Theoretical Computer Science 941 (2023) 104–120

Contents lists available at ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

A robust map matching method by considering memorized multiple matching candidates ☆

Wanting Li, Yongcai Wang *, Deying Li, Xiaojia Xu

School of Information, Renmin University of China, Beijing 100872, China

ARTICLE INFO

Article history:
Received 17 September 2022
Received in revised form 17 October 2022
Accepted 22 October 2022
Available online 18 November 2022

Keywords:
Map matching
Multiple candidate
Road continuity
Online map matching
Offline map matching

ABSTRACT

Map matching is to track the positions of vehicles on the road network based on the positions provided by GPS (Global Positioning System) devices. Balancing localization accuracy and computation efficiency is a key problem in map matching. Existing methods mainly use Hidden Markov Model (HMM) or historical transportation data to learn the transitional probabilities among road segments. Although the roads to explore can be remarkably reduced by the Markov assumption, miss-of-match and matching breaks may occur if the GPS data is highly noisy, and the transitional model needs to be learned offline. To address these problems, this paper presents Multiple Candidate Matching (MCM) to improve the robustness of map matching. MCM doesn't need to pre-train the transitional model nor the historical transportation information. MCM memorizes multiple historical matching candidates in the map matching process. It votes among historical matchings and current matchings, but generates limited number of road candidates in real-time to restrict the computation complexity. MCM for both online map matching and offline map matching are presented and their properties are analyzed theoretically and experimentally. Numerical experiments in large-scale data sets show that MCM is very promising in terms of accuracy, computational efficiency, and robustness. The matching break and miss-of-match problems can be resolved effectively when compared with the state-of-the-art map matching methods. Codes are outsourced at https://github.com/lindalee-inlab/MCM.

1. Introduction

GPS-based navigation is essential in daily driving, in which, a crucial requirement is to locate car to its correct route for generating correct navigation information. Due to the measurement noises of the GPS equipment, GPS reported positions might deviate from the real road. The GPS noises may be caused by different reasons, such as when the vehicles are under bridges or in tunnels [2] or satellite signals' multi-path effects [3–5] in built-up city areas.

In order to locate vehicles accurately, researchers proposed to match the trajectory of a vehicle with the known road network information. By using the continuity constraint of the vehicle's motion and the continuity characteristics of the roads, the vehicle localization problem becomes to find the most likely road that best matches the GPS trajectory, which is called the *map matching* problem [6]. Map matching can be classified into online map matching [7–11] and offline map

☆ This is an enhanced and extended version of a paper [1] presented at AAIM2022. This work is partially supported by the National Natural Science Foundation of China Grant No. 12071478, 61972404. Public Computing Cloud, Renmin University of China.
* Corresponding author.
E-mail addresses: lindalee@ruc.edu.cn (W. Li), ycw@ruc.edu.cn (Y. Wang), deyingli@ruc.edu.cn (D. Li), xuxiaojia@ruc.edu.cn (X. Xu).

https://doi.org/10.1016/j.tcs.2022.10.032
0304-3975/

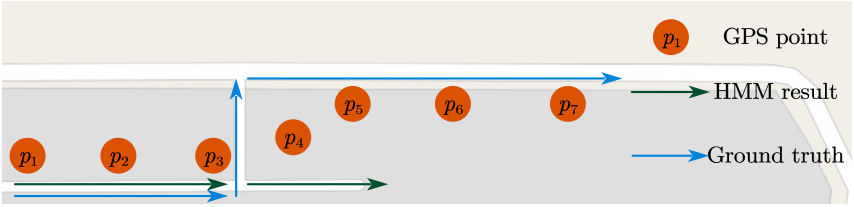


Fig. 1. Matching break. The matching break is a common problem in map-matching, which is mainly caused by trajectory outliers. Because of Markov assumption, HMM algorithm matches the wrong side road with a higher probability at point p_4, but there is no way to correct it, and a break occurs. (For interpretation of the colors in the figure(s), the reader is referred to the web version of this article.)

matching [12–18]. The former estimates the current road segment the vehicle is on immediately after a GPS data is collected. The latter recovers the traveled roads by offline processing the collected whole GPS trajectory.

Online map matching requires both matching accuracy and efficiency. For efficiency purpose, Markov assumption is widely adopted, which assumes the road-matching state at time t depends only on the states at time $t-1$, which is not related to the states and observations of early times. Based on the Markov assumption, two categories of methods, i.e., *Hidden Markov Model (HMM)* based methods [7–11] and *Multiple Hypothesis Techniques (MHT)* [19,11,20] are mainly proposed in literature for online map matching.

However, in online HMM algorithm, the matching results before time $t-1$ are not traced back. One mismatch at t may lead to cascaded errors in the later associations, which may cause errors, such as matching break [21] shown in Fig. 1. The green trajectory is the estimated trajectory using online HMM method. Because the side road ends at that point, a matching break happens. In addition, the HMM-based method needs to train the probability model in advance. The sliding window-based method has a certain delay.

To address the above problems of HMM, MHT [19,11,20] generates a variety of road hypotheses at time t through utilizing historical transportation information to filter the probabilities of choosing the subsequent road. In complex environments with large GPS errors, it still leads to subsequent matching errors due to a wrong early matching.

Offline map matching is batching the whole input trajectory to find the optimal matching path in the road network. Because the trajectory to road matching is more reliable than the single point matching [22,6,23], it has attracted great research attentions. Three kinds of approaches are mainly proposed in the literature: (1) Similarity model-based; (2) Hidden Markov Model (HMM)-based, and (3) Multiple Hypothesis Technique (MHT)-based.

Although users care only their current positions, historical data still has great value if historical matching candidates are tracked to correct the current matching failures. This paper relaxes the Markov assumption but still designs a highly efficient map matching algorithm, i.e., *Multiple route Candidate Matching (MCM)*. MCM is essentially to find the longest common sub-sequence between the GPS trajectory and the potential routes generated from the road network. MCM memories the possible historical matching candidates and prunes the impossible candidate paths by utilizing road and trajectory continuity to restrict computation complexity. So that the matching accuracy and efficiency are balanced. The contributions of MCM method is as following:

(1) MCM shows strong fault tolerance. The likelihood of multiple route candidates is tracked by a dynamic programming process using a similarity matrix. And a "last" label is used in each row of the similarity matrix to record the historical matching point. Even if the matching at t is wrong, because of saving the multiple matching candidates, the matching result can be corrected when the correct candidate comes to surface in subsequent matching.

(2) The most unlikely routes are autonomously excluded to control the number of "alive" candidates by continuity constraints so that the computation is efficient. In our early work [1], we used directly successive roads as the constraint of road continuity. When the GPS sampling frequency is low, the trajectory points may not be on the directly successive roads. So in this paper, we use the distance of *the shortest path* between two roads as the road continuity constraint for pruning. *The shortest path* is the distance along the path between two corresponding road points.

(3) We also trace back the optimal matching results in the offline stage through iteration, which can avoid the HMM break problem and effectively improve the matching accuracy.

Experiments on two widely used map matching datasets show that the proposed MCM method provides the highest mapping accuracy compared with state-of-the-art online and offline map matching algorithms. The decrease of GPS sampling frequency has little effect on MCM's matching accuracy than other methods. At the same time, the proposed pruning schemes in MCM using trajectory continuity keep the efficiency of MCM. We outsource the codes and provide offline and online demos for MCM for potential use by the society at https://github.com/lindalee-inlab/MCM.

2. Related work

Map matching can be divided into two application scenarios: online matching and offline matching. We introduce related works from these two aspects.

Table 1
Online map matching method.

HMM-based method	Name	Sliding window	Observation probability	Transition probability	Delay time	Pre training model
	ST matching [7]	Fixed size	|	Speed constraint	No delay	✓
	Online HMM [8]	Fixed size	Speed constraint	SVM (Speed and Distance)	Delay	✓
	Snapnet [9]	Fixed size	Vehicle heading; road level	|	No delay	✓
	Route choice HMM [10]	Variable size	|	Distance difference; free-flow travel time	Delay	✓
MHT-based method	Name	Bayesian filtering model	Observation probability	Transition probability	Delay time	Pre training model
	MHT [19]	EKF	|	Distance and Speed Bayesian probability	No Delay	|

2.1. Online map matching

Online map-matching outputs the current road segment the vehicle is on immediately after a GPS data is collected. Two categories of methods, i.e., *Hidden Markov Model (HMM)* based methods and *Multiple Hypothesis Techniques (MHT)* are mainly proposed in literature for online map matching.

(1) HMM-based online methods calculate the hidden Markov chain states in real time. In order to conduct real-time map matching by HMM-based method, two methods have been introduced, i.e., the approximate algorithm and the sliding window based algorithm.
   1) The approximate algorithm greedily calculates the current optimal result [7,9]. These methods use a sliding window to calculate the optimization function of a sub-trajectory including future GPS points and output the results in real-time without delay. The approximation algorithm retains only current optimal path in each step, which can not guarantee that the predicted state sequence as a whole is the most likely state sequence.
   2) The other method uses the sliding window Viterbi algorithm to calculate the optimal path in a period of time [8, 10,11]. By waiting for some GPS points after time t to fill a window of observations, the matched road at time t is evaluated using Viterbi algorithm. The future GPS information will help to improve the matching accuracy at t, but the location prediction needs to be delayed.

(2) To avoid pre-train models and delay, an online MHT method [19] is proposed. This method uses Bayesian filtering. The goal is to obtain the probability distribution of the state quantity at time $t-1$ when the prior probability is known and to estimate the posterior probability distribution of the state quantity at time t when the observation and transition probability matrix at time t are known. This method avoids delay but needs more historical transportation statistical knowledges to generate a route prediction model.

The advantages and disadvantages of various online map matching algorithms are compared, as shown in Table 1.

2.2. Offline map matching

Offline map-matching is performed after the whole trajectory is obtained. It aims for the optimal route matching with less constraints on the processing time. In offline map matching, there are mainly three kinds of methods: (1) Similarity model based; (2) Hidden Markov Model (HMM)-based; and (3) Multiple Hypothesis Technique (MHT)-based.

(1) The similarity model based methods refer to a category of approaches [24–29] that evaluate the road that is the closest to the GPS trajectory, geometrically and/or topologically. The main focus in this category is how to define the closeness. The representative algorithms include Fréchet distance [28] and Longest Common Subsequence (LCSS) [29] distance. Wei et al. [28] proposed to use Fréchet distance to measure the matching degree between the GPS sequence and the candidate road sequence. Zhu et al. used Longest Common Subsequence (LCSS) [29], which divides a trajectory into multiple segments and finds the shortest path on the map for each pair of start and endpoints of a trajectory segment. The shortest paths are then concatenated to form the final path while their corresponding LCSS scores are summed. The path whose LCSS score is the highest is regarded as the final matching result. The similarity model-based matching algorithms are relatively simple and easy to be implemented. But they are still susceptible to GPS noise and data sparsity.

(2) HMM-based methods are the most popular. HMM is a prevailing paradigm of dynamic programming model, which well suits the process of finding the most suitable roads (i.e., hidden state) matching with the GPS points (i.e., observed state). Newson et al. [12] make predictions for a given point based on the combination of the point's own position and the position of the point which precedes it. This means that topological information and travel distance can both be taken into account. The main advantage of using such an approach is to attain a relatively high accuracy whilst requiring much less processing time and memory than comparing the whole trajectory. Based on the framework of HMM, most of the recent efforts are devoted to improve accuracy by introducing new information such as speed limit ([13,30]), turning angle ([31]), and curvedness ([15]) or designing more robust and realistic objective functions for path inference ([13], [32], [33]). Some methods accelerate the processing time [18]. The HMM-based algorithms greatly improve the matching accuracy than similarity model based algorithms. The trajectory can be matched even when GPS points are noisy or in low sampling

Table 2
Offline map matching method.

Similarity model-based method	Name	Distance model	Pre training model	Matching break
	Fréchet [28]	Fréchet distance	|	|
	LCSS [29]	LCSS distance	|	|
HMM-based method	Name	Observation probability&Transition probability	Pre training model	Matching break
	Offline HMM [12]	|&|	✓	✓
	Interactive-voting matching [13]	|&Speed constraint	✓	✓
	Unsupervised HMM [32]	Antenna location&Speed constraint	✓	✓
	Quick matching [30]	Speed constraint&|	✓	✓
	Multistage matching [31]	Vehicle heading&		
Heading change difference	✓	✓		
	Driver path preference based HMM [33]	|&speed constraint; driver's travel preference	✓	✓
MHT-based method	Name	Bayesian filtering model	Pre training model	Matching break
	BBN [34]	Bayesian belief network	|	✓
	PT [35]	particle filtering model	|	✓

Table 3
NOTATION.

symbol	description	symbol	description
$\mathbf{T}$	GPS trajectory	p_i	the ind sample point in trajectory $\mathbf{T}$
x_i	longitude of the ind sample point	y_i	latitude of the ind sample point
t_i	timestamp of the ind sample point	G	Road Network
V	vertex set of road network G	v	vertex of V
E	road set of road network G	e	road of E
$e.\mathcal{S}$	start vertex of road e	$e.\mathcal{E}$	end vertex of road e
$e.\mathcal{L}$	polyline of road e	R	route
R^*	the best matching route of trajectory $\mathbf{T}$	$\mathbf{S}(p, e)$	node similarity of sample point p and road e
$M(\mathbf{T}, R)$	trajectory similarity of route R and trajectory $\mathbf{T}$	$\mathbf{M}(\mathbf{T}, G)$	matching function of road network G and trajectory $\mathbf{T}$

rates. However, the HMM algorithms are sensitive to outliers, which can easily cause "matching breaks", which means that the trajectory is failed to match any road segment at some point. Fig. 1 gives an example of matching break. Matching break happens after p_4. At the same time, pre-training transitional probabilities among states are generally required in HMM-based methods, which requires huge training data and prior training efforts.

(3) The MHT-based method uses Bayesian filtering technology to directly solve the map matching problem through sensor fusion and measurement correction. For example, Bayesian belief network (BBN) method [34] uses a Bayesian belief network to select the next route candidate in the route list. Particle filter (PF) method [35] uses particle filter model to recursively estimate the Probability Density Function (PDF) of the potential position as time and observations advance. These methods have good matching results and do not need to train the model in advance, but need to design the best Bayesian filtering model.

The above offline map matching algorithms are summarized in Table 2, which introduce more restriction information and increase the matching accuracy, but can not meet the real-time performance.

3. Problem model

3.1. Preliminaries

This section defines the map-matching problem and relevant concepts (Table 3):

Definition 1 *(Trajectory).* A trajectory $\mathbf{T}$ is a sequence of chronologically ordered spatial points $\mathbf{T}: p_1 \rightarrow p_2 \rightarrow \ldots \rightarrow p_n$ obtained from GPS sensor. Each point p_i consists of a 2-dimensional coordinate $\langle x_i, y_i \rangle$ and a timestamp t_i. $p_i = \langle x_i, y_i, t_i \rangle$

Definition 2 *(Road Network).* A road network (also known as map) is a directed graph $G = (V, E)$, in which a vertex $v = (x, y) \in V$ represents an intersection or a road end, and an edge $e = (\mathcal{S}, \mathcal{E}, \mathcal{L})$ is a directed road starting from vertex $\mathcal{S}$ and ending at $\mathcal{E}$ with a polyline $\mathcal{L}$ representing a sequence of spatial points.

Definition 3 *(Route).* A route R represents a sequence of connected edges in the road network, i.e. $R: e_1 \rightarrow e_2 \rightarrow \ldots \rightarrow e_l$, where $e_i \in E, i \in [1, l-1]$ and $e_i.\mathcal{E} = e_{i+1}.\mathcal{S}$.

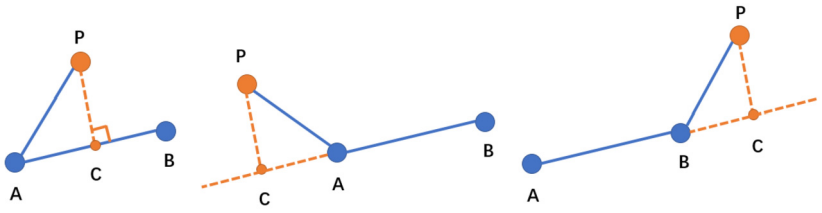


Fig. 2. AB is the segment; P is the point and C is P's projection on AB. (a) C is on AB; (b) C is not on AB and is closer to A; (c) C is not on AB and is closer to B.

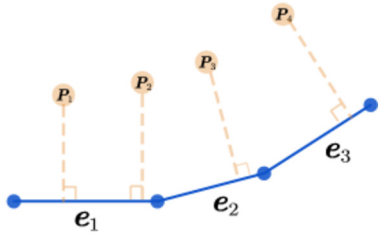


Fig. 3. Trajectory-to-route similarity by trajectory p_1, p_2, p_3, p_4 to route $e_1 \rightarrow e_2 \rightarrow e_3$.

Definition 4 *(Map Matching).* Given a road network $G(V, E)$ and a trajectory $\mathbf{T}$, the map-matching problem finds a route R^* that best represents the sequence of roads traveled by the trajectory $\mathbf{T}$.

3.2. Point-to-road similarity

There are many ways to define the similarity between a GPS point and a road segment. In our case, we use point-to-line distance with a threshold to define the point-to-road similarity. It truncates very low similarity for efficiency.

Definition 5 *(Point-to-Road Similarity).* The point-to-road similarity $\mathbf{S}(p_i, e_j)$ between a point $p_i \in \mathbf{T}$ and a road $e_j \in \mathbf{G}$ is defined by equation (1).

$$\mathbf{S}(p_i, e_j) = \begin{cases} \varepsilon - \left\| p_i - e_j \right\|_2, \text{if } \left\| p_i - e_j \right\|_2 < \varepsilon \\ 0, \text{ otherwise} \end{cases} \tag{1}$$

ε here is a threshold to eliminate the similarity calculation if the point is too far away from the road segment, which is helpful to reduce the amount of subsequent calculation. $\left\| p_i - e_j \right\|_2$ is the distance from the point p_i to the road segment e_j, which is defined by *the shortest distance from the point to the line segment* as shown in Fig. 2.

In the point to line distance as shown in Fig. 2, let's assume the line segment is AB and the point is P. The projection from P to AB is denoted by C. If C is on AB, $\|p_P - e_{AB}\|_2 = \|PC\|$. If C is not on AB and C is closer to A, then $\|p_P - e_{AB}\|_2 = \|PA\|$; If C is not on AB and C is closer to B, then $\|p_P - e_{AB}\|_2 = \|PB\|$;

3.3. Trajectory-to-route similarity

We then consider to evaluate the similarity between a GPS trajectory $\mathbf{T}$ and a route R on G. Suppose $\mathbf{T}$ is composed by a set of successively measured GPS points, i.e., $\mathbf{T} = \{p_1, p_2, \cdots, p_n\}$. Suppose R is composed by a set of sequentially connected edge segments, i.e., $R = \{e_1, e_2, \cdots, e_m\}$.

Definition 6 *(Trajectory-to-route Similarity).* Given $\mathbf{T} = \{\mathbf{p_1}, \mathbf{p_2}, \cdots, \mathbf{p_n}\}$ and $R = \{e_1, e_2, \cdots, e_m\}$, the trajectory to route similarity $M(\mathbf{T}, R)$ is defined as:

$$M(\mathbf{T}, R) = \sum_{i=1}^{n} S\left(p_i, e_{nearest}\left(p_i\right)\right) \tag{2}$$

where $e_{nearest}(p_i)$ is the route on R which has the minimum distance to p_i as shown in Fig. 3.

Then, let's denote R^* the route on G, which matches best with $\mathbf{T}$. Then the goal of map matching is to find the route with the best similarity score with $\mathbf{T}$.

$$R^* = \underset{R}{arg\max}\, M(\mathbf{T}, R) \tag{3}$$

where R is any route that can be generated from G.

4. Multiple candidate matching

Enumerating all the possible routes on G is computational complexity explosive. MCM generates routes in a controlled way. Instead of training the transitional model or using additional road level or travel speed information, MCM uses only the trajectory and the roads' continuity information.

MCM proposes a method by matching the trajectory to the map via generating multiple candidate routes. It tracks the matching probabilities of multiple routes and outputs the best matching result. MCM is mainly divided into two steps. Firstly, the candidate routes are generated based on the current "alive" matches in the matching table, *Q last* (storing the end edge of the alive routes), and the continuity constraint of the routes on the route graph; Secondly, the trajectory-to-road similarities for the multiple route candidates are evaluated using a dynamic programming model. In this step, the matching similarities to all potential routes will be evaluated; the best match is output as the current result, and some unlikely routes will be pruned for keeping computation efficiency.

MCM proposes an *score matrix* **M** to explore the likelihoods of all potential routes that may match with the GPS trajectory. The values in the matrix **M** represent the trajectory-to-road similarities of the candidate routes. The rows of the matrix represent the edge segments in the road network, and the columns of the matrix represent the GPS points on the trajectory ordered by the collecting time. The map matching process is indeed to update the score matrix **M**. Because the trajectory points are collected in order, at each time t, when a new point p_t is obtained, we need to fill a new column, i.e., the tth column of **M**.

The summation of all the point-to-road similarity values of a trajectory sequence is the trajectory-to-route similarity. We call each value in the score matrix as trajectory-to-route similarity value. Fig. 4 shows an example of a score matrix where $n = m = 7$. Map matching uses this score matrix. The concepts used in MCM and the steps to fill the score matrix are as follows.

4.1. Route candidates and the last alive matching pairs

For finding the route candidates on G that may match with **T**, we use roads' continuity information. Based on the neighbor edges we got in the previous step, If there is a path that conforms to the road topology in the last matching pairs, it indicates that one of the route candidates can be continued.

Definition 7 *(Alive Routes, i.e., Route Candidates).* Alive routes in **M** records the potential candidates of routes that may match with the trajectory **T**. Each route candidate is composed by a sequence of connected edges.

Suppose at time $t-1$, there are K alive route candidates in **M**. For each route candidate, we record only the last edge of each route to represent the route. This is because when a route's similarity score is obtained at time t, we can trace back the whole route from the last edge of that route in **M**.

Definition 8 *(Last Matching Pair).* The last edge on each route is saved as a *last matching pair* $last_k = (e, p)$, where e is an edge index and p is a point index. It means on the kth route, the last matching point p on **T** matches with the edge e on G. It also means that the entry (e, p) in **M** is the endpoint of the kth alive route.

At the time t, we assume the total number of alive routes is K, and these K alive routes' *last matching pairs* are saved in a queue data structure *Q Last*. The following functions are defined to return the edge index and point index in the kth route's last matching pair.

Definition 9 *(The last$(\cdot)$ Function).* Suppose $l_k = (e, p)$ is the last matching pair of the kth alive route, the function $e(l_k) = e$ returns the edge index saved in l_k and $p(l_k) = p$ returns the point index in l_k.

Then route generation considers the route continuity information on map G.

Definition 10 *(The near$(\cdot)$ Function).* A *near*() function is designed to restrict MCM to generate only reasonable routes based on the road network topology. $near(p_i, e_j) = \{e \in \varepsilon_r | \text{abs}\left(\left\|e_j - e_{j+1}\right\|_2 - \|p_i - p_{i+1}\|_2\right) < \varepsilon_{topo}\}$ where $\left\|e_j - e_{j+1}\right\|_2$ is the shortest distance between two edges calculated by Dijkstra algorithm.

We first take the midpoint of the current point p_i and the next point p_{i+1} as the center of the circle, take half the distance between the two points plus the error value ε_r as the radius, and select the road network in the circle as the edges involved in the calculation. In this way, many less likely candidate edges can be deleted. The next is to traverse all the edges

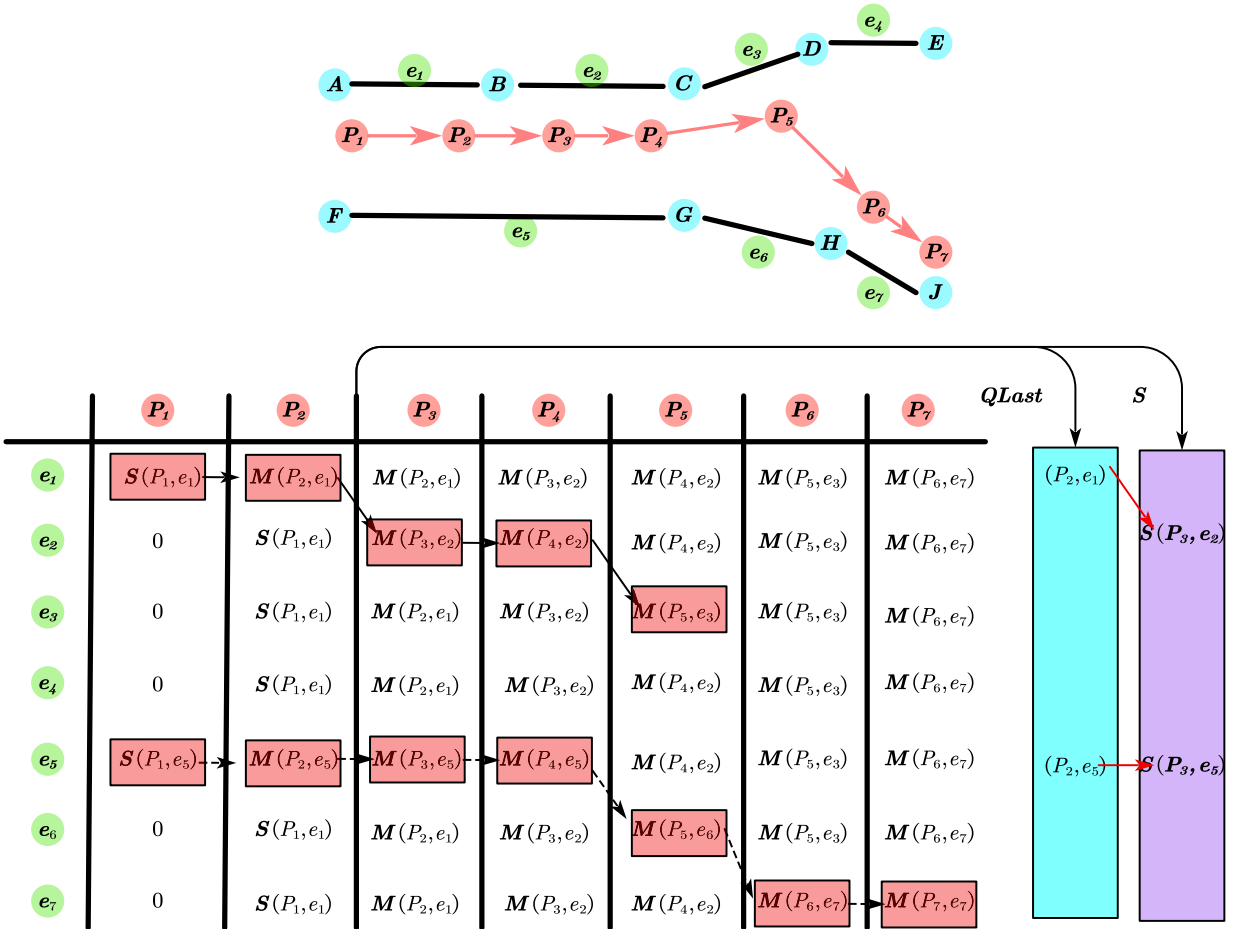


Fig. 4. The figure shows a score matrix. All the route candidates are shown in the figure. We calculate neighbors for each node (only cells within radius in ε in each column). The last alive matching pairs are store in queue $Q\,last$ and the point-to-road similarity matrix **S**. The dotted line is the trajectory which has maximum score.

within the circle and calculate the shortest distance between them and the edge e_j. If the difference between the shortest distance and the two points is less than the error threshold ε_{topo}, it is selected as a near edge of (p_i, e_j).

As shown in Fig. 5, the edges in the dotted circle are involved in the calculation. Suppose the corresponding edge of GPS1 is e_{ab}, and the corresponding point is p_{ab}. According to the position of GPS2, the edges that meet the condition in the circle are $near(p_1, e_{ab}) = \{e_{kl}, e_{km}, e_{mn}\}$. For example, the shortest path between e_{ab} and e_{km} is $\{|p_{ab}->p_{be}|, e_{be}, e_{ej}, e_{jg}, e_{gk}, |p_k->p_{km}|\}$ whose sum length is less than ε_{topo}, so e_{km} is a near edge of (p_1, e_{ab}).

4.2. Updating the score matrix

We use the dynamic programming method to calculate the score matrix. At time t, when a new GPS point p_t is obtained, we check all the alive routes' last edges, i.e., all the $l_k \in Q\,Last$. $\mathbf{M}(e_i, p_t)$ is filled by one of the following three cases:

(1) For each l_k we find $near(e(l_k))$, i.e., all connected edges of the last edge. Then we calculate the similarity scores $\mathbf{S}(e_i, p_t)$ for every $e_i \in near(e(l_k))$. If $\mathbf{S}(e_i, p_t) > 0$, the score of e_i obtained from the kth alive route, denoted by $\mathbf{M}_k(e_i, p_t)$ is calculated by:

$$\text{if } \mathbf{S}(e_i, p_t) > 0 \& e_i \in near(e(l_k)), \mathbf{M}_k(e_i, p_t) = \mathbf{M}(e(l_k), p(l_k)) + \mathbf{S}(e_i, p_t) \tag{4}$$

Then all the K alive routes will be processed to calculate (4). The updated score of $\mathbf{M}(e_i, p_t)$, i.e., the score at the ith row and tth column in **M** is filled by the highest score calculated from all the K route candidates.

$$\mathbf{M}(e_i, p_t) = \max_{k=1:K} \mathbf{M}_k(e_i, p_t) \tag{5}$$

(2) If an e_i is not in the near edge set of any route's last edge, but $\mathbf{S}(e_i, p_t) > 0$, a new route candidate will be generated. Its matching score is filled as:

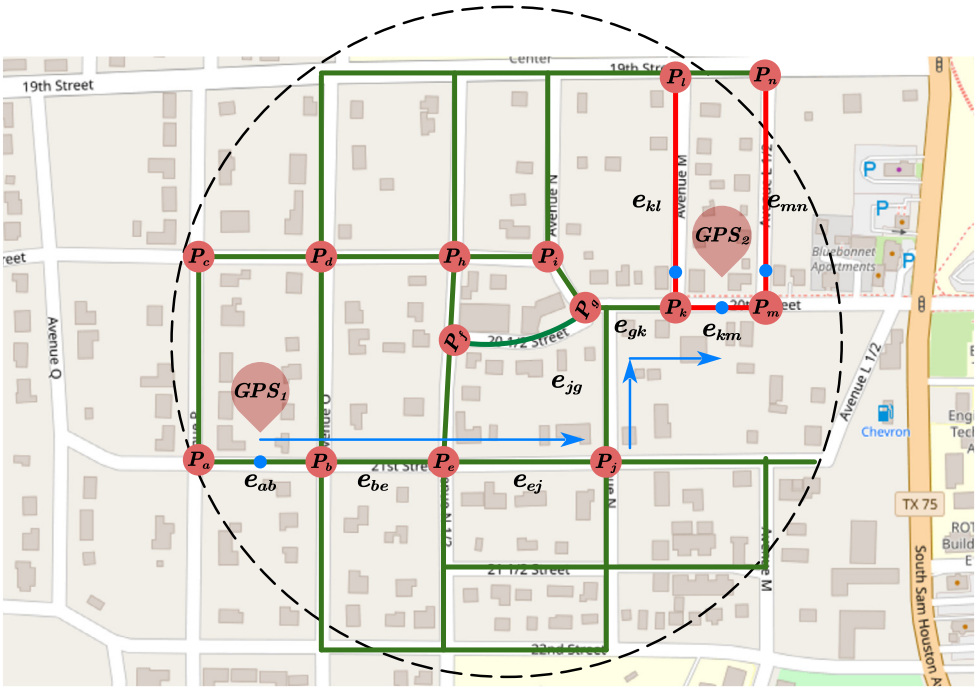


Fig. 5. The *near*() function to generate only reasonable routes.

$$\mathbf{M}(e_i, p_t) = \mathbf{S}(e_i, p_t), \text{if}\, \mathbf{S}(e_i, p_t) > 0 \& \forall l_k, e_i \notin near(l_k) \tag{6}$$

(3) If an edge e_i has $\mathbf{S}(e_i, p_t) = 0$, it means the point p_t is not likely to be matched with e_i, so the matching score at $\mathbf{M}(e_i, p_t)$ is filled by copying the highest score of the last column, which means it considers the last best match is still the best match.

$$\mathbf{M}(e_i, p_t) = \max_{k=1:m} \mathbf{M}(e_k, p_{t-1}) \tag{7}$$

The overall equation to fill the matching score at $\mathbf{M}(e_i, p_t)$ is therefore given in

$$\mathbf{M}(e_i, p_t) = \begin{cases} \max\limits_{k=1:K} \{\mathbf{M}(e(l_k), p(l_k)) + \mathbf{S}(e_i, p_t)\}, & \text{if } \mathbf{S}(e_i, p_t) > 0 \& e_i \in near(e(l_k)) \\ \mathbf{S}(e_i, p_t), & \text{if } \mathbf{S}(e_i, p_t) > 0 \& e_i \notin near(e(l_k)), \forall l_k \\ \max\limits_{k=1:m} \{\mathbf{M}(e_k, p_{t-1})\}, & \text{otherwise} \end{cases} \tag{8}$$

So the overall routine in MCM for the score matrix updating is as described below. The pseudocode for MCM is given in Algorithm 1.

(1) Initialize score matrix with zero and the last matching pair as empty. (Line 1-2)

(2) Find neighbor roads within radius in ε for the current view of the vehicle. (Line 3-7)

(3) Find the route candidates based on the last matching pairs and the neighboring roads. (Line 10)

(4) Finally, for each route candidate, calculate the similarity score with point-to-road similarity values, and update the score matrix and the last matching pairs. (Line 11&14&16)

An example is shown in Fig. 4. After completing the matching at time t_2, we get the alive candidate routes as $e_1 \to e_1$ and $e_5 \to e_5$, and the two last matching pairs are (e_1, p_2) and (e_5, p_2). At time t_3, we first find neighbor roads $\{e_2, e_5\}$. Then we match the route candidates based on the last matching pairs and the neighboring roads. In the route $e_1 \to e_1$, (e_2, p_3) satisfies the *near*(·) function. So we get $\mathbf{M}(p_3, e_2) = \mathbf{M}(p_2, e_1) + \mathbf{S}(p_3, e_2)$ by (8). The other alive routes obtained in the same way. At time t_3, the last matching pairs $Q.Last$ are updated to $\{(e_2, p_3), (e_5, p_3)\}$. So that the matching at time t_3 is finished and the matching scores are filled in the t-th column of $\mathbf{M}$.

At time t_6, only one alive route can be found and the last matching pair is updated to be (e_5, p_5) to (e_5, p_6). Finally we can get the score matrix M which stores the alive candidate routes as shown in Fig. 4.

5. MCM for online map matching

In online map matching, the problem is to find the associated roads for the trajectory up to time t.

Algorithm 1 Score Matrix updating.

Input: Graph $G = \{V, E\}$ and trajectory T
Parameter: Threshold ε
Output: Score Matrix **M**

```
1: global M = initialize(len(G), len(T))// global variable score matrix.
2: global QLast = initialize(1,len(G))// global variable record last alive matching node.
3: for j = 1 to n do
4:     if max_{i∈[1,m]} S(i, j) > 0 then
5:         M(i, :) = S(i, :); Qlast(i) = j;Update near(p_i, e_j);break;
6:     end if
7: end for
8: for j to n do
9:     for i = 1 to m do
10:        if S(i, j) > 0&&Qlast(k) > 0 then
11:            M(i, j) = max_{k∈[1,m]} {M(k, j − 1) + S(i, j)}; Qlast(i) = j; Qlast(k) = 0;Update near(p_i, e_j);
12:        else
13:            if S(i, j) > 0 then
14:                M(i, j) = S(i, j); Qlast(i) = j;Update near(p_i, e_j);// Begin again.
15:            else
16:                M(i, j) = max_{i∈[1,m]} {M(i, j − 1)}; Qlast(i) = 0;
17:            end if
18:        end if
19:    end for
20: end for
```

Algorithm 2 MCM online matching algorithm.

Input: Graph $G = \{V, E\}$ and trajectory T
Output: matches=Avector containing matching indices

```
1: Calculate score matrix M by column at time t when S(e_i, p_t) > 0.
2: Update QLast(i) = t.
3: Find out e_i that has the highest matching score with p_t, i.e., max_{i∈[1,m]} M(e_i, p_t).
4: matches(t) = e_i;
5: if S(e_i, p_t) = 0 then
6:     M(i, t) = max_{i∈[1,m]} {M(i, t − 1)}; Qlast(i) = 0;
7: end if
```

5.1. Algorithm

Based on the online updating of the matching matrix and the last queue, MCM outputs the candidate matching roads with the best matching score up to time t. So in online matching it can efficiently find the maximum $\mathbf{M}(e_i, p_t), i \in [1, m]$ at time t in case $\mathbf{S}(e_i, p_t) > 0$. Then it select this path as the matched route. So the overall routine in MCM for online matching is as described below. The pseudocode for MCM for online matching is given in Algorithm 2.

(1) For each route candidate, calculate the similarity score with point-to-road similarity values, and update the score matrix and the last matching pairs. (Line 1&2)

(2) Then, find the best matching road at t. (Line 3)

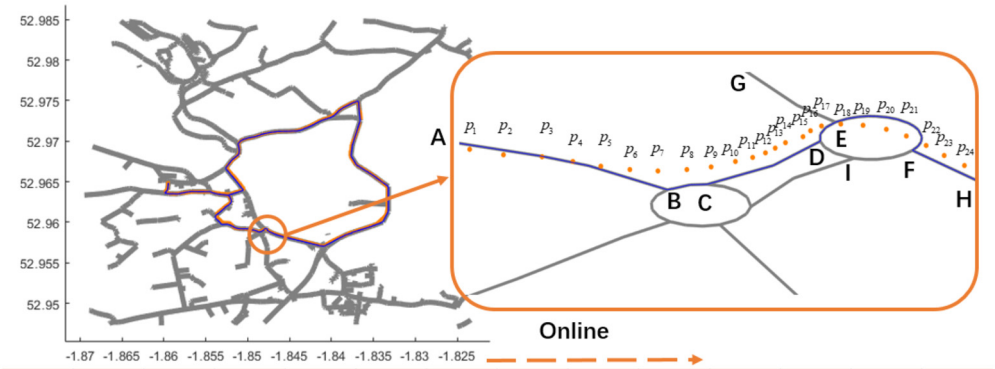
(3) Finally, fill the column t of the score matrix and update the last matching pairs. (Line 6)

5.2. Example

For example, as shown in Fig. 6, multiple candidate routes are generated in the online matching process. We fill in the matrix **M** by column during the online matching process. From p_1 to p_{15}, there is only one candidate route $\{AB, BC, CD\}$. At p_{16}, we find the matching pairs are (DE, p_{16}), (CD, p_{16}) and (EG, p_{16}). So there are three candidate routes $\{AB, BC, CD, DE\}$, $\{AB, BC, CD\}$ and $\{EG\}$. We give out the most likely route $\{AB, BC, CD, DE\}$ by $\max_{i\in[1,m]} M(e_i, p_{16})$ and trace three candidate routes at p_{17}. Finally at p_{24}, when we find the last matching pair is (FH, p_{24}), the most likely route can be traced back as $\{AB, BC, CD, DE, EF, FH\}$, which has the highest trajectory-to-road similarity following the transitions recorded in **M**.

5.3. Online algorithm feasibility

In the *near* function, let $\varepsilon = \varepsilon_r$, where ε here is a threshold to eliminate the similarity calculation (as defined in (1)) and ε_r is the error value as the radius given in Definition 10. Then *near* function can correctly find out all possible edges



	P1	P2	···	P8	P9	···	P15	P16	P17	P18	P19	···	P24
AB	10	20		65	70		93	95	98	108	117		148
BC	0	10		70	70		93	95	98	108	117		148
CD	0	10		65	75		95	98	98	108	117		148
DE	0	10		65	70		93	97	108	115	117		148
DI	0	10		65	70		93	95	98	108	117		148
IF	0	10		65	70		93	95	98	108	117		148
EG	0	10		65	70		93	3	107	117	117		148
EF	0	10		65	70		93	95	98	116	124		148
FH	0	10		65	70		93	95	98	108	117		156

Fig. 6. The orange path in the figure is the matching result of GPS points. From the figure, we can see that there are two inaccurate matches at p_{18}. But as we can seen at p_{19}, we finally match the better edge EF use continuous similarity of trajectories. Finally, we can deduce the path by finding the maximum value of **M**.

involved in the calculation. Traversing and calculating the shortest distance between two sides of the road network requires $O(mlogm)$. Therefore, *near* function reduce the number of calculated edges to reduce the computation complexity. Then we prove that at time t, $\max_{i\in[1,m]} \mathbf{M}(e_i, p_t)$ found by MCM method will output the ground truth route.

Theorem 1 *(Effectiveness of MCM online Method). Considering a trajectory* **T** *with length t, if the sum matching score between* **T** *and the ground true route* $\sum_{i=1}^{t} \left\| p_i - e_{real,i} \right\|_2$ *is the largest among all the candidate routes and* $\left\| p_i - e_{real,i} \right\| < \varepsilon$ *where* $e_{real,i}$ *is the ground truth edge matching with* p_i *on* **T**, *then MCM can correctly output the ground truth route as the optimal route at time t.*

Proof of MCM online Method. First of all, when $t = 1$, if $\sum_{i=1}^{1} \left\| p_i - e_{real,i} \right\|_2$ is the largest among all the candidate routes, then according to (8), $\max_{i\in[1,m]} \mathbf{M}(e_i, p_1)$ will output $e_{real,1}$ as the best matching route. So the theorem holds for $t = 1$. We suppose the claim holds at time k, i.e., if $\sum_{i=1}^{k} \left\| p_i - e_{real,i} \right\|_2$ calculate all the candidate routes and store them. We then consider the case at time $k+1$. At time $k+1$, if $\sum_{i=1}^{k+1} \left\| p_i - e_{real,i} \right\|_2$ is the largest among all the candidate routes, it may become the largest due to two cases.

(1) $\sum_{i=1}^{k} \left\| p_i - e_{real,i} \right\|_2$ is the largest among all the candidate routes, and $\sum_{i=1}^{k+1} \left\| p_i - e_{real,i} \right\|_2$ is still the largest among all the candidate routes. In this case, since the correct route has been output correctly up to time k, and since $\mathbf{S}(e_{real,k+1}, p_{k+1}) > 0$, $e_{real,k+1} \in near(e_{real,k})$, so **M** is updated by (4). $\mathbf{M}(e_{real,k+1}, p_{k+1})$ has the largest score. So the correct route is found by MCM at time $k+1$.

(2) $\sum_{i=1}^{k} \left\| p_i - e_{real,i} \right\|_2$ is not the largest among all the candidate routes, but $\sum_{i=1}^{k+1} \left\| p_i - e_{real,i} \right\|_2$ is the largest among all the candidate routes. In this case, an incorrect route maybe output at time k. But because $\mathbf{S}(e_{real,k}, p_k) > 0$, the correct route must be still alive at time k due to (4) and (6). Then since $e_{real,k+1} \in near(e_{real,k})$, **M** is updated by (4) and $\mathbf{M}(e_{real,k+1}, p_{k+1})$ become the new highest score matching pair. So the optimal route will be backtracked from $e_{real,k+1}$.

Finally, we prove that MCM method can complete the matching by saving the alive routes when there are outliers. We assume that the point p_{i-1} matches correctly, and $Q.last$ saves this path at this time. If the point p_i is an outlier, the HMM method will be interrupted because the transition probability matrix is 0. At point p_{i+1}, MCM method can still find topological continuous edges of p_{i-1} to complete matching through the alive route stored in $Q.last$ (equivalent to skipping p_i). □

6. MCM for offline map matching

Offline map matching is to perform map matching after all the trajectory points have been obtained.

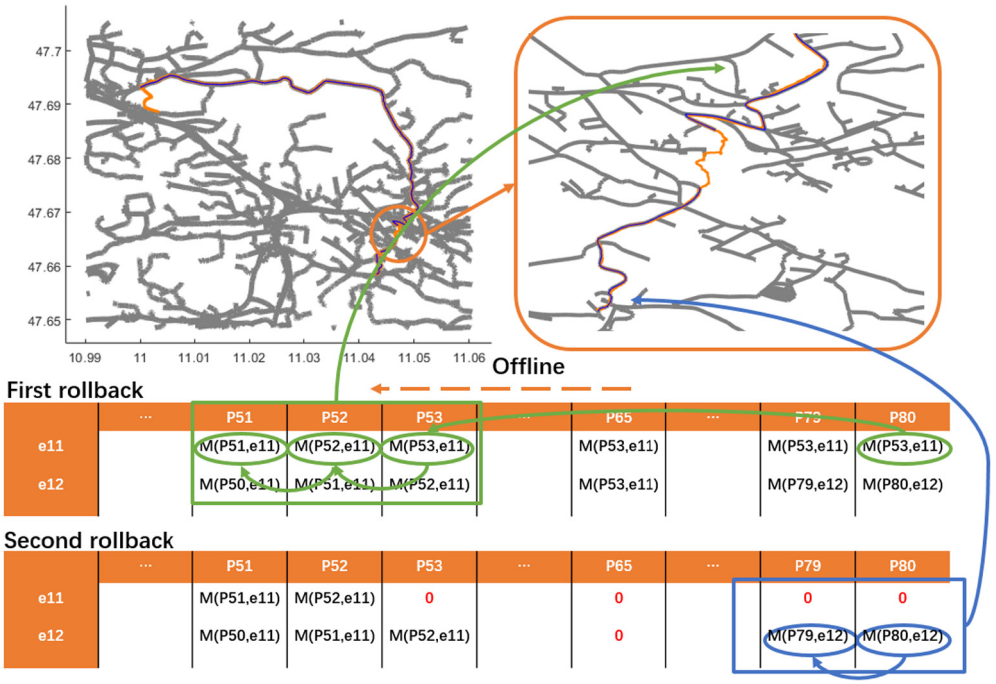


Fig. 7. There is a break point in matching process. Because of the noise, the matching break at the 53th GPS point. But the two sub-matchings are well, we need to find a method to connect the paths.

6.1. Algorithm

After collecting the whole trajectory, MCM method is used to generate the score matrix **M**. Then the best matching route is obtained by back-tracking in **M**. The offline map matching by MCM can solve the *matching break problem* which may appear in HMM-based methods. When matching breaks happen, because MCM records the matching similarities of multiple route candidates, we don't need to judge the HMM interrupt manually. That is, MCM can automatically resolve the match break problem. The method is as follows:

1. Firstly, for the trajectory **T** and the graph G, the best match route R^* is calculated according to the alignment matrix **M**. $\max_{i=[1,m]}\{\mathbf{M}(e_i, p_n)\} = \mathbf{M}(e_{end}, p_{end})$. So we get the best matching of trajectory **T** as $R^*_{[p_{begin}, p_{end}]} = \{e_{begin}, ..., e_{end}\}$. If the points on the whole trajectory have found matched edges, that is $p_{begin} == p_1 \text{and} p_{end} == p_n$, the trajectory matching is finished. Otherwise, the following recursion process is triggered to automatically find the matched sub-trajectories when there are matching breaks.
2. Recursive process. For the sub-trajectories that have not found matched routes, we repeat the following four steps for not matched sub-trajectories to find matched routes by matrix **M** as 1)-4).
   1). First, we set the matched results in **M** to 0, that is, $\mathbf{M}_{i\in[1,m], j\in[end,n]}(e_i, p_j) = 0$.
   2). Then, we segment these trajectories as $\{p_1, ..., p_{begin-1}\}, ..., \{p_{end+1}, ..., p_n\}$.
   3). Next, find the best matching routes for these sub-trajectories. For these trajectory segments, there are two situations:
      (a) The first situation is the last matching trajectory point p_{end} doesn't equal to p_n, that is, $\max_{i=[1,m]}\{\mathbf{M}(e_i, p_{begin-1})\}$ has not been calculated. Use $\max_{i=[1,m]}\{\mathbf{M}(e_i, p_{begin-1})\} = \mathbf{M}(e^{new}_{end}, p^{new}_{end})$, find out $R^*_{[p^{new}_{begin}, p^{new}_{end}]} = \{e^{new}_{begin}, ..., e^{new}_{end}\}$.
      (b) The second situation is the last matching trajectory point p_{end} equals to p_n, that is, $\max_{i=[1,m]}\{\mathbf{M}(e_i, p_{end})\}$ has been calculated. We use $\max_{i=[1,m]}\{\mathbf{M}(e_i, p_{end})\} = \mathbf{M}(e^{new}_{end}, p^{new}_{end})$, find out $R^*_{[p^{new}_{begin}, p^{new}_{end}]} = \{e^{new}_{begin}, ..., e^{new}_{end}\}$.
   4). Finally, we judge whether there are sub-trajectory segments that are not involved in the matching calculation.
3. Stop criterion. The stop criterion is that the whole trajectory is matched.

From the above method, we can see that we divide the trajectory into matched segments according to matrix **M**. By continuously iterating section by section to find the optimal matching, the matching break is avoided. The offline algorithm obtains the optimal solution.

6.2. Example

In HMM-based methods, match break appears when the GPS data is highly noisy or is missed in a period, or when the map information is incorrect. As an example shown in Fig. 7, when the map information is incorrect in the middle part of the route, the GPS trajectories cannot find a matched route, so HMM break happens [21]. When HMM break happens, the break needs to be solved by manual judgment to restart the HMM matching process until new matching pair is found.

In MCM for offline matching, we have finished matching from p_1 to p_{53}, so we change $\mathbf{M}_{i\in[1,m],j\in[53,80]}(e_i, p_j)$ to 0. And then, we find out the sub-trajectories which have not been matched as $\{p_{54} \rightarrow p_{80}\}$. We match the sub-trajectory segment $p_{54} \rightarrow p_{80}$ by $\max_{i=[1,m]}\{\mathbf{M}(e_i, p_{80})\}$ and finish matching.

6.3. Offline algorithm feasibility

We need to prove that when there are matching breaks, MCM can find the best matching method of segmented trajectories through iteration, so that the correct matching result can be obtained for the whole trajectory.

Theorem 2 *(MCM offline method). Consider there are breakpoints in the middle of the trajectory* **T**. *MCM iteration method can find the optimal result of all segments for whole trajectory.*

Proof of MCM offline method. Suppose the track length is $[1, n]$, and there are two breakpoints a and b in the middle, as $1 < a < b < n$. Now there are three situations.

(1) $\mathbf{M}[a, b] > \mathbf{M}[1, a-1]\&\mathbf{M}[b+1, n]$, i.e. $\max\limits_{i\in[1,m]} \mathbf{M}(i, n) = \mathbf{M}[a, b]$. After the first $[a, b]$ match, set the value $\mathbf{M}[a, b]$ in column n to 0, then there must be $\mathbf{M}[1, a-1]\&\mathbf{M}[b+1, n] > 0$, and the match can continue.

(2) $\mathbf{M}[1, a-1] > \mathbf{M}[a, b]\&\mathbf{M}[b+1, n]$, i.e. $\max\limits_{i\in[1,m]} \mathbf{M}(i, n) = \mathbf{M}[1, a-1]$. After the first $[1, a-1]$ match, set the value $\mathbf{M}[1, a-1]$ in column n to 0, then there must be $\mathbf{M}[a, b]\&\mathbf{M}[b+1, n] > 0$, and the match can continue.

(3) $\mathbf{M}[b+1, n] > \mathbf{M}[1, a-1]\&\mathbf{M}[a, b]$, i.e. $\max\limits_{i\in[1,m]} \mathbf{M}(i, n) = \mathbf{M}[b+1, n]$. After the first $[b+1, n]$ match, set the value $\mathbf{M}[b+1, n]$ in column n to 0, then there must be $\mathbf{M}[1, a-1]\&\mathbf{M}[a, b] > 0$, and the match can continue. □

It can be seen that the iteration will not end until the iteration condition is terminated, that is, the matching is not completed. It is guaranteed that the optimal result of all segments can be found.

7. Experiment

7.1. Datasets used in evaluation

Two datasets that are widely used in evaluating map matching methods are used in our experiments.

7.1.1. Washington dataset

The Washington Dataset presented by Newson et al. [12], is one of the most widely used benchmark data sets for testing map-matching algorithms. It contains GPS data from a drive around Seattle, WA, USA using SiRF Star III GPS chipset with WAAS (Wide Area Augmentation System) enabled. The journey was sampled at 1 Hz and contains just over two hours of driving in both challenging inner-city environments and the outer suburbs. The total route was 80 km long with 7531 data samples containing latitude and longitude pairs. Table 4 shows an overview of this dataset. Further details about this dataset can be found in [12].

7.1.2. Global dataset

Tracks featured in the global dataset originate from a publicly available collection called Planet GPX [36]. This dataset is part of the OpenStreetMap project [37]. Volunteers worldwide collected it over nine years for automated route shaping and turn restriction detection in the OpenStreetMap. The current version of the global dataset contains a selection of 100 records. This collection has 73 tracks with gaps, 25 tracks with “U”-turns, 24 tracks with loops, 3 tracks with hives, and 20 records with severe congruence issues. As mentioned above, the track lengths are limited in range from 5 to 100 kilometers. All tracks have the same sampling rate of 1 Hz. In total, the dataset contains 247,251 points and 2,695 kilometers of tracks. Table 5 shows an overview of this dataset. Further details about this dataset can be found in [38].

8. Results

8.1. Washington dataset results

8.1.1. Parameter selection

Fig. 8(a) shows the relationship between the map matching efficiency and the parameter selection in online matching. The point-to-road similarity threshold varies from 5 to 40. The running time increases with ε. Even the maximum running

Table 4
Washington Dataset Summary.

Metric	Data Value
Total number of journeys	1
Total kilometers of journeys	80
Sampling Rate	1 Hz
Total Samples	7531

Table 5
Global Dataset Summary.

Metric	Data Value
Total number of journeys	100
Total kilometers of journeys	2,695
Sampling Rate	1 Hz
Total Samples	247,251

Table 6
Online run-time of our proposed method with Washington dataset when $\varepsilon = 30$.

Average run-time	Maximum run-time	Minimum run-time
0.0482 s	0.0731 s	0.0057 s

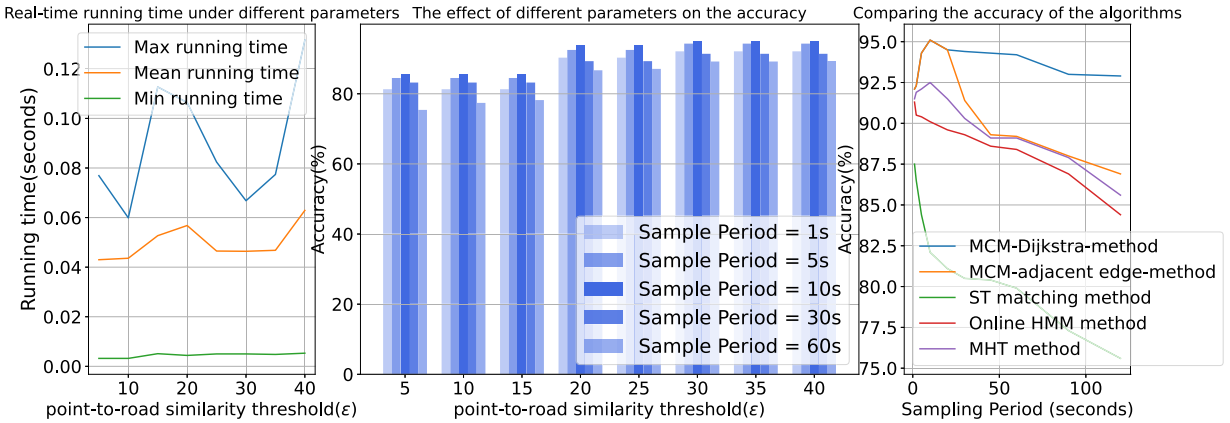


Fig. 8. (a) Efficiency under different ε for Washington dataset. (b) Different ε and different GPS sampling rate v.s. matching accuracy (% of points matched to correct road segment) for Washington dataset. (c) MCM vs HMM based method for Washington dataset.

time is far less than 1 second (on a laptop with Intel i7-8550U CPU), so it can well support online map matching, which generally takes GPS data at 1 Hz. Fig. 8(b) shows the accuracy of MCM for various sampling intervals and various threshold parameters. The horizontal axis represents the similarity threshold, the bar colors represent the sampling intervals, and the vertical axis represents the accuracy. Obviously, similarity threshold can help to improve the matching accuracy.

8.1.2. Accuracy comparison with other methods

In this paper, we hope to use road information as little as possible to make the MCM method suitable for different cases. So for performance comparison, we choose ST matching [7], Online HMM [8] and MHT [19] as the comparative methods.

Fig. 8(c) illustrates that the matching accuracy is the highest when the sampling rate is 10 Hz. The term 'accuracy' in this context refers to the percentage of GPS points matched to the correct segment of the road network. When the sampling rate decreases, the matching accuracy will decrease because the previous GPS points provide less information about the current GPS point. When the sampling rate is too high, unnecessary detour [21] will appear, and the accuracy of matching will also be reduced. So we must balance various factors and choose the best matching sampling rate. In the Washington data set, 10 hz is the best choice, and we choose $\varepsilon = 30$ (Table 6).

From the matching accuracy comparison in Fig. 8(c), we can see that the matching accuracy of MCM is 95%, which is better than all other three methods with a high sampling rate. With a low sampling rate, all online methods get less accuracy, under 90%. But our MCM method still performs better than the Online-HMM method and ST matching method. MHT is most close to MCM, and they are always better than the other two methods (Fig. 9). MCM also uses less information than MHT, which is, therefore, highly suitable for online map matching.

Fig. 9. Result for Washington dataset. Here is an overpass scene, we can see that our method can match the side road well, while other methods will match wrong.

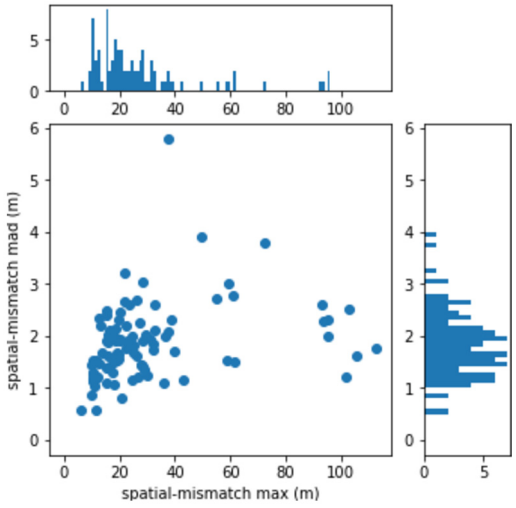


Fig. 10. The distribution of data in the global dataset.

8.2. Global dataset results

8.2.1. Dataset features and parameter selection

The distribution of data in the global dataset is shown in Fig. 10. It gives information about the spatial mismatch between the GPS points and their real routes. An attribute 'mad' is the median absolute deviation, and the attribute 'max' is the peak distance between the track and the route. In the dataset, the maximum spatial mismatch can be up to 100 meters. In order to obtain a consistent estimator of standard deviation, one has to take $\sigma = K * mad$ where K is a scale parameter that depends on the probability distribution function. For normal distribution $K \approx 1.4826$, for exponential distribution $K \approx 2.0781$ and for Rayleigh distribution $K \approx 2.230$. These distributions are often used in the literature to characterize spatial mismatch.

We can conclude from the previous section that as long as the choice of parameter ε can cover the point-to-road errors, the mismatch can be considered by MCM. The larger the parameter ε is selected, the more candidate routes will be calculated, and the lower the calculation efficiency is. So we don't consider the trajectory outliers and just choose the value $\varepsilon = 30$ that covers most of the errors.

8.2.2. Solving matching break in offline matching

The matching break is a common problem in map-matching, which is mainly caused by trajectory outliers. This happens more frequently in the HMM matching model when the correct state falls out of the candidate range of the outlier. Currently, most of the solutions [39] try to overcome this problem by identifying and removing the outliers to remedy the broken route. We apply HMM map matching method on the Global dataset with random down-sample and trajectory compression

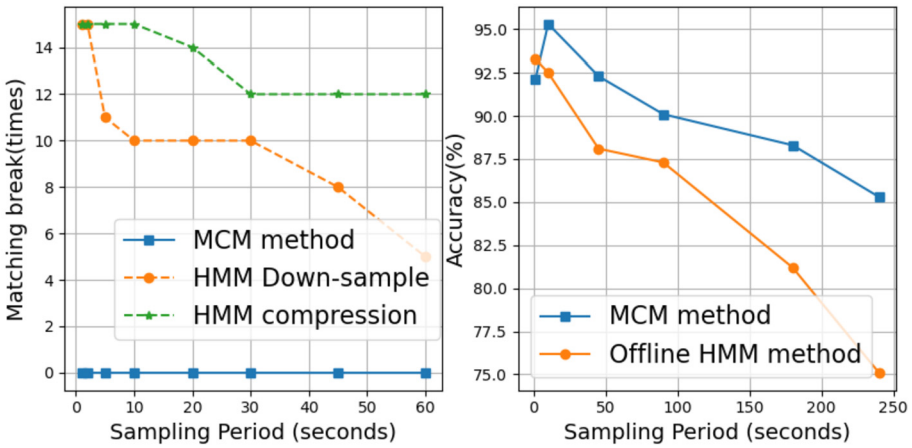


Fig. 11. (a) Matching break by random down-sample and trajectory compression with HMM method and proposed method. (b) Proposed method vs HMM based method: GPS sampling rate vs accuracy (% of points matched to correct road segment) for Global dataset.

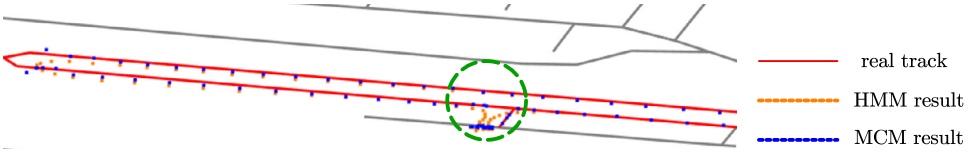


Fig. 12. Result for U-turns. We seek to map points to the right road, and at the same time, we can map points to the road in the right direction.

(Douglas-Peucker algorithm), respectively. In Fig. 11(a), the result shows that trajectory compression fails to prune outliers as this method is the least effective of all methods and has the most number of matching breaks. This is because outliers are usually preserved as outstanding points, which means more preprocessing step is required to remove such outliers. Compared with trajectory compression, the method of down-sampling the trajectory can sample uniformly and has a certain probability of discarding outliers. So the performance is better, but it cannot completely solve the problem of matching interruption. However, our method solves the match break problem as shown in Fig. 11(a).

Fig. 11(b) shows the accuracy of the methods for various sampling intervals. The horizontal axis represents the sampling interval, and the vertical axis represents accuracy. Fig. 11(b) illustrates the accuracy is the highest when the sampling rate is 10 Hz.

In the experiment, we used Newson method [12] as the offline HMM method, which is a major HMM-based method. As shown in Fig. 11(b), our MCM method has a 5%-10% better matching effect than the offline HMM method. And even with a lower sampling rate, our MCM method has an accuracy above 85%.

8.2.3. Solving map matching problems in complex cases

Our algorithm can also perfectly solve the map matching problem in the following complex cases.

(1) U-turns: the vehicle turned around in the middle of the street. We hope that the vehicle can be matched to the correct road (a two-way road that allows U-turns or one-way intersections in the opposite direction). As shown in Fig. 12, our algorithm can give the correct matching result for U-turns. The red track represents the true track. The offline HMM method represented by the orange track makes a U-turn in the middle of the disconnected one-way street, while our method completes the U-turn at the intersection, which is more in line with the actual situation.
(2) hives: a large volume of GPS points packed in a small area. In this case, many noise points are often collected and cannot correspond to the road. We hope the noise points can be filtered out and do not correspond to the wrong road. As shown in Fig. 13, these noise points are not mismatched in our algorithm. No pre-operation is required; they are discarded in the matching process.
(3) loops: the vehicle was traveling in circles. As shown in Fig. 14, in offline HMM, the probability of transition matrix for loop road segments is the same, so if a track is equidistant from multiple roads due to noise, it is difficult to complete the correct matching. And our algorithm can perfectly solve this problem through global matching.

Table 7 shows the average run-time of the map matching algorithms across both trials. We used an Intel i7-8550U CPU @ 1.80 GHz with 16GB of DDR3 RAM to perform the run-time analysis. It can be seen that the proposed approach offers a run-time reduction of over 68% using this data set. Furthermore, this has been made possible without creating a large table of pre-computed distances, which keeps the memory usage of the algorithm minimal.

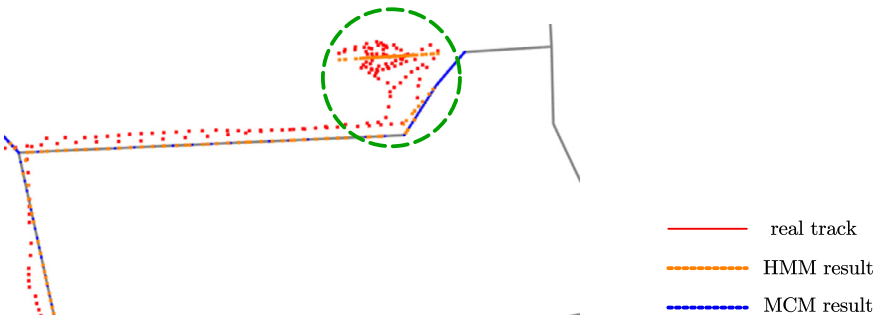


Fig. 13. Result for hives. The noise points are not mismatched.

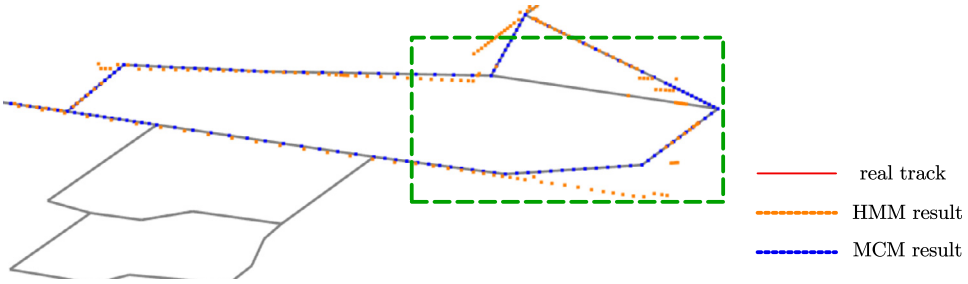


Fig. 14. Result for loops. MCM method can complete the correct matching in loop situation.

Table 7
Total average run-time of HMM method and our proposed method with Global dataset.

Offline HMM method	Proposed method	Reduction
171.31 s	54.73 s	68.05%

9. Conclusion and future work

In this paper, we propose a new method, i.e., MCM for online map matching. MCM tracks multiple alive route candidates while controlling the scale of candidates according to the continuity of the road by excluding unnecessary matching candidates. In offline matching, in the backtracking process, we use an iterative method to avoid match breaks. Extensive evaluations are conducted to verify the effectiveness of MCM. Through the Washington dataset, we verify the feasibility of MCM in online map matching. Through the Global data set, we verify the matching accuracy and the influence of sampling rate on our method. MCM does not need to offline train the transition probability. It needs to set a threshold to specify the maximum acceptable offset. MCM works well without the pain of preliminary data processing work while providing better robustness and accuracy. In future work, we can also modify the road continuity function, such as introducing semantic information to better distinguish overpass sections, and conduct road navigation while conducting map matching.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Yongcai Wang reports financial support was provided by National Natural Science Foundation of China, 61972404, 12071478.

References

[1] D.L.X.X. Li Wanting, Yongcai Wang, Mcm: a robust map matching method by tracking multiple road candidates, in: Algorithmic Aspects in Information and Management, Springer, 2022.
[2] Y. Cui, S.S. Ge, Autonomous vehicle positioning with gps in urban canyon environments, IEEE Trans. Robot. Autom. 19 (2003) 15–25.
[3] R. Chaggara, C. Macabiau, E. Chatre, Using gps multicorrelator receivers for multipath parameters estimation, in: Proceedings of the 15th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GPS 2002), 2002, pp. 477–492.
[4] M. Spangenberg, A. Giremus, P. Poire, J.-Y. Tourneret, Multipath Estimation in the Global Positioning System for Multicorrelator Receivers, 2007 IEEE International Conference on Acoustics, Speech and Signal Processing-ICASSP'07, vol. 3, IEEE, 2007, pp. III–1277.
[5] N. Blanco-Delgado, F.D. Nunes, Multipath estimation in multicorrelator gnss receivers using the maximum likelihood principle, IEEE Trans. Aerosp. Electron. Syst. 48 (2012) 3222–3233.
[6] D. Bernstein, A. Kornhauser, et al., An Introduction to Map Matching for Personal Navigation Assistants, 1996.

[7] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, Y. Huang, Map-matching for low-sampling-rate gps trajectories, in: Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, 2009, pp. 352–361.
[8] C.Y. Goh, J. Dauwels, N. Mitrovic, M.T. Asif, A. Oran, P. Jaillet, Online map-matching based on hidden markov model for real-time traffic sensing applications, in: 2012 15th International IEEE Conference on Intelligent Transportation Systems, IEEE, 2012, pp. 776–781.
[9] R. Mohamed, H. Aly, M. Youssef, Accurate real-time map matching for challenging environments, IEEE Trans. Intell. Transp. Syst. 18 (2016) 847–857.
[10] G.R. Jagadeesh, T. Srikanthan, Online map-matching of noisy and sparse location data with hidden markov and route choice models, IEEE Trans. Intell. Transp. Syst. 18 (2017) 2423–2434.
[11] L. Luo, X. Hou, W. Cai, B. Guo, Incremental route inference from low-sampling gps data: an opportunistic approach to online map matching, Inf. Sci. 512 (2020) 1407–1423.
[12] P. Newson, J. Krumm, Hidden markov map matching through noise and sparseness, in: Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, 2009, pp. 336–343.
[13] J. Yuan, Y. Zheng, C. Zhang, X. Xie, G.-Z. Sun, An interactive-voting based map matching algorithm, in: 2010 Eleventh International Conference on Mobile Data Management, IEEE, 2010, pp. 43–52.
[14] Y. Li, Q. Huang, M. Kerber, L. Zhang, L. Guibas, Large-scale joint map matching of gps traces, in: Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, 2013, pp. 214–223.
[15] Z. Zeng, T. Zhang, Q. Li, Z. Wu, H. Zou, C. Gao, Curvedness feature constrained map matching for low-frequency probe vehicle data, Int. J. Geogr. Inf. Sci. 30 (2016) 660–690.
[16] H. Wei, Y. Wang, G. Forman, Y. Zhu, H. Guan, Fast viterbi map matching with tunable weight functions, in: Proceedings of the 20th International Conference on Advances in Geographic Information Systems, 2012, pp. 613–616.
[17] B.Y. Chen, H. Yuan, Q. Li, W.H. Lam, S.-L. Shaw, K. Yan, Map-matching algorithm for large-scale low-frequency floating car data, Int. J. Geogr. Inf. Sci. 28 (2014) 22–38.
[18] M. Dogramadzi, A. Khan, Accelerated map matching for gps trajectories, IEEE Trans. Intell. Transp. Syst. (2021).
[19] S. Taguchi, S. Koide, T. Yoshimura, Online map matching with route prediction, IEEE Trans. Intell. Transp. Syst. 20 (2018) 338–347.
[20] G. Li, L. Lou, P. Zheng, et al., Route restoration method for sparse taxi gps trajectory based on bayesian network, Teh. Vjesn. - Stroj. Fak. 28 (2021) 668–677.
[21] P. Chao, Y. Xu, W. Hua, X. Zhou, A survey on map-matching algorithms, in: Australasian Database Conference, Springer, 2020, pp. 121–133.
[22] J.L. Bentley, H.A. Maurer, Efficient worst-case data structures for range searching, Acta Inform. 13 (1980) 155–168.
[23] C.E. White, D. Bernstein, A.L. Kornhauser, Some map matching algorithms for personal navigation assistants, Transp. Res., Part C, Emerg. Technol. 8 (2000) 91–108.
[24] S.S. Saab, A map matching approach for train positioning. i. Development and analysis, IEEE Trans. Veh. Technol. 49 (2000) 467–475.
[25] G. Taylor, G. Blewitt, D. Steup, S. Corbett, A. Car, Road reduction filtering for gps-gis navigation, Trans. GIS 5 (2001) 193–207.
[26] S. Brakatsoulas, D. Pfoser, R. Salas, C. Wenk, On map-matching vehicle tracking data, in: Proceedings of the 31st International Conference on Very Large Data Bases, 2005, pp. 853–864.
[27] H. Yin, O. Wolfson, A weight-based map matching method in moving objects databases, in: Proceedings. 16th International Conference on Scientific and Statistical Database Management, 2004, IEEE, 2004, pp. 437–438.
[28] H. Wei, Y. Wang, G. Forman, Y. Zhu, Map matching by fréchet distance and global weight optimization, Technical Paper, Department of Computer Science and Engineering, 2013, p. 19.
[29] L. Zhu, J.R. Holden, J.D. Gonder, Trajectory segmentation map-matching approach for large-scale, high-resolution gps data, Transp. Res. Rec. 2645 (2017) 67–75.
[30] R. Song, W. Lu, W. Sun, Y. Huang, C. Chen, Quick map matching using multi-core cpus, in: Proceedings of the 20th International Conference on Advances in Geographic Information Systems, 2012, pp. 605–608.
[31] M.M. Atia, A.R. Hilal, C. Stellings, E. Hartwell, J. Toonstra, W.B. Miners, O.A. Basir, A low-cost lane-determination system using gnss/imu fusion and hmm-based multistage map matching, IEEE Trans. Intell. Transp. Syst. 18 (2017) 3027–3037.
[32] L. Bonnetain, A. Furno, J. Krug, N.-E.E. Faouzi, Can we map-match individual cellular network signaling trajectories in urban environments? data-driven study, Transp. Res. Rec. 2673 (2019) 74–88.
[33] C. Song, X. Yan, N. Stephen, A.A. Khan, Hidden markov model and driver path preference for floating car trajectory map matching, IET Intell. Transp. Syst. 12 (2018) 1433–1441.
[34] T. Feng, H.J. Timmermans, Map matching of gps data with bayesian belief networks, J. East. Asia Soc. Transport. Stud. 10 (2013) 100–112.
[35] X. Wang, W. Ni, An improved particle filter and its application to an ins/gps integrated navigation system in a serious noisy scenario, Meas. Sci. Technol. 27 (2016) 095005.
[36] P.G. contributors, Planet gpx, http://planet.openstreetmap.org/gps/, 2009.
[37] O. contributors, Openstreetmap, http://openstreetmap.org/, 2009.
[38] M. Kubička, A. Cela, P. Moulin, H. Mounier, S.-I. Niculescu, Dataset for testing and training of map-matching algorithms, in: 2015 IEEE Intelligent Vehicles Symposium (IV), IEEE, 2015, pp. 1088–1093.
[39] H. Wu, W. Sun, B. Zheng, Is only one gps position sufficient to locate you to the road network accurately?, in: Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing, 2016, pp. 740–751.