

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/340480414>

A Trajectory Collaboration Based Map Matching Approach for Low-Sampling-Rate GPS Trajectories

Article in Sensors · April 2020

DOI: 10.3390/s20072057

CITATION

1

READS

97

3 authors, including:



Ge Cui

The University of Calgary

10 PUBLICATIONS 50 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:



Personalized Route Recommendation [View project](#)

Article

A Trajectory Collaboration Based Map Matching Approach for Low-Sampling-Rate GPS Trajectories

Wentao Bian ¹, **Ge Cui** ^{2,3,*} and **Xin Wang** ^{1,3} ¹ School of Information Science and Technology, Northwest University, Xi'an 710127, China; wtbian@stumail.nwu.edu.cn (W.B.); xcwang@ucalgary.ca (X.W.)² Intelligent Transportation Systems Research Center, Wuhan University of Technology, Wuhan 430070, China³ Department of Geomatics Engineering, University of Calgary, Calgary, AB T2N1N4, Canada

* Correspondence: cuig@ucalgary.ca

Received: 10 February 2020; Accepted: 3 April 2020; Published: 6 April 2020



Abstract: GPS (Global Positioning System) trajectories with low sampling rates are prevalent in many applications. However, current map matching methods do not perform well for low-sampling-rate GPS trajectories due to the large uncertainty between consecutive GPS points. In this paper, a collaborative map matching method (CMM) is proposed for low-sampling-rate GPS trajectories. CMM processes GPS trajectories in batches. First, it groups similar GPS trajectories into clusters and then supplements the missing information by resampling. A collaborative GPS trajectory is then extracted for each cluster and matched to the road network, based on longest common subsequence (LCSS) distance. Experiments are conducted on a real GPS trajectory dataset and a simulated GPS trajectory dataset. The results show that the proposed CMM outperforms the baseline methods in both, effectiveness and efficiency.

Keywords: map matching; low-sampling-rate GPS trajectories; trajectory collaboration; trajectory clustering

1. Introduction

With the development of positioning technology, massive GPS trajectory data has been continuously generated from vehicles such as cars, taxis and buses. A GPS trajectory is a sequence of GPS points which records the spatial track of a moving object. As a GPS trajectory can deviate from its actual location in the road network, caused by device malfunction, urban canyons and other positioning errors, many map matching methods have been proposed to locate GPS trajectories onto road networks. The core problem of map matching is the uncertainty issue of GPS trajectories, including the uncertain path between consecutive GPS points, and this problem becomes more severe for low-sampling-rate GPS trajectories.

At present, GPS trajectories with low sampling rates (the time interval between consecutive GPS readings exceeds 1 min) are collected for many applications as this can conserve battery life and help track vehicles for a long period. The current map matching algorithms are developed to determine the correct path of low-sampling-rate GPS trajectories, based on various features, such as spatial [1,2], temporal [2], speed constraint [3,4] and turning information [5]. However, these algorithms do not perform well when sampling rates are very low, i.e., the sampling rate exceeds three or four minutes, especially in a dense road network. Spatial features (e.g., distance similarity) and temporal features (e.g., speed similarity) [2,6] are sometimes ineffective in identifying the correct path between consecutive GPS points, when they are far from each other. Figure 1 shows an example of the map matching process based on spatial and temporal features. In the figure, point **a** (22.559° N, 114.067° E, 3:10:00 p.m.) and point **b** (22.551° N, 114.079° E, 3:14:00 p.m.) are two consecutive GPS points of a low-sampling-rate

trajectory. Two candidate paths (shown in blue and red) are identified with the lengths of 2.2 km and 2.8 km, and the average speeds moving along the two paths between **a** and **b** are 33 km/h and 42 km/h, respectively. As shown in the figure, the speed limits of road segments on the blue path are (60 km/h, 60 km/h, 60 km/h and 60 km/h), and on the red path are (60 km/h, 70 km/h, 70 km/h and 60 km/h). According to the spatial and temporal similarities of the ST-Matching algorithm [2], the blue path is more likely to be the map matching result than the red path, since the length of the blue path is shorter than that of the red path. Additionally, the cosine similarity value between average speed and the speed limits of road segments on the blue path is larger than the cosine similarity value on the red path. However, the red path is the actual path taken by the user. The example shows that using only spatial and temporal features sometimes does not effectively reduce the large uncertainty and distinguish the correct path for GPS trajectories with very low sampling rates.

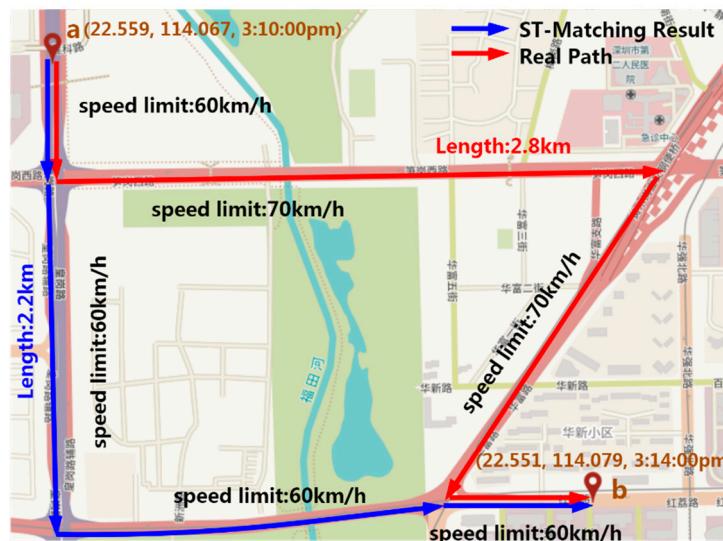


Figure 1. Example of map matching result based on spatial and temporal model for a low-sampling-rate GPS trajectory.

Other methods attempt to reduce the uncertainty of low-sampling-rate GPS trajectories with the addition of historical high-sampling-rate GPS trajectories. For instance, historical high-sampling-rate GPS trajectories are used to detect frequent path patterns, and the path of a low-sampling-rate GPS trajectory is estimated based on the discovered patterns [7,8]. However, historical high-sampling-rate GPS trajectories are not always available. Furthermore, the quality of these map matching methods heavily relies on the effectiveness of the discovered patterns. With rapid road development or traffic control, the frequent path patterns, discovered from historical GPS trajectory data, may not be applicable for the current situation. Therefore, creating effective map matching for low-sampling-rate GPS trajectories without auxiliary data continues to be a challenging problem.

This paper intends to solve the uncertainty issue of low-sampling-rate GPS trajectories through trajectory collaboration. Similar low-sampling-rate GPS trajectories are aggregated into clusters, and a collaborative GPS trajectory is then generated for map matching from each cluster with information supplementation. However, this method presents several challenges.

First, an effective clustering algorithm for low-sampling-rate GPS trajectories is significant. There are many methods of trajectory clustering, such as the partition and group method [9], trajectory clustering with Fréchet distance [10] and clustering algorithm for network constraint trajectories (NETSCAN) [11]. These algorithms are designed for different applications. For example, the clustering algorithm in [9] is designed for flock or typhoon trajectories, but does not consider road constraint in map matching. The clustering algorithms in [10,11] consider road constraint and calculate the geometric similarity between GPS points, but they are not applicable for low-sampling-rate trajectories.

The reason for this is that the distance between consecutive GPS points is large (maybe larger than 500 m) in low-sampling-rate GPS trajectories, so that the geometric similarity between trajectories would be small, even on the same path. Exploring an effective method of clustering low-sampling-rate GPS trajectories is worth studying.

Second, extracting useful information from a group of similar GPS trajectories to reduce the path uncertainty issue is required. After trajectory clustering, low-sampling-rate GPS trajectories in each cluster are still separated from each other and are not collaborated for map matching. An issue remains understanding how to use a group of similar GPS trajectories to reduce uncertainty by trajectory collaboration and further benefit map matching.

In this paper, a collaborative map matching algorithm called CMM is proposed to address low-sampling-rate GPS trajectories. CMM processes a group of similar GPS trajectories together, rather than as individual trajectories, which makes map matching more efficient. In the preprocessing step, CMM removes outliers from low-sampling-rate GPS trajectories. Spatial indices, including an R-tree and a Path-Forest, are built to facilitate candidate path search. Then, a DBSCAN (Density-Based Spatial Clustering of Applications with Noise) algorithm is extended to cluster GPS trajectories into different groups, based on path similarity instead of geometric similarity. Path similarity is designed for similarity measurement between low-sampling-rate GPS trajectories by comparing their candidate paths. Next, a resampling method is applied to generate a new collaborative GPS trajectory from each trajectory cluster. Finally, the generated collaborative GPS trajectory is matched to the path candidate with the highest LCSS (Longest Common Subsequence) similarity.

The specific contributions of this research are highlighted below:

1. This paper proposes a collaborative map matching algorithm called CMM to address low-sampling-rate GPS trajectories based on trajectory clustering and resampling. CMM does not rely on any additional information, such as historical high-sampling-rate GPS trajectories or traffic flow information.
2. DBSCAN clustering algorithm is extended to aggregate similar low-sampling-rate GPS trajectories together, based on path similarity, which can effectively decrease the negative impact of geometric similarity on clustering of low-sampling-rate GPS trajectories.
3. A resampling method, based on a sliding window, is proposed to generate a new collaborative GPS trajectory from each trajectory cluster. The generated collaborative GPS trajectory integrates the collective information of similar low-sampling-rate GPS trajectories.
4. Experiments have been conducted based on a simulated GPS trajectory dataset, and the results show that the proposed CMM algorithm outperforms the baseline methods in both, effectiveness and efficiency.

This paper is organized as follows: Section 2 introduces related works on map matching. Section 3 provides a detailed discussion on the proposed CMM algorithm. Section 4 discusses experiments based on both a real and a simulated GPS trajectory dataset. Section 5 presents the conclusions and discusses future works resulting from this research.

2. Related Work

2.1. Map Matching Problem

Map matching for low-sampling-rate GPS trajectories is an active research area, and different algorithms have been proposed. Lou et al. [2] proposed the ST-Matching algorithm for low-sampling-rate GPS trajectories, based on spatial-temporal analysis. The algorithm calculates observation probability, based on distance between GPS point and road segment, transition probability based on distance similarity and speed similarity. Then, a sequence of road segments, with the highest score, is identified as the map matching result. However, it is not reliable for filtering out incorrect candidate paths and determining the correct path, based on spatial and temporal information when the

sampling rate exceeds 5 min. Hsueh et al. [3] extended the ST-Matching algorithm to STD-Matching algorithm by considering real-time direction of the GPS reading. The advantage of STD-Matching is that it can effectively filter out road segments with the wrong direction, which improves map matching accuracy, but it still confronts the same issue with STD-Matching when the sampling rate is very low. Quddus et al. [1] proposed a map matching method, based on weight-based shortest path, called stMM. Four different weights are assigned to each of the candidate links, including: (1) weight for perpendicular distance, (2) weight for bearing difference, (3) weight for shortest-path distance and (4) weight for heading difference. Then, the minimum weight path is calculated as the map matching result. Liu et al. [4] proposed a spatial and temporal conditional random field (ST-CRF) algorithm and considered consistency of driving direction. This algorithm calculates emission probability based on distance between GPS point and candidate point, transition probability by five influencing factors: (1) Distance between GPS point and candidate point, (2) shortest path distance and Euclidean distance between two GPS points, (3) average speed and speed constraint of road segment, (4) middle-point spatial distribution, and (5) driving direction. Dynamic programming was used to obtain the maximum probability path. Yin et al. [5] used length of the candidate route and road turning angle to estimate cost of the candidate route. Then, the authors computed the likelihood of a candidate path based on its cost and determined the path with the highest likelihood as the map matching result. In [1,5], they ignored vehicle speed and speed constraint of road segment, which may cause the final selected path to be unreachable under speed constraint. The above algorithms use a variety of features to handle map matching for low-sampling-rate GPS trajectories. Spatiotemporal features, including distance similarity, speed similarity and direction similarity, are used to filter out unreasonable candidate paths, reduce computation time and improve accuracy. However, when the sampling rate becomes very low, the uncertainty between consecutive GPS points becomes much larger, so it is very difficult to determine the correct path relying only on these features.

Some other map matching methods take advantage of historical GPS trajectories with high sampling rates. Kai et al. [7] developed a map matching method, called HRIS (History-based Route Inference System). Historical high-sampling-rate GPS trajectories were first located on a road network to obtain historical routes with existing map matching techniques. Then, the low-sampling-rate trajectories were partitioned into a sequence of consecutive GPS location pairs, searched historical GPS trajectories through location pairs and popular routes were calculated between two locations. Next, HRIS connected consecutive local routes, based on a scoring function, which considered both the popularity of local routes and the confidence for connecting them. Finally, a dynamic programming algorithm was designed to calculate a global route with the highest score. Huang et al. [8] proposed a method to solve map matching for a low-sampling-rate GPS trajectory by mining frequent path patterns from historical high-sampling-rate GPS trajectories. This method performs a conventional map matching algorithm to obtain the paths of historical high-sampling-rate GPS trajectories. Next, the paths (and sub-paths) and their frequencies are stored in an FP (Frequent Pattern)-Forest index. Finally, the maximum probability path is calculated for a low-sampling-rate GPS trajectory based on the discovered frequent paths. The algorithms [7,8] above use additional historical high-sampling-rate GPS trajectory data to obtain frequent paths between any two locations in the road network because third-party historical GPS trajectory data can compensate for the uncertainty between consecutive GPS points with low sampling rates. However, these algorithms are susceptible to the quality of historical trajectory data, and drivers may not follow the frequent paths with rapid change of traffic conditions.

2.2. Trajectory Clustering Problem

Lee et al. [9] proposed a clustering algorithm for GPS trajectories. This algorithm partitioned the trajectory into sub-trajectories based on minimum description length (MDL) and computed perpendicular, parallel and angle distances to measure similarity between sub-trajectories. Smaller distance means higher similarity between sub-trajectories. Finally, it clustered the sub-trajectories based on DBSCAN algorithm [12]. Although the algorithm is efficient, it is only suitable for trajectory data

without road constraints, such as hurricane movement trajectories and animal migration trajectories. Buchin et al. [10] also divided GPS trajectories into sub-trajectories. They measured the similarities based on Fréchet distance [13] and the discrete Fréchet distance between GPS sub-trajectories, respectively. However, computation cost for Fréchet distance is expensive. Additionally, this algorithm was designed for clustering GPS trajectories to detect commuting patterns in urban road networks, and it cannot be used for map matching. Kharrat et al. [11] proposed a trajectory clustering algorithm with road network constraint. This algorithm first computes the number of moving objects transiting from one road section to another, based on historical GPS trajectories. Next, it searches the densest road sections and utilizes them to generate dense paths on the road network. Last, it classifies the trajectories of moving objects into these dense paths. However, the performance of this algorithm relies on the traffic pattern discovered from additional trajectory datasets, and the movement of vehicles may not follow the discovered traffic pattern with the rapid development of road systems. None of these trajectory clustering methods work for map matching based on trajectory collaboration.

Many algorithms have been proposed to measure the similarity between GPS trajectories, which would be useful for trajectory clustering. For example, Marinescu-Istodor and Fränti [14] proposed a grid-based method to compute four different route measures - novelty, noteworthiness, similarity and inclusion. For similarity measurement, they first transformed route into cell representation and then utilized the Jaccard Index to measure the amount of similarity.

3. Methodology

3.1. Preliminaries

- Definition 1. Road network.

A road network is a directed graph, $G = (V, E)$, where V is a set of vertices representing the terminal points of road segments, and E is a set of directed edges representing the road segments. Vertex $v_i \in V$ is a terminal point of road segments. Edge $e_i \in E$ is a road segment with a starting point $e_j.start$ and an end point $e_j.end$, where $e_j.start \in V$ and $e_j.end \in V$.

- Definition 2. Path.

A path is a sequence of consecutive road segments, denoted as $\text{path} = \{e_1, e_2, \dots, e_m\}$, where e_i is a road segment with speed constraint and $e_i.end = e_{i+1}.start (1 \leq i \leq m)$. The first and last vertex of path is denoted as $\text{path}.first$ and $\text{path}.last$, and $\text{path}.first = e_1.start$ and $\text{path}.last = e_m.end$.

Figure 2 gives an example of two paths. In Figure 2, two paths, path_1 and path_2 , can be constructed by three road segments e_1, e_2 and e_3 . To be more specific, $\text{path}_1 = \{e_1, e_2\}$, and $\text{path}_2 = \{e_1, e_3\}$. In Figure 2, $\text{path}_1.first = v_1$ and $\text{path}_1.last = v_4$.

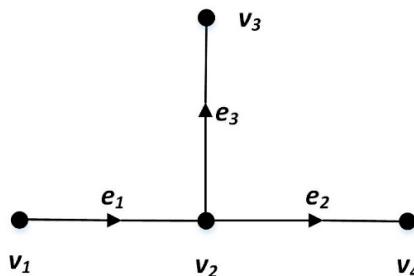


Figure 2. Example of Path.

- Definition 3. GPS point.

A GPS point p is a 4-tuple denoted as: $p = (t, lat, lng, dir)$ where t is the timestamp of the GPS point, and lat, lng and dir are the latitude, longitude and direction of the location of the GPS point at time t .

- Definition 4. GPS trajectory.

A GPS trajectory is a sequence of GPS points $trj = \{p_1, p_2, \dots, p_m\}$ where, $p_i.t - p_{i-1}.t > 0, 1 \leq i \leq m$.

The map matching problem can be represented as follows: given a set of low-sampling-rate GPS trajectories, S , for each GPS trajectory $trj \in S$, this paper aims to locate trj onto the road network so that the path $path$ of trj can be obtained.

3.2. Overview of Collaborative Map Matching

In this paper, a collaborative map matching (CMM) algorithm is proposed to handle low-sampling-rate GPS trajectories. Figure 3 shows the framework of the CMM algorithm, which can be divided into three steps. The first is data preprocessing, where outliers are removed from GPS trajectories. To speed up the query operations on road segments and paths, an R-tree spatial index is built for road segment query, and a Path-Forest index is built to store paths in the road network. In the second step, k shortest distance paths [15] are first calculated out as candidate paths for each GPS trajectory. Then, the similarity between GPS trajectories is measured based on path similarity, and the DBSCAN algorithm is extended to cluster GPS trajectories. Next, a collaborative GPS trajectory with a high sampling rate is generated by resampling of a group of similar low-sampling-rate GPS trajectories in each cluster. In the last step, the LCSS similarity between the generated collaborative GPS trajectory and its candidate paths is calculated, and the path with the largest similarity is taken as the map matching result for all trajectories in the cluster. Each step will be discussed in detail.

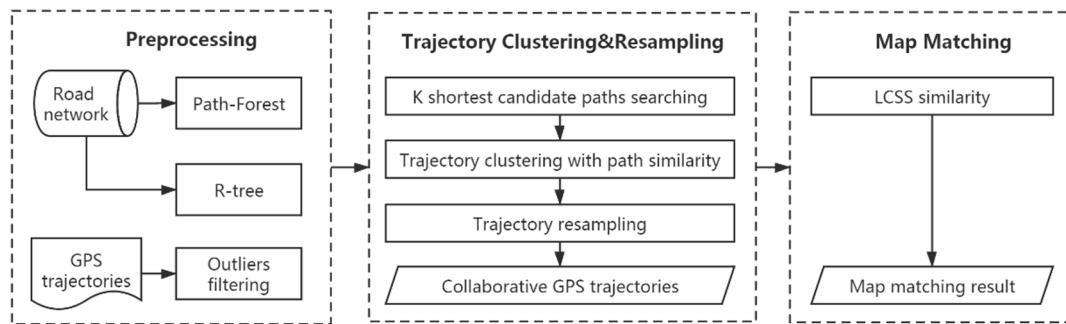


Figure 3. The framework of CMM algorithm.

3.3. Preprocessing

Data preprocessing is an important part of the proposed CMM algorithm. The first task is removing the outlier. If the difference between a GPS point and its nearest road segment exceeds the corresponding thresholds (including distance and direction), the GPS point will be taken as an outlier and removed. Second, an R-tree is built on road segments to facilitate the road segment query through spatial proximity. Finally, a Path-Forest index (Path-Forest) is built to store all paths between two road segments where path length is smaller than threshold l_Δ , which could improve the efficiency of candidate route search in the road network. The Path-Forest is similar to the FP-Forest in [8], the difference being that FP-Forest is designed to store frequent paths (sub-paths) and their frequencies and Path-Forest is utilized to store all paths between any two road edges with length under l_Δ . In a Path-Forest, each road segment in the road network will be taken as the root of a tree and the paths sourcing from the road segment will be stored in the tree. Each tree is associated with a hash table, where the key is the road segment ID and the value is a list storing the sequence of the corresponding road segments in the paths.

It is efficient to retrieve all paths between two road segments in a Path-Forest. Figure 4 shows an example of path query from road segment e_1 to e_6 . First, the tree with root e_1 and its associated hash table is searched out. In the hash table, the list pointing to e_6 will be identified. For each element in the

list, the path from e_1 to e_6 will be obtained by tracing back, starting from e_6 in the tree. Thus, two paths from e_1 to e_6 will be searched out as $\{e_1, e_2, e_4, e_6\}$ and $\{e_1, e_3, e_5, e_6\}$.

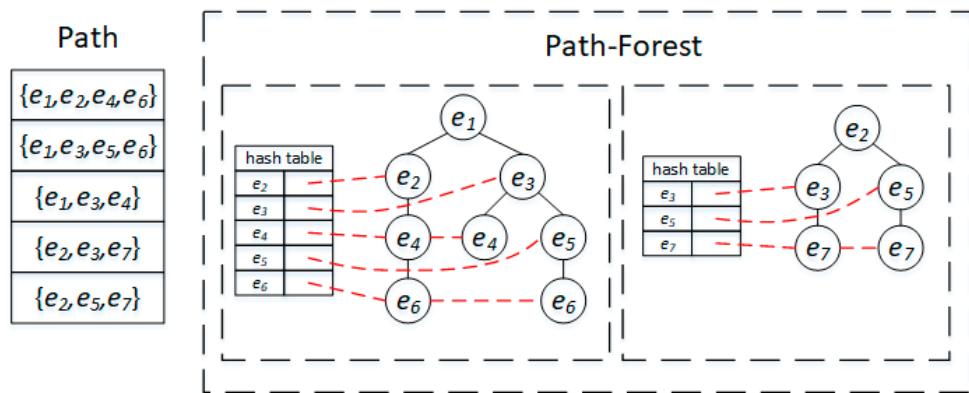


Figure 4. The Path-Forest.

3.4. Trajectory Clustering with Path Similarity

In this section, DBSCAN algorithm with path similarity is utilized for clustering low-sampling-rate GPS trajectories. The candidate paths are searched out for each low-sampling-rate GPS trajectory, and the similarity between GPS trajectories is measured based on the candidate paths.

3.4.1. Candidate Path Search

Given a search radius r , the candidate road segments for each GPS point can be queried out from an R-tree, and the paths to connect the candidate road segments of consecutive GPS points can be searched out from the Path-Forest. Unreasonable candidate paths should be pruned. In this paper, the candidate path pruning is implemented, based on a heading threshold β and an estimated temporal threshold, σ . The candidate path search with pruning method can be divided into two parts: candidate road segment query for each GPS point and path search in the Path-Forest with speed constraint.

1. Candidate Road Segment Query. When querying the surrounding road segments of a GPS point with radius, r (set as 100 m) in the R-tree, if the heading difference between a candidate road segment and the direction of the GPS point exceeds threshold β (set as 60°), the road segment will be dismissed from the query. Thus, candidate road segments with a similar direction for the GPS points could be searched.
2. Path Search. A complete candidate path of a GPS trajectory connects the candidate road segments from the first GPS point to the last GPS point, so it must search paths by connecting the candidate road segments of consecutive GPS points. However, the number of candidate paths for a GPS trajectory would be substantial in a road network. Thus, time constraint σ is calculated and used to filter out unreasonable paths and facilitate path search,

$$\sigma = \sum_{k=1}^m \frac{e_k.length}{e_k.speedlimit} \quad (1)$$

where σ is the calculated time threshold from GPS point p_{i-1} to the next point p_i , and e_k is a road segment on the candidate path from p_{i-1} to p_i . Additionally, $e_k.speedlimit$ is the maximum speed allowed on e_k . Hence, threshold σ stands for the minimum time required to travel from p_{i-1} to p_i . If the time interval $t_{i-1 \rightarrow i}$ between p_{i-1} and p_i in Equation (2) is smaller than threshold σ , the path is considered unreasonable and is pruned.

$$t_{i-1 \rightarrow i} = p_i.t - p_{i-1}.t \quad (2)$$

Given a GPS trajectory $trj = \{p_1, p_2, \dots, p_m\}$, the pseudo-code of CPS is shown in Algorithm 1. The algorithm can be divided into two parts. The first part is candidate road segment query with radius r for each GPS point in lines 1–3. The set CE_i is the set of candidate road segments of the GPS point p_i in GPS trajectory trj . The second part is path search by connecting candidate road segments in lines 4–15. $Path$ is the set of candidate road segment sequences for trj . As the algorithm starts from the candidate road segments of the first GPS point, in line 4, each candidate road segment in CE_1 is taken as a *path* and added into $Path$. From lines 5–15, the algorithm utilizes the Path-Forest to search candidate paths $paths$ from p_{i-1} to p_i , removing unreasonable paths through pruning and adding them into the candidate path set $Path'$. Last, the previous candidate paths in $Path$ are connected to the current paths in $Path'$, and $Path$ is updated with the result of connection.

Algorithm 1: Candidate Path Search (CPS)

Input: GPS trajectory $trj = \{p_1, p_2, \dots, p_m\}$; the set of edges in road network E ; search radius r ; heading change threshold β ;

Output: a set of $Path = \{path_1, path_2, \dots, path_n\}$

1. **for** ($i = 1; i \leq m; i++$)
2. $CE_i \leftarrow \text{CandidateRoadSegmentQuery}(p_i, E, r, \beta)$
3. **end for**
4. $Path \leftarrow CE_1$
5. **for** ($i = 2; i \leq m; i++$)
6. $Path' \leftarrow \emptyset$;
7. **foreach** (p in $Path$)
8. **foreach** (ce in CE_i)
9. $\sigma \leftarrow \sum_{k=1}^m e_k.length / e_k.speedlimit$
10. $paths \leftarrow \text{PathForestQuery}(p.lastedge, ce, \sigma);$
11. $Path' \leftarrow paths;$
12. **end foreach**
13. **end foreach**
14. $Path \leftarrow \text{PathConnect}(Path, Path');$;
15. **end for**

3.4.2. Trajectory Clustering

In order to supplement the missing information of low-sampling-rate GPS trajectories, similar GPS trajectories are clustered together. In this paper, the DBSCAN clustering method is extended for low-sampling-rate GPS trajectory clustering, based on path similarity instead of geometric similarity, as geometric similarity does not perform well for similarity measurements between low-sampling-rate GPS trajectories. The path similarity method measures similarity between low-sampling-rate GPS trajectories by comparing their candidate paths based on the Jaccard dissimilarity index. As the number of potential candidate paths for a GPS trajectory would be overwhelming, the k candidate paths with the shortest distance for each GPS trajectory are selected as candidate paths for path similarity measurement.

The Jaccard dissimilarity index $d(trj_1, trj_2)$ between two GPS trajectories trj_1 and trj_2 is calculated based on the ratio of candidate paths whose dissimilarity is smaller than ε_p to the total number of path combinations between two candidate sets. The calculation of $d(trj_1, trj_2)$ is shown in Equation (3).

$$d(trj_1, trj_2) = 1 - \frac{|\pi(trj_1) \cap \pi(trj_2)|}{|\pi(trj_1) \cup \pi(trj_2)|} \quad (3)$$

$$|\pi(trj_1) \cap \pi(trj_2)| = \{d(path_i, path_j) | \varepsilon_p | path_i \in \pi(trj_1), path_j \in \pi(trj_2)\}.count$$

$$|\pi(trj_1) \cup \pi(trj_2)| = \pi(trj_1).count * \pi(trj_2).count$$

where $\pi(trj)$ denotes the set of candidate paths of the GPS trajectory trj , $|\pi(trj_1) \cap \pi(trj_2)|$ is the intersection of the candidate path sets of two trajectories and is composed of paths whose dissimilarity is smaller than ε_p . $|\pi(trj_1) \cup \pi(trj_2)|$ is the union of the two candidate path sets and is composed of all combinations between the candidate paths of the two trajectories.

The dissimilarity between two paths $path_i$ and $path_j$ is defined in Equation (4).

$$d(path_i, path_j) = 1 - \frac{2 \times sim1 \times sim2}{sim1 + sim2} \quad (4)$$

$$sim1 = \frac{LCSS(path_i, path_j).count}{path_i.count}, sim2 = \frac{LCSS(path_i, path_j).count}{path_j.count}$$

where $LCSS(path_1, path_2)$ is the longest common subsequence [16] between two paths, and it calculates the number of common road segments in the two paths. For example, given two paths, $path_1 = \{e_1, e_2, e_3, e_4, e_5\}$ and $path_2 = \{e_1, e_3, e_4, e_6\}$, the value $LCSS(path_1, path_2) = \{e_1, e_3, e_4\}$, $path_1.count = 5$, $path_2.count = 4$.

DBSCAN identifies clusters for spatial data based on density and requires two parameters: ε describing the maximum distance of a neighborhood, and $MinPts$ describing the minimum number of points required to form a dense neighborhood. In this paper, the parameter ε is considered in two aspects, the maximum distance difference ε_l between endpoints of trajectories, and the maximum dissimilarity ε_s between two GPS trajectories based on path similarity. Thus, if the OD (origin-destination) pairs of GPS trajectories are distant from each other, they will be aggregated into different clusters.

The neighboring trajectories of each low-sampling-rate GPS trajectory could be obtained by Algorithm 2. As R-tree and Path-Forest have been built for GPS trajectories, it is efficient to retrieve the candidate paths around this trajectory (line 2). The dissimilarity measurement will filter out the GPS trajectories in which distance between endpoints is larger than threshold ε_l , and dissimilarity between two GPS trajectories is larger than threshold ε_s .

Algorithm 2: Neighbor Query in DBSCAN

Input: a set of GPS trajectory $trj = \{p_1, p_2, \dots, p_m\}$, parameters $\varepsilon_l, \varepsilon_s$
Output: a set of $trjs = \{trj_1, trj_2, \dots, trj_n\}$

1. $trjs \leftarrow \emptyset;$
2. $trjs_{nei} \leftarrow rtree.query(trj.mbr);$
3. $curPaths \leftarrow trj.candidate;$
4. **foreach** (t in $trjs_{nei}$)
 5. $neiPaths \leftarrow t.candidate;$
 6. **if** ($dis(t.p_1, trj.p_1) < \varepsilon_l$ and $dis(t.p_n, trj.p_m) < \varepsilon_l$)
 7. $d_n \leftarrow \text{JaccardDissimilarity}(curPaths, neiPaths, \varepsilon_s)$
 8. **if** ($d_n < \varepsilon_s$)
 9. $trjs \leftarrow t$
 10. **end if**
 11. **end if**
 12. **end foreach**

Therefore, GPS trajectories can be clustered with the DBSCAN method with path similarity. Given certain values of $MinTrjs$, ε_l , ε_s , a GPS trajectory will be considered as a core trajectory if the number of GPS trajectories in its neighborhood exceeds the parameter $MinTrjs$. Then, a cluster can be generated based on the core trajectory with the conventional DBSCAN algorithm. If a GPS trajectory can be aggregated into more than one cluster, it will be assigned to the cluster with the largest number of GPS trajectories.

Figure 5 shows an example of trajectory clustering for four GPS trajectories (denoted in black and blue). The GPS trajectories Trj_1, Trj_2, Trj_4 are moving on the same path (black arrow), and the GPS

trajectory Trj_3 is moving on another path (blue arrow). With $\varepsilon_s = 0.3$ and $MinTrjs = 2$ in the DBSCAN algorithm, the final clustering result for the four GPS trajectories is $Cluster_1 = \{Trj_1, Trj_3, Trj_4\}$, $Noise = \{Trj_2\}$. Tables 1–3 show the road segment sequence of each path, the candidate paths of each GPS trajectory, and the dissimilarity between GPS trajectories in Figure 5, respectively.

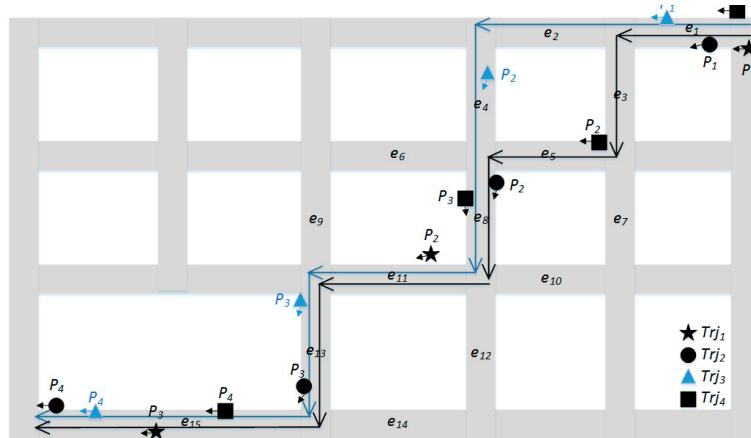


Figure 5. Clustering for low-sampling-rate GPS trajectories.

Table 1. Paths and the corresponding road segment sequences.

| Path ID | Road Segment Sequence in the Path |
|-------------------|---|
| path ₁ | {e ₁ , e ₂ , e ₄ , e ₈ , e ₁₁ , e ₁₃ , e ₁₅ } |
| path ₂ | {e ₁ , e ₃ , e ₅ , e ₈ , e ₁₁ , e ₁₃ , e ₁₅ } |
| path ₃ | {e ₁ , e ₃ , e ₇ , e ₁₀ , e ₁₁ , e ₁₃ , e ₁₅ } |
| path ₄ | {e ₁ , e ₂ , e ₄ , e ₆ , e ₉ , e ₁₃ , e ₁₅ } |
| path ₅ | {e ₁ , e ₃ , e ₅ , e ₈ , e ₁₂ , e ₁₄ , e ₁₅ } |

Table 2. Trajectories and the corresponding candidate paths.

| Trajectory ID | Candidate Paths |
|------------------|---|
| Trj ₁ | path ₁ , path ₂ , path ₃ |
| Trj ₂ | path ₁ , path ₂ |
| Trj ₃ | path ₁ , path ₄ |
| Trj ₄ | path ₂ , path ₅ |

Table 3. Dissimilarity between trajectories.

| $d(Trj_i, Trj_j)$ | Trj_1 | Trj_2 | Trj_3 | Trj_4 |
|-------------------|---------|---------|---------|---------|
| Trj_1 | - | 0.16 | 0.5 | 0.33 |
| Trj_2 | 0.16 | - | 0.25 | 0.25 |
| Trj_3 | 0.5 | 0.25 | - | 0.75 |
| Trj_4 | 0.33 | 0.25 | 0.75 | - |

3.5. Trajectory Resampling

GPS trajectories are aggregated into several groups with the DBSCAN clustering method. However, GPS trajectories in a cluster still confront the uncertainty issue due to low sampling rates. Therefore, for each cluster, it is necessary to integrate the low-sampling-rate GPS trajectories into a collaborative GPS trajectory, so they can supplement the missing information for each other. For example, the three GPS trajectories Trj_1, Trj_2, Trj_4 can supplement the missing information for each other in Figure 5. The generated collaborative GPS trajectory should integrate spatial information of low-sampling-rate GPS trajectories in the cluster for map matching. Thus, the trajectory resampling is different from

averaging trajectory segment problems, which intends to obtain a representative trajectory for maintaining shape characteristics [17].

In this paper, a sliding window algorithm for trajectory resampling is proposed to generate collaborative GPS points from each trajectory cluster. The pseudocode of the algorithm is shown in Algorithm 3. At the beginning, the first and end points of all GPS trajectories in the cluster are collected into two sets of points, *firstPs* and *endPs* (line 2). Then, a GPS point is randomly selected from *firstPs* as the seed point, denoted as *pSeed* (line 3). A window composed of two circles, C_1 and C_2 , centering at *pSeed* with radius r_s and $2 * r_s$ respectively, is generated (line 5). In the window, the GPS points inside C_1 can be retrieved quickly, and the center of the obtained GPS points is calculated as a new resampling GPS point, denoted as *pNew* (line 6–7). Next, a GPS point between C_1 and C_2 is randomly selected as a new seed point, *pSeed*. If there is no GPS point between C_1 and C_2 , the radius r_s is expanded to build a larger circle C_2 until a GPS point is obtained as the new seed point (line 8–9). Last, the window will slide forward until the end points in *endPs* are reached, and the sequence of generated sampling points can make up a collaborative GPS trajectory from the low-sampling-rate GPS trajectories in the cluster (line 4–11).

Algorithm 3: Sliding-Window Algorithm for Trajectory Resampling

Input: a set of GPS trajectories $trjs = \{trj_1, trj_2, \dots, trj_n\}$, search radius r_s

Output: a collaborative GPS trajectory $resultTrj = \{p_1, p_2, \dots, p_m\}$

1. $resultTrj \leftarrow \emptyset;$
 2. $firstPs \leftarrow trjs.FirstPs, endPs \leftarrow trjs.LastPs;$
 3. $pSeed \leftarrow random(firstPs);$
 4. **while** ($pC_1C_2.contains(endPs)$)
 5. create C_1, C_2 with radius r_s and $2 * r_s$;
 6. $pC_1 \leftarrow rtree.query(pSeed, C_1);$
 7. $pNew \leftarrow mean(pC_1);$
 8. $pC_1C_2 \leftarrow rtree.query(pSeed, C_1, C_2);$
 9. $pSeed \leftarrow random(pC_1C_2);$
 10. $resultTrj \leftarrow pNew;$
 11. **end while**
-

Figure 6 shows the procedure of trajectory resampling in a cluster. In this figure, five trajectories (green, red, blue, purple and yellow) are grouped into a cluster; the small black arrow is the real-time heading of a GPS point. In Figure 6a, the red GPS point is randomly selected as the seed point, and two circles C_1 and C_2 are generated, centering as the seed point. In C_1 , five GPS points are retrieved, and the center point is created as a new resampling point and denoted as P_1 in Figure 6b. In Figure 6a, two GPS points exist between C_1 and C_2 , and the next seed point will be randomly selected from them. As the window reaches the end GPS points of the cluster, a collaborative GPS trajectory is generated and composed from P_1 to P_7 , as shown in Figure 6c.

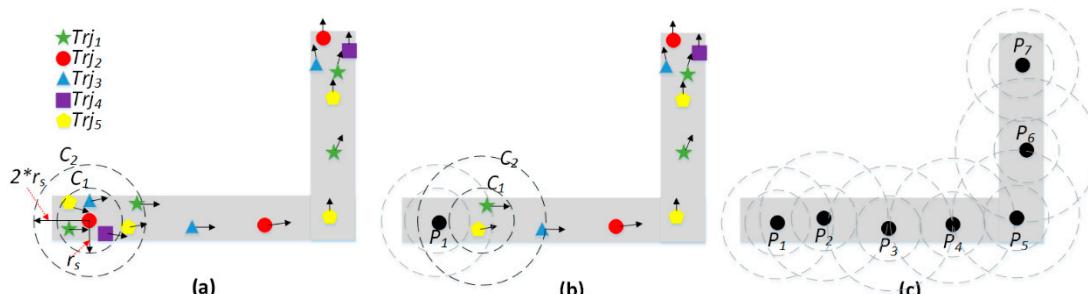


Figure 6. Trajectory cluster resampling. (a) Seed selection and window generation. (b) Point resampling and window slide. (c) Collaborative trajectory generation.

3.6. Map Matching for the Supplemented Trajectory

After trajectory resampling, a collaborative GPS trajectory is generated from each cluster, and it will be matched to one of the candidate paths. Given a GPS trajectory $trj = \{p_1, p_2, \dots, p_m\}$ and a candidate path $path = \{e_1, e_2, \dots, e_n\}$, LCSS [18] is utilized to measure the similarity between trajectory and path. The path with the largest LCSS value will be taken as the map matching result for all low-sampling-rate GPS trajectories in the corresponding cluster:

$$\text{Sim}(p_i, e_j) = \begin{cases} 0, & \text{Dist}(p_i, e_j) > \varepsilon_d \\ 1 - \frac{\text{Dist}(p_i, e_j)}{\varepsilon_d}, & \text{otherwise} \end{cases}. \quad (5)$$

$\text{Dist}(p_i, e_j)$ is the Euclidean distance from a GPS point p_i to the closest point on the road segment e_j . If $\text{Dist}(p_i, e_j) > \varepsilon_d$, this GPS point p_i cannot be matched to this road segment e_j . The subsequence of a trajectory trj from its first GPS point to the i -th GPS point is defined as $trj(i) = \{p_1, p_2, \dots, p_i\}$, $1 \leq i \leq m$ and the subsequence of $path$ from the first road segment to the j -th road segment as $path(j) = \{e_1, e_2, \dots, e_j\}$, $1 \leq j \leq n$. The LCSS value, denoted as $L(trj(i), path(j))$, between $trj(i)$ and $path(j)$ is defined as,

$$L(trj(i), path(j)) = \max\{ L(trj(i), path(j-1)), L(trj(i-1), path(j)) + \text{Sim}(p_i, e_j) \} \quad (6)$$

With Equation (6), a dynamic programming algorithm could be used to calculate the LCSS distance between trj and $path$, and the candidate path with the largest LCSS value is selected as the map matching result.

4. Experiment

4.1. Study Area

In this section, the performance of the proposed CMM method was evaluated using real and simulated GPS trajectory datasets in a study area (22.532° N– 22.576° N and 114.052° E– 114.103° E) of Shenzhen, China. The method was implemented in C#. The experiments were conducted on a 3.6 GHz Core i7 PC with 16GB of RAM. The road network in the selected region contains 7007 road vertices and 7949 road segments, which is shown in Figure 7.

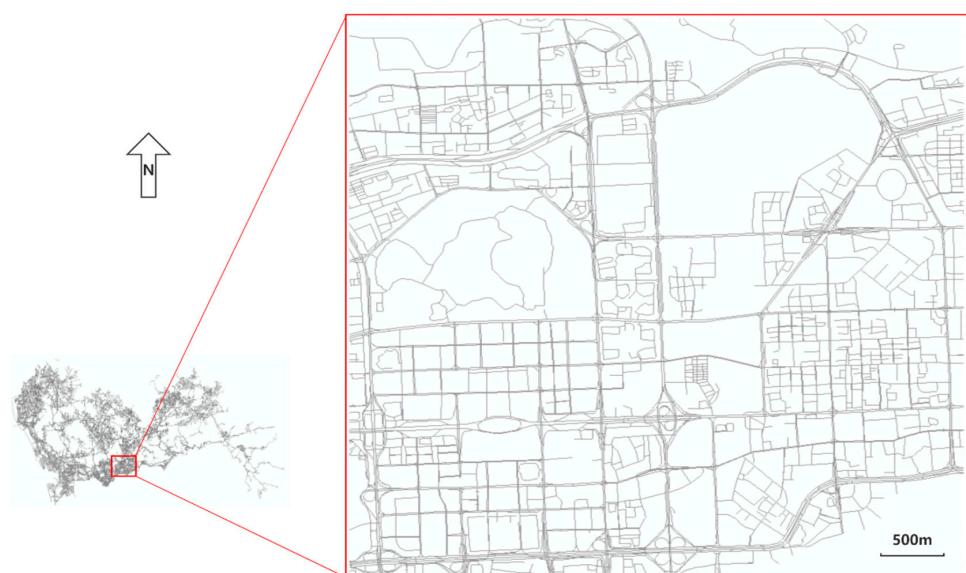


Figure 7. Study area of Shenzhen map.

4.2. Dataset

In this paper, experiments were conducted to evaluate the performance of the proposed CMM algorithm on both, a real and a simulated GPS trajectory dataset.

4.2.1. Real GPS Trajectory Dataset

The real taxi GPS trajectory dataset [19], containing more than 46 million GPS points, was collected in Shenzhen, China. The format of GPS points, includes the taxi ID, longitude, latitude, timestamp, speed and occupancy status. The sampling rate of the GPS trajectories is approximately 15 s. The trajectory data was displayed on the map, and 100 GPS trajectories were selected for validation. The 100 GPS trajectories were selected by visual inspection so that they are diverse in length and located at different locations in the study area. The 100 GPS trajectories are manually labelled for ground truth. To obtain low-sampling-rate GPS trajectories, each GPS trajectory is sampled multiple times with the sampling rate of 3 min and different starting points. In total, 734 GPS trajectories with sampling rate around 3 min were generated. The instantaneous direction of GPS points of the generated trajectory is also manually labelled, as the direction information is unavailable in this real GPS trajectory dataset.

Figure 8a shows the length of the majority of GPS trajectories in the real dataset is between 1 km and 4 km, Figure 8b shows that the number of GPS points in each trajectory is approximately 3 to 5.

4.2.2. Trajectory Simulation

In this paper, GPS trajectories were simulated to investigate the performance of map matching methods in two aspects. First, the performance of map matching methods versus different sampling rates (from 2 min to 10 min) was studied, based on simulated trajectory data, which is difficult to obtain from a real GPS trajectory dataset. Second, GPS trajectories were simulated to study the performance of map matching methods under three different scenarios T_1 , T_2 and T_3 . T_1 represents GPS trajectories moving along a path with multiple turns. T_2 represents GPS trajectories moving along a path with a large detour. T_3 represents GPS trajectories moving along two separate paths which vary slightly at a short sub-path. Figure 8 illustrates the road paths where GPS trajectories are simulated. Figure 9a,b represent scenario T_1 ; Figure 9c represents scenario T_2 ; and Figure 9d,e represent scenario T_3 . In Figure 9f, 50 GPS trajectories were simulated on each road path, and the simulated GPS trajectories are shown in different colors.

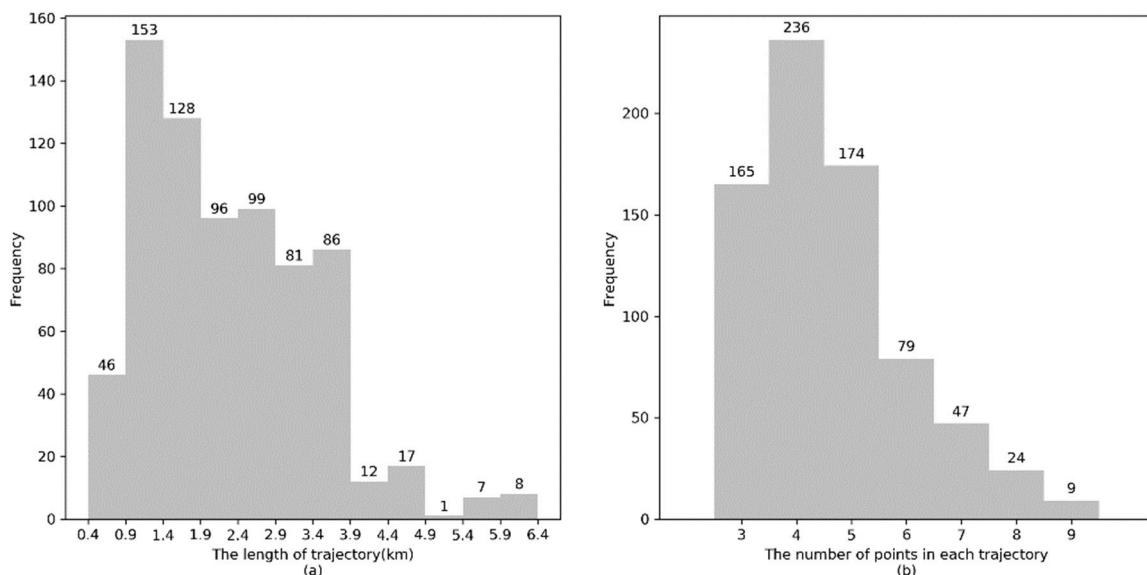


Figure 8. Distribution of real GPS trajectories. (a) Trajectory length. (b) Number of points in each trajectory.



Figure 9. Spatial distribution of selected road paths and simulated GPS trajectories. (a,b) Road paths under scenario T₁. (c) Road path under scenario T₂. (d,e) Road paths under scenario T₃. (f) Spatial distribution of simulated GPS trajectories.

Simulated GPS points are associated with a set of attributes including trip ID, time, longitude, latitude, direction. To simulate a GPS trajectory on path $path = \{e_1, e_2, \dots, e_m\}$, the velocity of the vehicle moving on the path is first estimated using Equation (7):

$$\alpha = \sum_{k=1}^m \left(e_k.speedlimit * \frac{e_k.length}{path.length} \right). \quad (7)$$

The weighted average maximum speed limit α (m/s) on the path is calculated as the velocity of the simulated GPS trajectory. $e_k.speedlimit$ and $e_k.length$ are the speed limit and the length of the road segment e_k , respectively. $path.length$ is the total length of the path. The procedure of GPS trajectory simulation is as follows:

1. Simulate the first point p_1 based on the starting node of the path, and is denoted as p_{prev} .
2. Estimate the location of the next GPS point p_{next} . First, a random velocity $v = random(0, \alpha)$ is generated for the vehicle. Given the sampling rate, the travelled distance from p_{prev} to p_{next} along the path can be calculated as $v*t$. Thus, the location of p_{next} on the path can be obtained and denoted as $(p_{next.x}, p_{next.y})$. Location bias $x_{bias} \sim N(0, 40)$ and $y_{bias} \sim N(0, 40)$ is added to simulate the position deviation of the vehicle away from the road. Thus, the location of p_{next} is simulated as $(p_{next.x} + x_{bias}, p_{next.y} + y_{bias})$.
3. Simulate the direction of p_{next} . With the location of p_{next} , the road segment where p_{next} is located is identified and the direction of the road segment is denoted as $direction_{road}$. Then, a random $direction_{bias} = random(-30, 30)$ is generated as the heading bias of p_{next} . Thus, the direction of p_{next} can be simulated as $direction_{road} + direction_{bias}$.
4. Mark p_{next} as p_{prev} , and repeat steps 1–3 to simulate the next GPS point until the vehicle reaches the end node along the path. Set a unique trajectory ID for the simulated GPS trajectory.

The length distribution of simulated GPS trajectories and the number of GPS points in each simulated GPS trajectory are illustrated in Figure 10a, and Figure 10b, respectively. It can be observed that the length of the majority of simulated GPS trajectories is approximately 4.5 km and the number of GPS points in a trajectory is approximately 3 to 5.

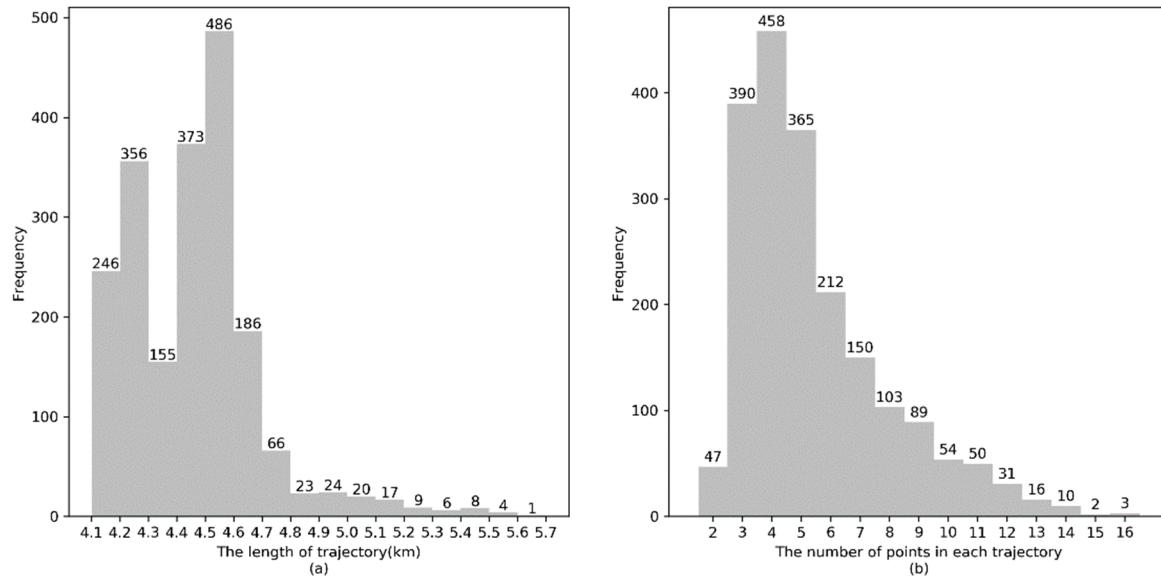


Figure 10. Distribution of simulated GPS trajectories. (a) Trajectory length. (b) Number of points in each trajectory.

4.3. Parameter Configuration

In this section, the performance of CMM is compared with two baseline methods, ST-Matching [2] and STD-Matching [3]. The preprocessing step is the same for all three methods, including outlier removal, the construction of R-tree and Path-Forest. The configuration of CMM is set in Table 4. The experiments are conducted based on the configuration without specific statement.

Table 4. Configuration of CMM.

| Parameters | | |
|--|-----------------|-------|
| Path distance threshold in Path-Forest | l_Δ | 6 km |
| k candidate path for path similarity measurement | k | 3 |
| Distance threshold in trajectory clustering | ε_l | 50 m |
| Dissimilarity threshold in trajectory clustering | ε_s | 0.8 |
| MinTrjs in trajectory clustering | MinTrjs | 5 |
| Trajectory-Resampling radius | r_s | 50 m |
| LCSS distance threshold | ε_d | 100 m |

Two metrics, precision by length of road segments and recall by length of road segments, are used to evaluate the effectiveness of the map matching methods, and they are defined as follows:

$$\text{Precision} = \frac{\text{Length of correctly matched road segments}}{\text{Length of predicted path}}$$

$$\text{Recall} = \frac{\text{Length of correctly matched road segments}}{\text{Length of ground true path}}$$

4.4. Performance Evaluation on Real Trajectory Dataset

In this paper, the performance of the proposed CMM method and two baseline methods was investigated using a real GPS trajectory dataset. First, the effectiveness and efficiency of the three methods were compared in terms of precision, recall and running time. Second, parameter tuning was conducted to investigate the impact of two significant parameters on the CMM method. Third, the effectiveness of path similarity was studied and compared with the LCSS and Edit Distance on Real Sequence (EDR) methods.

4.4.1. Overall Performance Evaluation

This section investigates precision, recall and running time of the CMM method, ST-Matching method and STD-Matching method. The average running time per GPS point of the three methods are shown in Table 5. It can be observed that the average running time per GPS point of CMM is approximately 2.97 milliseconds (ms), while the average running time per GPS point for ST-Matching and STD-Matching method are 29.60 ms, and 28.37 ms, respectively. CMM is much more efficient than ST-Matching and STD-Matching because once the trajectories are clustered in the same cluster, CMM conducts map matching for all GPS trajectories belonging to the cluster, while the other two methods process the GPS trajectory individually.

Table 5. Average Running Time per GPS Point on Real GPS Trajectory Dataset.

| Method | Time (ms) |
|--------------|-------------|
| ST-Matching | 29.60 |
| STD-Matching | 28.37 |
| CMM | 2.97 |

The precision and recall of the three map matching methods are shown in Table 6. The precision and recall of CMM are 0.865 and 0.872 for real GPS trajectories with the sampling rate of around 3 min, which performs better than both ST-Matching and STD-Matching. ST-Matching has the smallest precision and recall because it could not distinguish the correct path effectively based on the spatial and temporal features. STD-Matching involves direction information and has a better performance than ST-Matching. The CMM method groups similar GPS trajectories together and solves the uncertainty issue by generating a new collaborative trajectory with information supplementation of the low-sampling-rate GPS trajectories, which demonstrates that CMM has the best performance out of the three methods in terms of precision and recall.

Table 6. Precision and Recall on Real GPS Trajectory Dataset.

| Method | Precision | Recall |
|--------------|--------------|--------------|
| ST-Matching | 0.817 | 0.721 |
| STD-Matching | 0.855 | 0.792 |
| CMM | 0.865 | 0.872 |

4.4.2. Performance Evaluation for Path Similarity

This section evaluates the performance of the proposed path similarity method. Understanding how the similarity measure is affected by sampling rates was investigated. The similarity between the trajectory with varied sampling rates and the origin trajectory is expected to be 100%.

The performance of the path similarity method was compared with two widely used methods, Longest Common Subsequence (LCSS) [16] and Edit Distance on Real Sequence [20] methods. LCSS and EDR are count-based methods. LCSS counts the number of matched pairs, and EDR counts the cost of operations needed to fix the unmatched pairs. The sampling rates could have a great impact on the distance calculation between the two trajectories, especially the GPS trajectories with a

low sampling rate. However, the count-based similarity measurement methods are less sensitive to sampling rates. To be more specific, regardless of distance between the two consecutive GPS points, count-based similarity measurement methods do not change the value. The experiment results are shown in Figure 11.

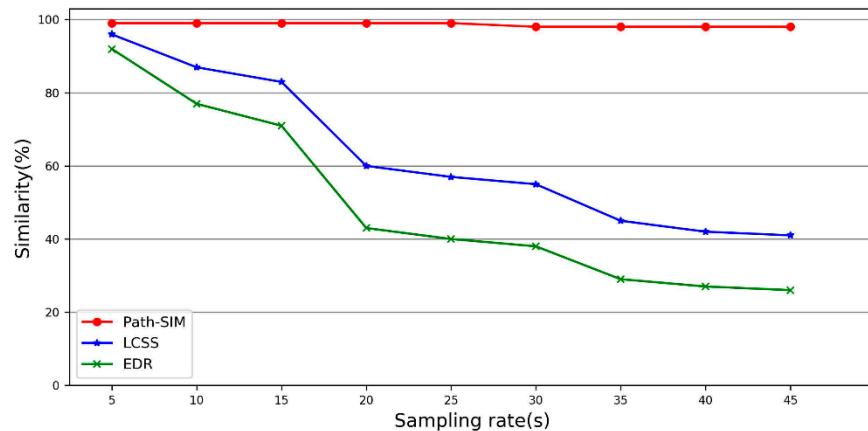


Figure 11. Performance evaluation for path similarity (Path-SIM), LCSS and EDR.

It can be observed in Figure 11 that path similarity method (Path-SIM) has better performance than LCSS and EDR methods. Path-SIM is almost not affected by sampling rates because the candidate paths for a trajectory have a small change with the varied sampling rates. However, LCSS and EDR are more sensitive to the change of sampling rates compared to Path-SIM.

4.4.3. Parameter Tuning

CMM has two important parameters - the k value for the top k shortest distance paths, and the dissimilarity threshold ε_s for defining neighborhood in the extended DBSCAN. This section investigates the impact of two parameters, k and ε_s on the performance of CMM, based on real GPS trajectories. In this experiment, k is set from 1 to 15, and ε_s is 0.1 to 0.8.

Figure 12a,b display the precision and recall of map matching with different k and ε_s values. First, when ε_s is fixed, the precision and recall increase as k grows from 1 to 3. When k is set as 1, similar GPS trajectories cannot be well clustered together using path similarity because the shortest distance path is usually not the real path for some GPS trajectories. Thus, GPS trajectories in a cluster may not be enough to supplement information for each other, which results in precision and recall being smaller than when k is set as 3. With k increasing from 3 to 15, the precision decreases gradually, while the recall stays steady. This arises because a greater k will result in a larger number of unreasonable candidate paths with detours, which would compromise the precision of CMM but has a small impact on the recall of CMM. Second, when k is fixed, the precision and recall increase with the growth of ε_s because more similar GPS trajectories can be aggregated with a larger ε_s , which leads to more spatial information of similar GPS trajectories being incorporated together for map matching.

Figure 12c shows the running time of CMM rises with the growth of k and declines with the growth of ε_s . With a greater k , more candidate paths are searched out for GPS trajectories, which increases the computation time for trajectory similarity measurements in trajectory clustering. A smaller ε_s value causes a smaller number of GPS trajectories in one cluster and increases the total number of clusters. Therefore, CMM requires more running time to conduct map matching, as shown in the figure.

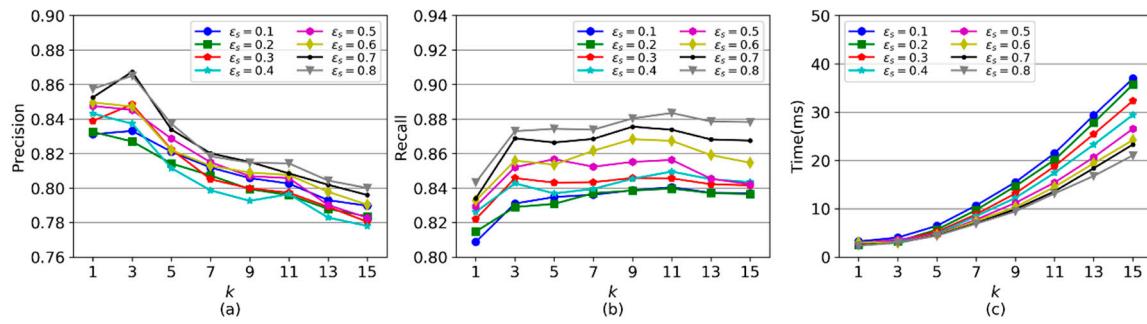


Figure 12. The performance of CMM with different values of k and ε_s . (a) Precision. (b) Recall. (c) Running time.

4.5. Performance Evaluation on Simulated Trajectory Dataset

In this paper, the performances of CMM, ST-Matching and STD-Matching were also investigated on a simulated GPS trajectory dataset. First, the impact of sampling rate on the performance of the three map matching methods was investigated. Second, the three map matching methods were studied under different scenarios. Third, the impact of trajectory data size on the performance of the three map matching methods was studied.

4.5.1. Performance versus Sampling Rate

Figure 13 shows the performance of CMM, ST-Matching and STD-Matching, with different sampling rates, on simulated GPS trajectories. It can be observed that CMM always performs better than ST-Matching and STD-Matching when GPS sampling rate varies from 2 min to 10 min. Compared to ST-Matching and STD-Matching, the precision and recall of CMM are much less sensitive to the sampling rates. When the sampling rate is very low, e.g., 8 min, consecutive GPS points become more distant from each other, resulting in large uncertainty for the real path of the trajectory. Thus, the precision and recall of ST-Matching and STD-Matching drop significantly based on spatial and temporal features. CMM supplements the spatial information of low-sampling-rate GPS trajectories with trajectory collaboration, which can alleviate the larger uncertainty issue caused by lower sampling rates.

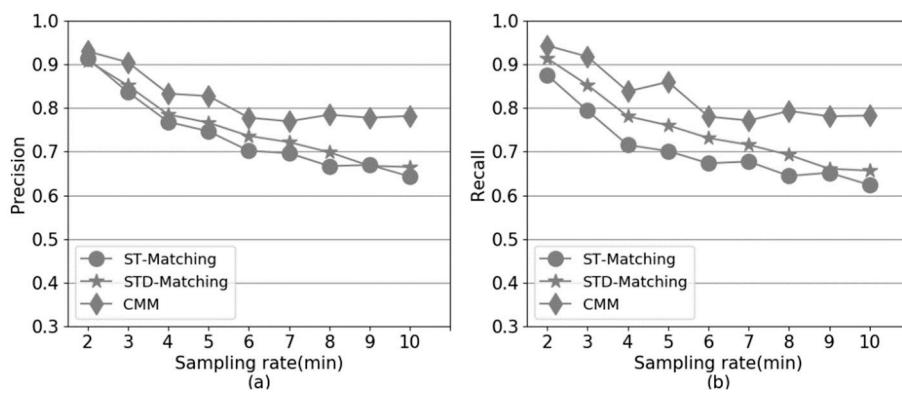


Figure 13. The performance comparison versus different sampling rates. (a) Precision. (b) Recall.

4.5.2. Performance Evaluation under Different Scenarios

The performance of the three map matching methods was investigated under three scenarios T_1 , T_2 and T_3 . The experiment results are displayed in Figure 14. It can be observed in Figure 14a–d that CMM performs better than ST-Matching and STD-Matching under scenarios T_1 (multiple turns) and T_2 (detour). For scenario T_3 , the precision and recall of CMM, ST-Matching and STD-Matching are all over

0.8. It can be concluded that CMM can effectively aggregate low-sampling-rate GPS trajectories into different clusters based on path similarity, if two groups of trajectories move along distinct road paths.

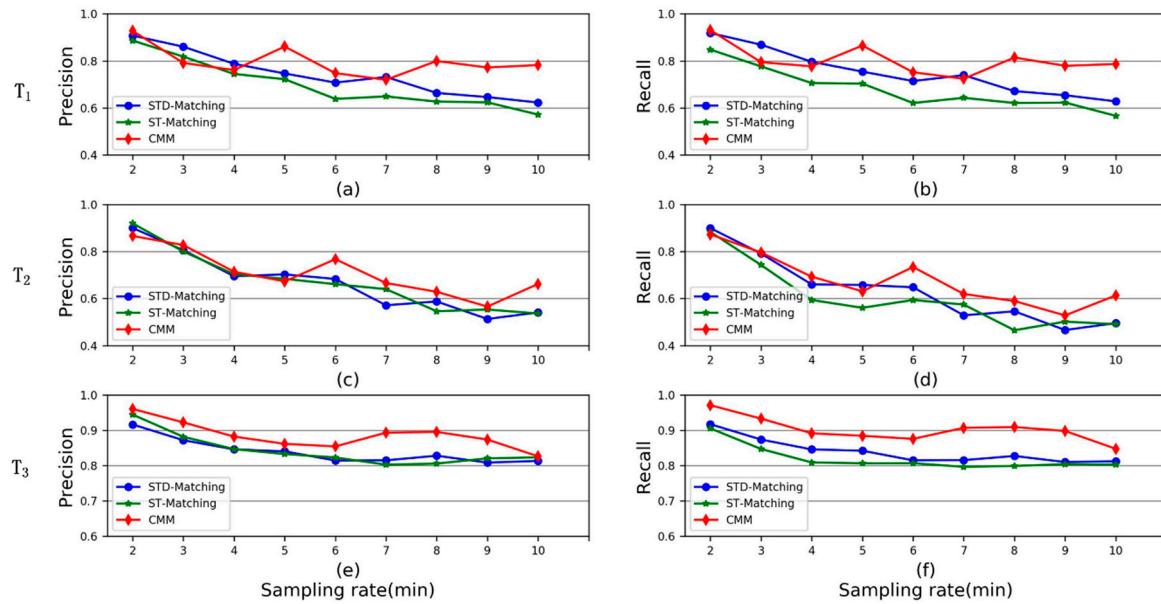


Figure 14. The performance comparison of CMM, ST-Matching and STD-Matching under different scenarios. (a,b) Scenario T₁. (c,d) Scenario T₂. (e,f) Scenario T₃.

Figure 15 visualizes the map matching results for a simulated GPS trajectory under T₂. Figure 15a shows that the sampling rate of the simulated GPS trajectory is very low, and it is very difficult to identify the real path of the trajectory. Figure 15b shows the cluster where the low-sampling-rate GPS trajectory is aggregated with the path similarity method. Figure 15c represents the collaborative trajectory of the cluster after resampling, and Figure 15d shows the map matching result of the collaborative trajectory.

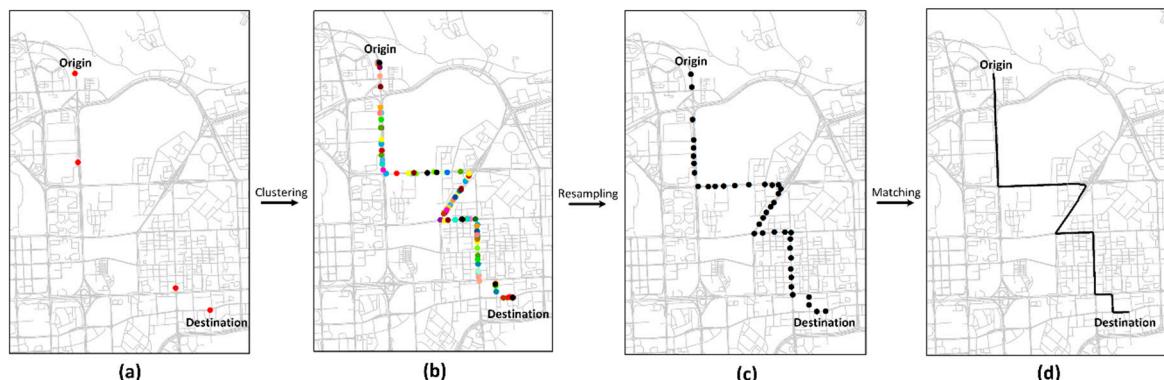


Figure 15. Visualization of map matching for a simulated low-sampling-rate GPS trajectory. (a) Low-sampling rate GPS trajectory. (b) Trajectory cluster. (c) Collaborative trajectory. (d) Map matching result.

Figure 16 illustrates the incorrect paths (shown in the dashed lines) identified by the ST-Matching algorithm under different scenarios. It can be observed in the figure that incorrect map matching components are a result of the correct candidate path not being distinguished from other candidate paths, based on spatial and temporal features, if consecutive GPS points are far from each other.



Figure 16. Incorrect map matching parts of ST-Matching algorithm. (a,b) Scenario T1. (c) Scenario T2. (d,e) Scenario T3.

4.5.3. CMM Performance versus Trajectory Data Size

This section evaluates the impact of trajectory data size on the performance of CMM. The performance of CMM is studied with a varied number of simulated GPS trajectories (50, 300, 500, 700, 1000) on the paths.

In Figure 17a,b, the precision and recall of CMM grow with an increase in trajectory data size. A greater GPS trajectory data size, results in further GPS trajectories used to make up for the missing information. This, in turn, leads to the generation of a collaborative GPS trajectory containing more information after resampling, and thereby, reducing the uncertainty of the group of low-sampling-rate GPS trajectories.

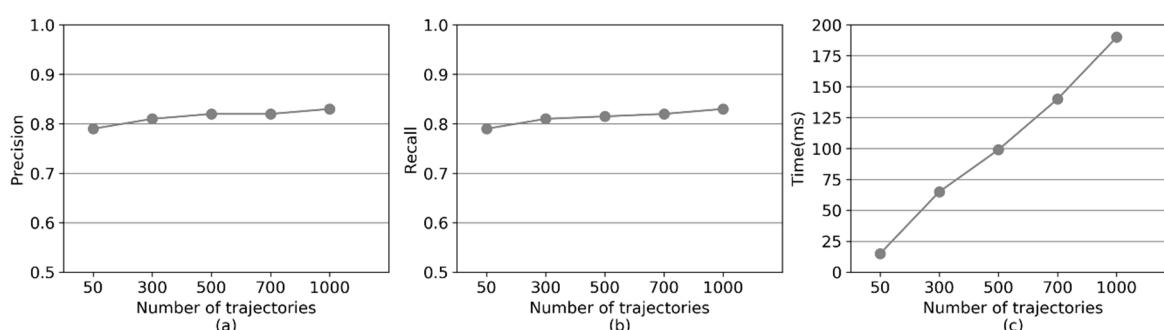


Figure 17. The performance of CMM with different numbers of GPS trajectories. (a) Precision. (b) Recall. (c) Running time.

Figure 17c shows the running time of CMM with different sizes of GPS trajectory data. The average running time per GPS point grows linear with increased trajectory data size. This is because the running time of CMM is mainly composed of two parts—trajectory similarity calculation and candidate path search. With Path-Forest, the candidate path search is efficient. Thus, the time cost for candidate path search does not grow, even if the size of trajectory data is greatly expanded. However, the running time for trajectory similarity calculation will increase with the growing number of GPS trajectories.

5. Conclusions and Future Work

In this paper, a collaborative map matching method (CMM) was proposed for low-sampling-rate GPS trajectories. CMM first groups similar GPS trajectories into clusters based on path similarity and then supplements the missing information of GPS trajectories by resampling. A collaborative trajectory is then extracted for each cluster and is matched to the road network. Experiments were conducted in a real, and a simulated, GPS trajectory dataset. The experiment results show that the proposed CMM outperforms the baseline method in both, effectiveness and efficiency.

In CMM, the similarity measurement for low-sampling-rate GPS trajectory is considered, based on path similarity, which overcomes the deficiency of geometry similarity. A trajectory resampling algorithm is proposed to extract a high-sampling-rate GPS trajectory from each cluster of low-sampling-rate GPS trajectories, which could effectively reduce the uncertainty issue of map matching.

This work can be improved in the following ways: First, the current path similarity method only considers the number of common road segments when measuring the similarity between candidate paths. In future, more features can be involved for path similarity measurement. For instance, the average moving speed of GPS trajectories on candidate paths can be considered for path similarity measurement. Additionally, the grid-based method C-SIM was proposed and seems promising for measuring similarity between the trajectories [14]. The method uses only the spatial aspect of routes and ignores the order of traveling, which makes it simple and fast. It converts the GPS points to cells and fills the gap between cells using linear interpolation if two consecutively generated cells are not adjacent. However, the size of the grid is crucial and should be carefully determined for low sampling trajectories. For this study, if the grid size is too large, dissimilar GPS trajectories would be aggregated into one group; if the grid size is too small, similar GPS trajectories may not be effectively clustered into one group. Therefore, how to utilize the grid-based method in map matching for a low-sampling-rate trajectory will be investigated in future. Second, for each trajectory cluster, only location information of GPS points was considered for trajectory resampling. However, the speed and heading information of new generated GPS points should be estimated after resampling, which could produce a better map matching result. The CMM method will be deployed on a distributed computing platform to handle massive low-sampling-rate GPS trajectories throughout the city.

Author Contributions: Conceptualization, G.C. and X.W.; Formal analysis, W.B. and G.C.; Funding acquisition, X.W.; Investigation, W.B. and G.C.; Supervision, G.C. and X.W.; Validation, W.B.; Visualization, W.B.; Writing—original draft, W.B. and G.C.; Writing—review & editing, G.C. and X.W. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the National Natural Science Foundation of China under Grant Number 61772420 and the Natural Sciences and Engineering Research Council of Canada Discovery Grant provided to Xin Wang.

Conflicts of Interest: The authors declare that there is no conflict of interests regarding the publication of this paper.

References

- Quddus, M.; Washington, S. Shortest path and vehicle trajectory aided map-matching for low frequency GPS data. *Transp. Res. C Emerg. Technol.* **2015**, *55*, 328–339. [[CrossRef](#)]

2. Lou, Y.; Zhang, C.; Zheng, Y.; Xie, X.; Wang, W.; Huang, Y. Map-matching for low-sampling-rate GPS trajectories. In Proceedings of the 17th ACM Sigspatial International Conference on Advances in Geographic Information Systems, Seattle, WA, USA, 3–6 November 2009; pp. 352–361. [[CrossRef](#)]
3. Hsueh, Y.L.; Chen, H.C. Map matching for low-sampling-rate GPS trajectories by exploring real-time moving directions. *Inf. Sci.* **2018**, *433–434*, 55–69. [[CrossRef](#)]
4. 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.* **2017**, *18*, 1241–1254. [[CrossRef](#)]
5. Yin, Y.; Shah, R.R.; Wang, G.; Zimmermann, R. Feature-based Map Matching for Low-Sampling-Rate GPS Trajectories. *Trans. Spat. Algorithms Syst.* **2018**, *4*, 1–24. [[CrossRef](#)]
6. Hu, G.; Shao, J.; Liu, F.; Wang, Y.; Shen, H.T. IF-Matching: Towards Accurate Map-Matching with Information Fusion. *IEEE Trans. Knowl. Data Eng.* **2016**, *29*, 114–127. [[CrossRef](#)]
7. Zheng, K.; Zheng, Y.; Xie, X.; Zhou, X. Reducing uncertainty of low-sampling-rate trajectories. In Proceedings of the International Conference on Data Engineering, Arlington, VA, USA, 1–5 April 2012; pp. 1144–1155. [[CrossRef](#)]
8. Huang, Y.; Rao, W.; Zhang, Z.; Zhao, P.; Yuan, M.; Zeng, J. Frequent Pattern-Based Map-Matching on Low Sampling Rate Trajectories. In Proceedings of the International Conference on Mobile Data Management (MDM), Aalborg, Denmark, 25–28 June 2018; pp. 266–273. [[CrossRef](#)]
9. 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, 12–14 June 2007; pp. 593–604. [[CrossRef](#)]
10. Buchin, K.; Buchin, M.; Gudmundsson, J.; Löffler, M.; Luo, J. Detecting Commuting Patterns by Clustering Subtrajectories. In Proceedings of the International Symposium on Algorithms and Computation, Gold Coast, Australia, 15–17 December 2008; pp. 644–655. [[CrossRef](#)]
11. Kharrat, A.; Popa, I.S.; Zeitouni, K.; Faiz, S. Clustering algorithm for network constraint trajectories. In Proceedings of the Headway in Spatial Data Handling, Montpellier, France, 23–25 July 2008; pp. 631–647.
12. Ester, M.; Kriegel, H.P.; Sander, J.; Xu, X. A density-based algorithm for discovering clusters in large spatial databases with noise. In Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, Portland, OR, USA, 2–4 August 1996; pp. 226–231.
13. Alt, H.; Godau, M. Computing the Fréchet distance between two polygonal curves. *Int. J. Comput. Geom. Appl.* **1995**, *5*, 75–91. [[CrossRef](#)]
14. Mariescu-Istodor, R.; Fränti, P. Grid-based method for GPS route analysis for retrieval. *ACM Trans. Spat. Algorithms Syst.* **2017**, *3*, 28. [[CrossRef](#)]
15. Yen, J.Y. Finding K Shortest Loopless Paths in a Network. *Manage. Sci.* **1971**, *17*, 712–716. [[CrossRef](#)]
16. Vlachos, M.; Kollios, G.; Gunopulos, D. Discovering similar multidimensional trajectories. In Proceedings of the 18th International Conference on Data Engineering, San Jose, CA, USA, 26 February–1 March 2002; pp. 673–684.
17. Yang, J.W.; Mariescu-Istodor, R.; Fränti, P. Three Rapid Methods for Averaging GPS Segments. *Appl. Sci.* **2019**, *9*, 4899. [[CrossRef](#)]
18. Zhu, L.; Holden, J.R.; Gonder, J.D. Trajectory segmentation map-matching approach for large-scale, high-resolution GPS data. *Transp. Res. Board* **2017**, *2645*, 67–75. [[CrossRef](#)]
19. Zhang, D.; Zhao, J.; Zhang, F.; He, T. UrbanCPS: A cyber-physical system based on multi-source big infrastructure data for heterogeneous model integration. In Proceedings of the ACM/IEEE Sixth International Conference on Cyber-Physical Systems, Seattle, WA, USA, 14–16 April 2015; pp. 238–247.
20. Chen, L.; Özsü, M.T.; Oria, V. Robust and fast similarity search for moving object trajectories. In Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, Baltimore, MD, USA, 14–16 June 2005; pp. 491–502.



© 2020 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<http://creativecommons.org/licenses/by/4.0/>).