, Orlando, FL, USA, 5–8 November 2013; pp. 43–49. [CrossRef]
73. Su, H.; Zheng, K.; Zeng, K.; Huang, J.; Zhou, X. STMaker: A system to make sense of trajectory data. *Proc. VLDB Endow.* **2014**, *7*, 1701–1704. [CrossRef]
74. Su, H.; Zheng, K.; Zeng, K.; Huang, J.; Sadiq, S.; Yuan, N.J.; Zhou, X. Making sense of trajectory data: A partition-and-summarization approach. In Proceedings of the 2015 IEEE 31st International Conference on Data Engineering, Seoul, Republic of Korea, 13–17 April 2015; pp. 963–974. [CrossRef]
75. Liu, M.; He, G.; Long, Y. A semantics-based trajectory segmentation simplification method. *J. Geovisualization Spat. Anal.* **2021**, *5*, 19. [CrossRef]
76. Zhang, K.; Zhao, D.; Liu, W. Online vehicle trajectory compression algorithm based on motion pattern recognition. *IET Intell. Transp. Syst.* **2022**, *16*, 998–1010. [CrossRef]
77. Chen, H.; Chen, X. A trajectory ensemble-compression algorithm based on finite element method. *ISPRS Int. J. Geo-Inf.* **2021**, *10*, 334. [CrossRef]
78. Zhao, Y.; Shang, S.; Wang, Y.; Zheng, B.; Nguyen, Q.V.H.; Zheng, K. Rest: A reference-based framework for spatio-temporal trajectory compression. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, London, UK, 19–23 August 2018; pp. 2797–2806. [CrossRef]
79. Zheng, K.; Zhao, Y.; Lian, D.; Zheng, B.; Liu, G.; Zhou, X. Reference-based framework for spatio-temporal trajectory compression and query processing. *IEEE Trans. Knowl. Data Eng.* **2019**, *32*, 2227–2240. [CrossRef]
80. Ta, N.; Li, G.; Chen, B.; Feng, J. Semantic-aware trajectory compression with urban road network. In Proceedings of the Web-Age Information Management: 17th International Conference, WAIM 2016, Nanchang, China, 3–5 June 2016; Springer: Cham, Switzerland, 2016; pp. 124–136. [CrossRef]
81. Yin, H.; Gao, H.; Wang, B.; Li, S.; Li, J. Efficient trajectory compression and range query processing. *World Wide Web* **2022**, *25*, 1259–1285. [CrossRef]
82. Zheng, Y.; Zhang, L.; Ma, Z.; Xie, X.; Ma, W.-Y. Recommending friends and locations based on individual location history. *ACM Trans. Web* **2011**, *5*, 1–44. [CrossRef]

83. Etemad, M.; Etemad, Z.; Soares, A.; Bogorny, V.; Matwin, S.; Torgo, L. Wise sliding window segmentation: A classification-aided approach for trajectory segmentation. In Proceedings of the Advances in Artificial Intelligence: 33rd Canadian Conference on Artificial Intelligence, Canadian AI 2020, Ottawa, ON, Canada, 13–15 May 2020; Springer: Cham, Switzerland, 2020; pp. 208–219. [CrossRef]
84. Lee, J.-G.; Han, J.; Whang, K.-Y. Trajectory clustering: A partition-and-group framework. In Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, Beijing, China, 11–14 June 2007; pp. 593–604. [CrossRef]
85. Alvares, L.O.; Bogorny, V.; Kuijpers, B.; de Macedo, J.A.F.; Moelans, B.; Vaisman, A. A model for enriching trajectories with semantic geographical information. In Proceedings of the 15th Annual ACM International Symposium on Advances in Geographic Information Systems, Seattle, WA, USA, 7–9 November 2007; pp. 1–8. [CrossRef]
86. Palma, A.T.; Bogorny, V.; Kuijpers, B.; Alvares, L.O. A clustering-based approach for discovering interesting places in trajectories. In Proceedings of the 2008 ACM Symposium on Applied Computing, Fortaleza, Brazil, 16–20 March 2008; pp. 863–868. [CrossRef]
87. Rocha, J.A.M.; Times, V.C.; Oliveira, G.; Alvares, L.O.; Bogorny, V. DB-SMoT: A direction-based spatio-temporal clustering method. In Proceedings of the 2010 5th IEEE International Conference Intelligent Systems, London, UK, 7–9 July 2010; pp. 114–119. [CrossRef]
88. Leiva, L.A.; Vidal, E. Warped k-means: An algorithm to cluster sequentially-distributed data. *Inf. Sci.* **2013**, *237*, 196–210. [CrossRef]
89. Buchin, M.; Driemel, A.; Kreveld, M.V.; Sacristán Adinolfi, V. Segmenting trajectories: A framework and algorithms using spatiotemporal criteria. *J. Spat. Inf. Sci.* **2011**, *3*, 33–63. [CrossRef]
90. Buchin, M.; Driemel, A.; Van Kreveld, M.; Sacristán, V. An algorithmic framework for segmenting trajectories based on spatio-temporal criteria. In Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems, San Jose, CA, USA, 2–5 November 2010; pp. 202–211. [CrossRef]
91. Yan, Z.; Giatrakos, N.; Katsikaros, V.; Pelekis, N.; Theodoridis, Y. SeTraStream: Semantic-aware trajectory construction over streaming movement data. In Proceedings of the Advances in Spatial and Temporal Databases: 12th International Symposium, SSTD 2011, Minneapolis, MN, USA, 24–26 August 2011; Springer: Berlin/Heidelberg, Germany, 2011; pp. 367–385. [CrossRef]
92. Soares Júnior, A.; Moreno, B.N.; Times, V.C.; Matwin, S.; Cabral, L.d.A.F. GRASP-UTS: An algorithm for unsupervised trajectory segmentation. *Int. J. Geogr. Inf. Sci.* **2015**, *29*, 46–68. [CrossRef]
93. Xu, W.; Dong, S. Application of artificial intelligence in an unsupervised algorithm for trajectory segmentation based on multiple motion features. *Wirel. Commun. Mob. Comput.* **2022**, *2022*, 9540944. [CrossRef]
94. Etemad, M.; Júnior, A.S.; Hoseyni, A.; Rose, J.; Matwin, S. *A Trajectory Segmentation Algorithm Based on Interpolation-Based Change Detection Strategies*; EDBT/ICDT Workshops: Lisbon, Portugal, 2019; p. 58. [CrossRef]
95. Etemad, M.; Soares, A.; Etemad, E.; Rose, J.; Torgo, L.; Matwin, S. SWS: An unsupervised trajectory segmentation algorithm based on change detection with interpolation kernels. *GeoInformatica* **2021**, *25*, 269–289. [CrossRef]
96. Markos, C.; James, J.; Da Xu, R.Y. Capturing uncertainty in unsupervised GPS trajectory segmentation using Bayesian deep learning. In Proceedings of the AAAI Conference on Artificial Intelligence, Online, 2–9 February 2021; pp. 390–398. [CrossRef]
97. Dabiri, S.; Heaslip, K. Inferring transportation modes from GPS trajectories using a convolutional neural network. *Transp. Res. Part C: Emerg. Technol.* **2018**, *86*, 360–371. [CrossRef]
98. Junior, A.S.; Times, V.C.; Renso, C.; Matwin, S.; Cabral, L.A. A semi-supervised approach for the semantic segmentation of trajectories. In Proceedings of the 2018 19th IEEE International Conference on Mobile Data Management (MDM), Aalborg, Denmark, 25–28 June 2018; pp. 145–154. [CrossRef]
99. Dabiri, S.; Lu, C.-T.; Heaslip, K.; Reddy, C.K. Semi-supervised deep learning approach for transportation mode identification using GPS trajectory data. *IEEE Trans. Knowl. Data Eng.* **2019**, *32*, 1010–1023. [CrossRef]
100. Quddus, M.A.; Ochieng, W.Y.; Noland, R.B. Current map-matching algorithms for transport applications: State-of-the art and future research directions. *Transp. Res. Part C Emerg. Technol.* **2007**, *15*, 312–328. [CrossRef]
101. Wei, H.; Wang, Y.; Forman, G.; Zhu, Y. Map matching: Comparison of approaches using sparse and noisy data. In Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Orlando, FL, USA, 5–8 November 2013; pp. 444–447. [CrossRef]
102. Bernstein, D.; Kornhauser, A. An introduction to map matching for personal navigation assistants. *Transp. Res. Board* **1998**, *122*, 1082–1083. [CrossRef]
103. White, C.E.; Bernstein, D.; Kornhauser, A.L. Some map matching algorithms for personal navigation assistants. *Transp. Res. Part C: Emerg. Technol.* **2000**, *8*, 91–108. [CrossRef]
104. Taylor, G.; Blewitt, G.; Steup, D.; Corbett, S.; Car, A. Road reduction filtering for GPS--GIS navigation. *Trans. GIS* **2001**, *5*, 193–207. [CrossRef]
105. Brakatsoulas, S.; Pfoser, D.; Salas, R.; Wenk, C. On map-matching vehicle tracking data. In Proceedings of the 31st International Conference on Very Large Data Bases, Trondheim, Norway, 30 August–2 September 2005; pp. 853–864. Available online: https://dl.acm.org/doi/epdf/10.5555/1083592.1083691 (accessed on 22 November 2025).

106. Alt, H.; Efrat, A.; Rote, G.; Wenk, C. Matching planar maps. *J. Algorithms* **2003**, *49*, 262–283. [CrossRef]
107. Quddus, M.A.; Ochieng, W.Y.; Zhao, L.; Noland, R.B. A general map matching algorithm for transport telematics applications. *GPS Solut.* **2003**, *7*, 157–167. [CrossRef]
108. Chawathe, S.S. Segment-based map matching. In Proceedings of the 2007 IEEE Intelligent Vehicles Symposium, Istanbul, Turkey, 13–15 June 2007; pp. 1190–1197. [CrossRef]
109. Zhao, X.; Cheng, X.; Zhou, J.; Xu, Z.; Dey, N.; Ashour, A.S.; Satapathy, S.C. Advanced topological map matching algorithm based on D–S theory. *Arab. J. Sci. Eng.* **2018**, *43*, 3863–3874. [CrossRef]
110. Yu, Q.; Hu, F.; Ye, Z.; Chen, C.; Sun, L.; Luo, Y. High-frequency trajectory map matching algorithm based on road network topology. *IEEE Trans. Intell. Transp. Syst.* **2022**, *23*, 17530–17545. [CrossRef]
111. Pyo, J.-S.; Shin, D.-H.; Sung, T.-K. Development of a map matching method using the multiple hypothesis technique. In Proceedings of the ITSC 2001. 2001 IEEE Intelligent Transportation Systems. Proceedings (Cat. No. 01TH8585), Oakland, CA, USA, 25–29 August 2001; pp. 23–27. [CrossRef]
112. Marchal, F.; Hackney, J.; Axhausen, K.W. Efficient map matching of large global positioning system data sets: Tests on speed-monitoring experiment in Zürich. *Transp. Res. Rec.* **2005**, *1935*, 93–100. [CrossRef]
113. Schuessler, N.; Axhausen, K.W. *Map-Matching of GPS Traces on High-Resolution Navigation Networks Using the Multiple Hypothesis Technique (MHT)*; Arbeitsberichte Verkehrs-und Raumplanung; ETH Zurich, Institute for Transport Planning and Systems: Zurich, Switzerland, 2009; Volume 568. [CrossRef]
114. Chen, B.Y.; Yuan, H.; Li, Q.; Lam, W.H.; Shaw, S.-L.; Yan, K. Map-matching algorithm for large-scale low-frequency floating car data. *Int. J. Geogr. Inf. Sci.* **2014**, *28*, 22–38. [CrossRef]
115. Liu, X.; Liu, K.; Li, M.; Lu, F. A ST-CRF map-matching method for low-frequency floating car data. *IEEE Trans. Intell. Transp. Syst.* **2016**, *18*, 1241–1254. [CrossRef]
116. Li, W.; Wang, Y.; Li, D.; Xu, X. A robust map matching method by considering memorized multiple matching candidates. *Theor. Comput. Sci.* **2023**, *941*, 104–120. [CrossRef]
117. Ning, S.; Liu, H.; Zhang, S.; Jiang, Y.; Han, J.; Liu, S.; Fang, J.; Tan, N.; Chai, H.; Zhang, B. Estimation and Prediction of Road Free Flow Speed with More Efficient DNN Map Matching Results. 2022. Available online: http://urban-computing.com/urbcomp2022/file/UrbComp2022_paper_0498.pdf (accessed on 22 November 2025).
118. Newson, P.; Krumm, J. Hidden Markov map matching through noise and sparseness. In Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Seattle, WA, USA, 4–6 November 2009; pp. 336–343. [CrossRef]
119. Goh, C.Y.; Dauwels, J.; Mitrovic, N.; Asif, M.T.; Oran, A.; Jaillet, P. Online map-matching based on hidden markov model for real-time traffic sensing applications. In Proceedings of the 2012 15th International IEEE Conference on Intelligent Transportation Systems, Anchorage, AK, USA, 16–19 September 2012; pp. 776–781. [CrossRef]
120. Yang, C.; Gidofalvi, G. Fast map matching, an algorithm integrating hidden Markov model with precomputation. *Int. J. Geogr. Inf. Sci.* **2018**, *32*, 547–570. [CrossRef]
121. Xie, Y.; Zhou, K.; Miao, F.; Zhang, Q. High-Accuracy off-line map-matching of trajectory network division based on weight adaptation HMM. *IEEE Access* **2020**, *8*, 7256–7266. [CrossRef]
122. Luo, L.; Hou, X.; Cai, W.; Guo, B. Incremental route inference from low-sampling GPS data: An opportunistic approach to online map matching. *Inf. Sci.* **2020**, *512*, 1407–1423. [CrossRef]
123. Lou, Y.; Zhang, C.; Zheng, Y.; Xie, X.; Wang, W.; Huang, Y. Map-matching for low-sampling-rate GPS trajectorie. In Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Seattle, WA, USA, 4–6 November 2009; pp. 352–361. [CrossRef]
124. Yuan, J.; Zheng, Y.; Zhang, C.; Xie, X.; Sun, G.-Z. An interactive-voting based map matching algorithm. In Proceedings of the 2010 Eleventh International Conference on Mobile Data Management, Kansas City, MO, USA, 23–26 May 2010; pp. 43–52. [CrossRef]
125. Teng, W.; Wang, Y. Real-time map matching: A new algorithm integrating spatio-temporal proximity and improved weighted circle. *Open Geosci.* **2019**, *11*, 288–297. [CrossRef]
126. Hu, H.; Qian, S.; Ouyang, J.; Cao, J.; Han, H.; Wang, J.; Chen, Y. AMM: An Adaptive Online Map Matching Algorithm. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 5039–5051. [CrossRef]
127. Zheng, K.; Zheng, Y.; Xie, X.; Zhou, X. Reducing uncertainty of low-sampling-rate trajectories. In Proceedings of the 2012 IEEE 28th International Conference on Data Engineering, Washington, DC, USA, 1–5 April 2012; pp. 1144–1155. [CrossRef]
128. Jagadeesh, G.R.; Srikanthan, T. Online map-matching of noisy and sparse location data with hidden Markov and route choice models. *IEEE Trans. Intell. Transp. Syst.* **2017**, *18*, 2423–2434. [CrossRef]
129. Zhao, K.; Feng, J.; Xu, Z.; Xia, T.; Chen, L.; Sun, F.; Guo, D.; Jin, D.; Li, Y. DeepMM: Deep learning based map matching with data augmentation. In Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Chicago, IL, USA, 5–8 November 2019; pp. 452–455. [CrossRef]

130. Jin, Z.; Kim, J.; Yeo, H.; Choi, S. Transformer-based map-matching model with limited labeled data using transfer-learning approach. *Transp. Res. Part C Emerg. Technol.* **2022**, *140*, 103668. [CrossRef]
131. Jiang, Z.; Huang, A.; Qi, G.; Guan, W. A Framework of Travel Mode Identification Fusing Deep Learning and Map-Matching Algorithm. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 6401–6415. [CrossRef]
132. Wang, X.; Gilliam, C.; Kealy, A.; Close, J.; Moran, B. Probabilistic map matching for robust inertial navigation aiding. *Navig. J. Inst. Navig.* **2023**, *70*, navi.583. [CrossRef]
133. Jiang, L.; Chen, C.-X.; Chen, C. L2mm: Learning to map matching with deep models for low-quality gps trajectory data. *ACM Trans. Knowl. Discov. Data* **2023**, *17*, 1–25. [CrossRef]
134. Harder, D.; Shoushtari, H.; Sternberg, H. Real-Time Map Matching with a Backtracking Particle Filter Using Geospatial Analysis. *Sensors* **2022**, *22*, 3289. [CrossRef]
135. Peker, A.U.; Tosun, O.; Acarman, T. Particle filter vehicle localization and map-matching using map topology. In Proceedings of the 2011 IEEE Intelligent Vehicles Symposium (IV), Baden-Baden, Germany, 5–9 June 2011; pp. 248–253. [CrossRef]
136. Yu, Z.; Wu, H.; Yin, Z.; Liu, K.; Zhang, R. Vessel trajectory segmentation: A survey. In Proceedings of the International Conference on Database Systems for Advanced Applications, Tianjin, China, 17–20 April 2023; Springer Nature: Cham, Switzerland, 2023; pp. 166–180. [CrossRef]
137. Ma, X.; Zhou, P.; He, X. Advances in multi-source navigation data fusion processing methods. *Mathematics* **2025**, *13*, 1485. [CrossRef]
138. Miranda-Pascual, À.; Guerra-Balboa, P.; Parra-Arnau, J.; Forné, J.; Strufe, T. An overview of proposals towards the privacy-preserving publication of trajectory data. *Int. J. Inf. Secur.* **2024**, *23*, 3711–3747. [CrossRef]