Incremental route inference from low-sampling GPS data: an opportunistic approach to online map matching

Linbo Luo[a,*], Xiangting Hou[b,c], Wentong Cai[c], Bin Guo[d]

[a]*School of Cyber Engineering, Xidian University, Xi'an, China*
[b]*Complexity Institute, Interdisciplinary Graduate School, Nanyang Technological University, Singapore*
[c] *School of Computer Science and Engineering, Nanyang Technological University, Singapore*
[d]*School of Computer Science, Northwestern Polytechnical University, Xi'an, China*

Abstract

With the surging of smart device sensing and mobile networking, GPS data has been widely available for identifying vehicle position and route on the road map. For many real-time applications, such as traffic sensing and route recommendation, it is critical to immediately infer travelling route with incoming GPS data. In this paper, an opportunistic approach to online map matching is proposed to incrementally infer routes from low-sampling GPS data with low output latency. Unlike the hidden Markov model (HMM)-based approach, which often experiences certain delay between the GPS observation and inference, our algorithm can produce immediate inference when a new GPS point becomes available. Furthermore, a rollback mechanism is provided to correct the already inferred route when some abnormal situations are detected during the opportunistic inference process. We evaluate the proposed algorithm using real dataset of GPS trajectories over 100 cities around the world. Experimental results show that our algorithm is better than, or at least comparable to the state-of-the-art algorithms in terms of inference accuracy. More importantly, our algorithm can yield much shorter output latency and require less execution time, which is critical for many real-time navigation applications and location-based services.

Keywords: online map matching; GPS data analysis; trajectory mining

1. Introduction

Map matching is an important technique to infer the travelling route of a vehicle from a sequence of observed GPS data. It is a critical data processing step for many intelligent transport and smart city applications. Map matching problem has long been studied by both computer science and transportation research communities. However, traditional map matching algorithms are mostly designed in an offline context (i.e., an entire sequence of observed GPS points is available for inference) and the GPS data is usually collected under high sampling rate. Recently, however, there is an increasing need to perform map matching for real-time navigation and location-based services, such as route recommendation. This calls for the map matching algorithm to return the matching results immediately with incoming samples. Furthermore, due to the requirements of online transmission, energy consumption and storage, GPS data has to be collected in a low-sampling rate (e.g., one sample per minute). This introduces many uncertainties for inferring the traveling route from these GPS data. Therefore, online map matching with low-sampling GPS data has become a very challenging task.

The key difference of online map matching from offline one is the unavailability of future GPS data when inferring the route for the current GPS point. This makes it impossible to leverage the global knowledge gained from a complete GPS sequence. To alleviate this problem, hidden Markov model (HMM)-based approaches [8, 17, 23, 37] for online map matching have been proposed, in which the matching process has to be delayed by several sampling intervals so that a reliable route up to a certain GPS point (usually not the current one) can be inferred by leveraging the information of subsequent GPS points. In these methods, a latent sequence of road segments that estimates the route up to a given GPS point can be obtained based on some convergence conditions (e.g., online Viterbi decoder in [8]). Such conservative matching process helps the GPS data being well analyzed so as to ensure the accuracy of inference.

Despite its popularity, HMM-based approaches for online map matching still

have some limitations. Firstly, the algorithms conservatively generate the inferred route only after certain convergence condition (e.g., detection of convergent point in [8]) is met. Therefore, the time gap between the GPS observation and inference always exists. We refer to such time gap as *output latency*, which is the time period between the time when the latest observed GPS point is generated and the time when the corresponding route is inferred. Secondly, for each pair of two consecutive GPS points, the algorithms usually require the calculation of all possible routes between the two points for local route inference [39]. This can introducing an increasing computational cost when analyzing complex and large-scale road network.

With the consideration of the aforementioned limitations of existing approach, we present an opportunistic approach to online map matching that can incrementally infer vehicle's trajectories from low-sampling GPS data in real-time and have shorter output latency compared to HMM-based approaches. The main idea is that instead of conservatively inferring the route with certain delay, the route up to the current GPS point is always obtained using a best-effort method. In cases that opportunistic inference strategy is not successful, the inferred route is reexamined and corrected by a rollback mechanism. To this end, an incremental route inference algorithm with rollback (INC-RB) is proposed. This paper extends from our previous work [13] where the core idea of our opportunistic inference is outlined. In this paper, we provide a more systematic depiction and refined design of our INC-RB. A new rollback condition that considers abnormal driving behaviors is added in the rollback mechanism of INC-RB. More evaluations are conducted with respect to inference accuracy, output latency and execution efficiency. In addition, the performance of our rollback mechanism is analyzed with more details and the limitations of our approach is also discussed.

The key contributions of this paper are summarized as follows.

1. We advocate an opportunistic approach to online map matching, which is capable of generating an immediate inference route up to the latest GPS

observation. Such an approach can help to reduce the output latency when comparing with the conservative approach as in HMM-based methods. Furthermore, in the local inference process of our approach, it is not necessary to explore all the possible routes, which can help to reduce the search space as well as the number of times to perform the local route inferences.

2. A rollback mechanism is proposed to complement the opportunistic inference process to ensure the accuracy of the matching results. A set of rollback conditions are designed for detecting the abnormal situations where the opportunistic approach is considered to be not successful and a rollback correction procedure is proposed for correcting the already inferred route.

3. Extensive evaluations of the proposed INC-RB are conducted using a large-scale real dataset of GPS trajectories under different sampling rates. Our INC-RB are compared with three state-of-the-art online map matching algorithms in terms of inference accuracy, output latency and execution efficiency. The results show that INC-RB is better than, or at least comparable to the compared algorithms with respect to inference accuracy. Moreover, our INC-RB has much shorter output latency and requires less execution time thanks to our opportunistic inference strategy.

The remainder of this paper is organized as follows. Section 2 provides the related work about the existing map matching strategies. Section 3 gives the problem statement, basic workflow of online map matching and explanation of conservative strategy in HMM-based method. Section 4 presents the overall framework of the proposed algorithm. In Section 5, the detailed design of our algorithm is described. Section 6 discusses the experiment design, results and limitations. In section 7, we conclude the paper and discuss the future work.

2. Related work

Depending on the amount of GPS data that are available when map matching is performed, the existing work can be generally grouped into two classes: *offline* (or called global, post-processing) map matching [22, 32] and *online* (or called incremental, real-time) map matching [11]. An offline algorithm assumes a complete sequence of the observed GPS points is available and it infers the entire traveling route at one go with the global knowledge of all GPS points. In contrast, an online algorithm can only leverage the information of GPS points up to the latest one and it produces the partial inferred route in real time. In general, the offline algorithms are more accurate, whereas the online algorithms can return the matching results faster [5].

Early research on map matching mainly focuses on the development of offline algorithms. In the earliest stage, the offline map matching mainly relies on the geometrical analysis [3, 20, 30, 38], such as the closeness and shape of road segments. Later, topological information of road networks (e.g., the connectivity between road segments) is also taken into account [5, 10, 33]. Although the geometrical and topological approaches work well with high-sampling GPS data, their performances degrade when inaccurate and low-sampling data are presented. To deal with high uncertainty due to the noise and sparseness, the mathematical and probabilistic models such as Kalman Filter [28], fuzzy logic [31], belief theory [7, 26], Conditional Random Field (CRF) [16, 24], Hidden Markov model (HMM) [25, 27, 29, 40, 41], force-directed graph drawing [34] and ant colony optimization (ACO) [9] are then introduced in conjunction with geometrical, topological and directional analysis. In particular, the HMM-based model has gained popularity in recent years and it is shown to be the-state-of-the-art for offline map matching with noisy and sparse GPS data [18].

Compared to offline map matching, research on online map matching is still in its infancy. The common approaches for online map matching are also based on the HMM model. The pioneer work on applying HMM for online map matching is proposed by Goh et al [8], in which emission probability is used for matching

probable road segment for each GPS point and transition probability is used for inferring the route connecting the matched segments of two consecutive GPS points in low-sampling rate settings. In particular, the Viterbi algorithm of HMM is extended into an online version that can infer the most likely partial route up to a convergence point and a bounded variable sliding window (BVSW) method is proposed to detect such convergence point. One drawback of the method in [8] is that there is no guarantee when and whether such convergence point can be found and it therefore introduces inference delay. The subsequent research mostly follows the HMM paradigm proposed in [8]. Wang and Zimmermann [37] propose the statistics-based online map matching algorithm called *Eddy* with an improved online Viterbi decoder to ensure the bounded delay and bounded error. Liang et al. [23] propose the online learning map matching (OLMM) algorithm that incorporates an online learning process to dynamically estimate the standard deviation of GPS error for improving matching accuracy. Jagadeesh and Srikanthan [17] present several heuristic optimizations to prune the unlikely states and reduce the shortest-path search space for improving the efficiency of HMM. In [18], Jagadeesh and Srikanthan combine the HMM model with a route choice model to reassess the HMM-generated partial route for accuracy improvement. Kang et al. [19] extends the HMM-based model by introducing constraints based on motion laws (e.g., speed limitation). In [39], Yin et al. propose a feature-based model for determining transition probability in HMM and the model can be applied to both offline and online map matching.

The above-mentioned online map matching works extend the HMM-based model in [8] with objective of improving either accuracy or efficiency. But, they may achieve one objective while sacrificing the other. For example, in [18], the reassessment of HMM inferred routes by a route choice model inevitably introduces further computational cost in the entire map matching process. Moreover, all these works still require the detection of the convergence point as in [8]. This implies that it can only obtain the partial inferred route up to the convergence point that may be far from the latest GPS observation.

With the aim of achieving both accuracy and efficiency in real-time, Hashemi

and Karimi [12] propose a weight-based map matching algorithm without resorting to advanced probabilistic models such as HMM. The proposed algorithm identifies the best segment for a GPS point by considering three topological criteria and adopts a novel approach to dynamically determine the weights of these criteria when new GPS comes. A confidence level for each assigned segment is determined to indicate the confidence on the solution. The algorithm has shown superior performance for complex urban environment. However, since the algorithm solely focuses on the task of matching probable road segment for each GPS point, it is mainly applicable for high-sampling rate GPS data in navigation applications.

It is also worth noting that the information fusion approach, which leverages both GPS data and auxiliary sensing data to improve map matching accuracy, has recently gained popularity. For instance, Hu et al. [15] propose a map matching algorithm that supports spatial-temporal analysis based on additional speed and direction information together with positional information of GPS data. In [1], the data collected from inertial sensors of mobile phone is used to detect road semantics for improving HMM-based model. Atia et al. [2] fuse the data from inertial sensors with GPS data using extended Kalman filter and propose a HMM-based map matching algorithm with a least-square regression step to estimate vehicle's position at lane level. Vehicle heading direction data is also considered as an important source of information to improve GPS-only map matching and it has been utilized in both offline map matching [14] and online map matching [6] respectively. In [35], Taguchi et al. propose an online map matching algorithm with a probabilistic route prediction model for reducing output latency. The algorithm leverages the historical information on vehicle's velocity to train model parameters. This class of methods has shown the benefits of capturing motional characteristics of vehicle as a new dimension for route inference. But, it is subject to the equipment of additional sensors, which may not be always available in difference scenarios.

In this work, we focus on online map matching for low-sampling GPS data with the aim of achieving a balanced tradeoff between accuracy and efficiency.

To ensure generality of the proposed approach, we assume only the positional GPS data and road network information are available during online inference process and no pre-training of inference model using historical data is needed. Unlike the HMM paradigm that we refer to as the conservative approach (see further explanation in section 3.3), we adopt an opportunistic approach that is capable of producing the partial inferred route up to the latest GPS point in most of the cases during online inference. A rollback mechanism is introduced as a complementary measure when such opportunistic approach does not succeed.

3. Preliminaries

In this section, we firstly state our problem with a set of definitions related to our problem. Secondly, we briefly explain the basic overflow of online map matching process. Lastly, we describe the key idea of the conservative approach adopted by the existing HMM-based online map matching algorithms.

3.1. Problem statement

We first introduce the definitions necessary for the development of our algorithm, and state our problem.

Definition 1 (GPS point): A GPS point g_i contains the spatial-temporal information of a vehicle. Each point is associated with longitude and latitude coordinates with a timestamp as: $g_i =< g_i.lat, g_i.lon, g_i.time >$.

Definition 2 (Partial GPS trajectory): In a live streaming environment of data collection, a partial GPS trajectory T_i is a sequence of GPS points $T_i = \{g_0 \rightarrow g_1 \rightarrow ... \rightarrow g_i\}$ up to the current time (assuming, $g_i.time = time\_current$).

Definition 3 (Road segment): A road segment s is a directed edge $s =< s.start, s.end, s.length >$ represented by one start node (i.e., $s.start$), one end node (i.e., $s.end$), and a length value (i.e., $s.length$).

Definition 4 (Road network): A road network is a directed graph $G(V, E)$, where V is the vertex set representing the intersections of the road segments and E is the edge set representing the road segments.

Definition 5 (Route): A route l is a sequence of connected road segments.

Definition 6 (Inferred partial route): A inferred partial route L_i is the route that a vehicle is believed to travel given the observation of a partial GPS trajectory T_i.

Our problem: When a new GPS point g_i becomes available, we aim to produce the most probable inferred partial route L_i given the information of partial GPS trajectory T_i and the underlying road network G.

3.2. Basic workflow of online map matching

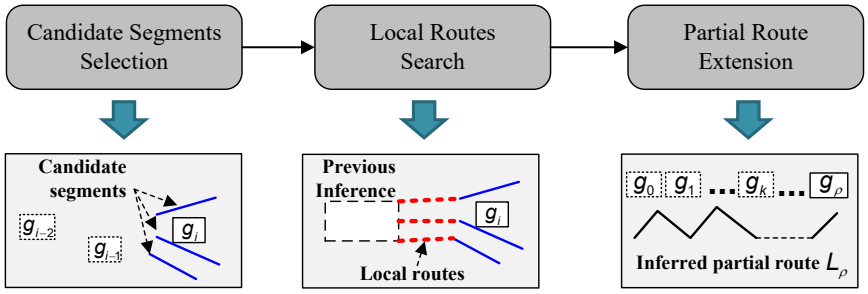


Figure 1: Workflow of online map matching

Despite different algorithms being developed, the basic workflow of online map matching can be generally divided into three tasks: *candidate segments selection*, *local routes search* and *partial route extension*. As shown in Figure 1, when a new GPS point g_i becomes available, the first task is to select a set of candidate road segments that g_i most likely belongs to. The candidate segments are usually selected with the consideration of measurement errors of GPS points which are commonly modeled as the Gaussian distribution. Next, between the current GPS point g_i and the previous GPS point g_{i-1}, there could exist many possible routes that a vehicle can travel. The second task is thus to search the most probable local routes between g_{i-1} and g_i. Specifically, each local route starts from one candidate segment of g_{i-1} and ends at one candidate segment of g_i. Last, since the online map matching is executed in an incremental manner, we assume there has already existed an inferred partial route L_k up to certain

GPS point g_k where $k \leq i-1$. The third task is to extend L_k further given the inference information obtained from the first and second tasks. We refer the inferred partial route after extension as L_ρ. In most existing HMM-based online map matching algorithms, the index ρ is usually less than the current GPS point index i. That is, there exists a delay between the GPS observation and route inference. The reason for this will be elaborated in the next subsection.

3.3. Conservative approach in HMM-based algorithms

In the existing HMM-based online map matching algorithms (e.g., [8, 18, 23]), the online Viterbi algorithm is used for the task of partial route extension. In online Viterbi algorithm, the inferred partial route is only extended when a convergence point is detected. Fig 2 illustrates the convergence point detection process. When a new GPS point g_i is added, three candidate segments (i.e., s_i^1 to s_i^3) are mapped for g_i. The local routes (i.e., the black dashed lines) between g_{i-1} and g_i are then generated. Among all the candidate local routes, the routes shown as the black arrow lines are the inferred local routes (i.e., the black solid lines) which represent the most probable local routes ending at s_i^1 to s_i^3 respectively. Then, the online Viterbi algorithm traces backward to check for a convergence point where all the inferred local routes of s_i^1 to s_i^3 converge to this segment. If such convergence point (e.g., s_{i-2}^3 as shown in Fig 2) is found, the inferred partial route up to the convergence point can be determined by selecting the series of the inferred local routes (i.e., the red lines) that end at that convergent point. It is also possible that such convergence point cannot be founded at the current step. In such a case, the extension of inferred partial route is postponed until a convergent point is found in the subsequent inferences with new GPS data.

As illustrated above, due to the nature that online map matching cannot obtain all the GPS observations at one time, the HMM-based models can only generate inferred partial route that may not corresponds to the partial GPS trajectory up to the latest GPS point. The partial route extension based on convergent point detection can thus be considered as a conservative approach. In

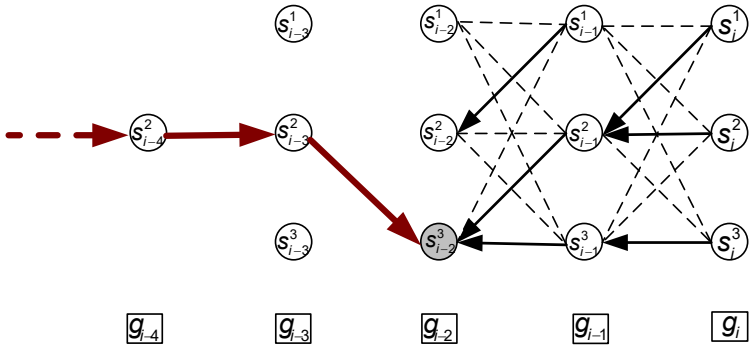


Figure 2: Illustration of conservative strategy

such an approach, the best case is that the convergence point is one of candidate road segments of g_{i-1} given a new GPS point g_i is added. Hence, the inferred partial route can only be obtained at most up to g_{i-1}. Therefore, the output latency always exists. Moreover, there is no guarantee that the convergence point can be always found at the current step, which further delays the inference process.

4. Framework overview

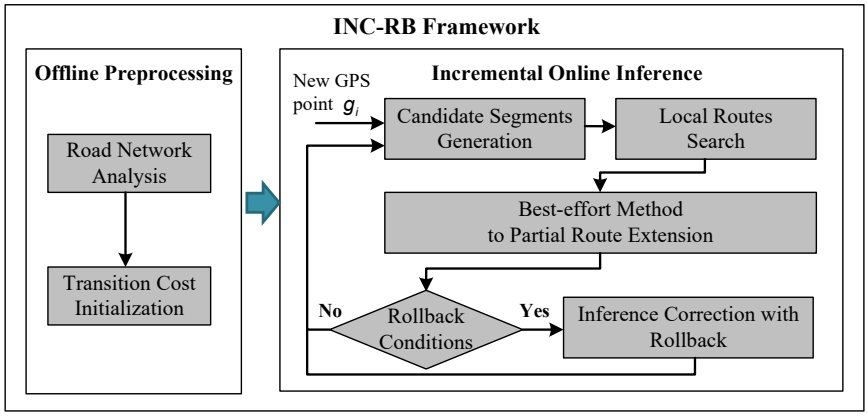


Figure 3: Framework overview

The overall framework of our online map matching process is shown in Fig. 3. The entire process generally includes two stages: offline preprocessing and incremental online inference. A *road network analysis* and *transition cost initialization* module are introduced at the offline preprocessing stage for speeding up the online inference. In the road network analysis module, the network data in a given area of interest is first extracted and the road network information is then analyzed, aiming to extract road features such as length, turning angle, speed limit, etc. The transition cost initialization module is responsible for assigning the initial cost of the nodes in the extracted network based on the corresponding road features. These transition cost values will be utilized for determining the transition probability of local routes in the incremental online inference stage.

The incremental online inference in our framework follows the basic workflow of online map matching as outlined in section 3.2 with an extension of a rollback mechanism for inference correction. The *candidate segments generation* module is responsible for generating the candidate segments of the new GPS point, when a new GPS point is available. The *local routes search* module is then responsible for searching a set of local routes, each of which connects a candidate segment of the new GPS point to the last road segment in the previous inferred partial route. Next, we adopt a *best-effort method to partial route extension* that aims to produce inferred partial route up to the newly added GPS point. During the online inference process, our map matching process also checks a set of rollback conditions that examine some abnormal situations (e.g., no local route is found). If any of such rollback conditions is detected, a rollback mechanism to correct the pervious inferred partial route is performed to ensure the quality of inference. Otherwise, our map matching process will proceed to next round of incremental online inference once the next GSP point is available.

5. Detailed design of map matching process

5.1. Offline preprocessing

The purpose of offline preprocessing is to perform some pre-calculations that are needed by the incremental online inference in the next stage. The analyses performed in this stage are mainly based on the road network information that are available offline. The two main tasks are the road network analysis and transition cost initialization which are described as follows:

Road network analysis: To infer travelling routes efficiently, the area in a given road network that is of interest of the study is first extracted. The network area extraction is performed on the OpenStreetMap network data where the road segments are represented in the form of ESRI Shapefile. A user can specify the boarders of the area as a two-dimensional polygon. For example, we can use a rectangle to specify the area of interest, where the borders of the area can be defined by the longitude and latitude of the four corner points of the rectangle. The road segments within the specified area are then extracted. Each road segment is represented as polyline associated with road attributes including segment name, segment length, longitude and latitude of the nodes in the segment and etc. To estimate the transition cost for route inference, a set of features are evaluated based on the extracted road attributes. The features can include information related to a segment itself such as length, historical traffic flow, speed limit and etc. They can also include the features regarding the relationship between different segments such as turning angle between two consecutive road segments. It should be noted that we do not limit the choice of features in our framework and researchers can define and choose their own features depending on the availability of different road attributes and target applications. In our current work, we use a set of features including the road length and turning angle to initialize the transition cost in the offline preprocessing stage.

Transition cost initialization: In this module, the transition cost is initialized to the transition points in the extracted road network. The transition point is defined as the intersection that connects one incoming road segment

with at least two or more outgoing road segments. The transition cost from the incoming road segment to one of outgoing road segment s_i is calculated based on the input as the set of features being used: $c^{s_i} = \phi(f_1^{s_i}, f_2^{s_i}, .., f_n^{s_i})$ where ϕ denotes a function that returns the cost value based on the given feature values. A smaller transition cost value indicates the road segment s_i is more likely to be included in the inferred travelling route. To ensure the generality of our framework, we do not limit the form of the function ϕ. It can be weighted summation of the input features or non-linear combination of them. We adopt the transition cost assignment method in [40], which is a weighted summation of segment length feature and turning angle feature. The calculated transition costs for every transition point are then saved in our database organized with an R-tree indexing scheme. During the online inference stage, these initial transition costs are retrieved and dynamically updated based on additional information obtained in real time, such as distance to GPS points.

5.2. Procedure of incremental online inference

Incremental online inference is the key part of our framework. It adopts an opportunistic approach which incrementally infers the route with the extension of the existing inferred partial route whenever a new GPS point becomes available. In addition, a rollback mechanism is used to correct the already inferred route when some abnormal situations are detected through the examination of a set of rollback conditions. The general procedure of our incremental online inference with rollback (INC-RB) is descibed in Algorithm 1.

As shown in Algorithm 1, INC-RB receives the inputs of a new GPS point g_i, the inferred partial route L_{i-1} up to the previous GPS point g_{i-1} and a rollback depth that controls the level of rollback. The algorithm returns the inferred partial route L_i up to the GPS point g_i as the output. To achieve this, the candidate segments of g_i are firstly generated (line 1) based on the emission probability using gaussian kernel as:

$$p(s_i^f|g_i) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{dist(s_i^f, g_i)^2}{2\sigma^2}} \quad (1)$$

Algorithm 1: INC-RB

Input: next GPS point g_i, inferred partial route L_{i-1}, rollback depth dep;

Output: Updated inferred partial route L_i;

1 Obtain candidate segments S_i of g_i;
2 Determine the cost values of the transition points in the road network;
3 **if** *i=1* **then**
4 Obtain candidate segments of g_0;
5 Infer all the candidate local routes between candidate segments of g_0 and those of g_1, then sort and save them to the ranking table RT_1;
6 l_1^* = the route with the highest rank in RT_1;
7 $L_i \leftarrow l_1^*$;
8 **else**
9 Infer all the candidate local routes between the last segment in L_{i-1} and all candidate segments of g_i, then sort and save them to the ranking table RT_i;
10 l_i^* = the route with the highest rank in RT_i;
11 $L_i \leftarrow L_{i-1} \cup l_i^*$;
12 **if** *rollback conditions are met* **then**
13 $L_i \leftarrow$ *Rollback* (L_i, *dep*, $< RT_{i-dep}, ..., RT_{i-1} >$);

where σ is the estimated standard deviation of GPS measurement noise, $dist(s_i^f, g_i)$ indicates the minimum great circle (MGC) distance [4] between the candidate road segment s_i^f and the GPS point g_i.

The cost values of the transition points in the extracted area of road network are then determined (line 2) for local routes search. The determination of cost value considers both initial transition costs set in the offline preprocessing stage as well as the real-time cost that relates to the distances of road segments to the new GPS point g_i. Specifically, we use the same way to calculate the cost values as described in [40].

For the local route search, we consider two cases. The first case is that g_i is the second GPS point (line 3) in the whole GPS trajectory (i.e., no previous inferred partial route yet), the local route search is thus needed to be performed between each pair of candidate segments $\{s_0^f, s_1^t\}$ of the first GPS point (i.e., g_0) and the second GPS point (i.e., g_1) respectively. The cost of a candidate

local route $l_1^{(f,t)}$ is denoted as $c(l_1^{(f,t)})$, which is calculated as the summation of the transition costs of the transition points in the local route $l_1^{(f,t)}$. The transition probability that the actual route from g_0 to g_1 follows the candidate local route $l_1^{(f,t)}$ is then derived. Specifically, we consider both the cost of candidate local route and the MGC distance between g_0 and g_1 to derive the transition probability $p(l_1^{(f,t)})$ as follows:

$$p(l_1^{(f,t)}) = exp(-\alpha \times \frac{c(l_1^{(f,t)})}{dist(g_0, g_1)}) \tag{2}$$

where $dist(g_0, g_1)$ is the MGC distance between g_0 and g_1, α is the scaling factor.

The selection score for the candidate local route $l_1^{(f,t)}$ is thus calculated based on both the emission and transition probabilities as follows:

$$sc(l_1^{(f,t)}) = p(s_0^f|g_0) \times p(l_1^{(f,t)}) \times p(s_1^t|g_1) \tag{3}$$

Based on the calculated selection scores, the candidate local routes between g_0 and g_1 are next sorted are stored in the ranking table RT_1. The highest-score route in the ranking table is selected as the inferred local route l_1^* and added to L_1 (line 6 and 7).

The second case is that g_i is the third or subsequent GPS point (line 8). That is, there has already been an inferred partial route (i.e., $i > 1$). In such case, L_i will be generated by extending from the last road segment s_{i-1}^{lst} in L_{i-1}. Here, unlike the HMM-based conservative approach which requires pair-wise local route search, our local route search only needs to be performed between s_{i-1}^{lst} and the candidate segments of g_i (i.e., 1 to n instead of n to n search).

Unlike in Eq. 2, the transition probability in this case is derived with the consideration of the previous decisions (i.e., the partially inferred route). Thus, the transition probability for $l_i^{(lst,t)}$ is calculated as follows:

$$p(l_i^{(\text{lst},t)}) = exp(-\alpha \times \frac{c(l_i^{(\text{lst},t)}) + c(L_{i-1})}{\sum_{j=1}^{i} dist(g_{j-1}, g_j)}) \tag{4}$$

where $c(l_i^{(\text{lst},t)})$ denotes the cost of the candidate local route $l_i^{(\text{lst},t)}$ connecting s_{i-1}^{lst} and s_i^t, $c(L_{i-1})$ is the cost of L_{i-1} which is determined as $c(L_{i-1}) = \sum_{j=1}^{i-1} c(l_j^*)$

(l_j^* represents the inferred local route between g_{j-1} and g_j) and $dist(g_{j-1}, g_j)$ is the MGC distance between g_{i-1} and g_i.

The selection score of $l_i^{(\mathrm{lst},t)}$ is thus calculated through combining Eq. 1 and Eq. 4 as follows:

$$sc(l_i^{(\mathrm{lst},t)}) = p(l_i^{(\mathrm{lst},t)}) \times p(s_i^t|g_i) \tag{5}$$

Based on the calculated selection scores, the candidate local routes are sorted and stored in the ranking table RT_i (line 9). The highest-score route in the ranking table is then chosen as the inferred local route l_i^* to extend L_{i-1} (line 11). The route inference for the GPS point g_i is then ended if there is no rollback condition detected. Otherwise, a rollback correction mechanism will be triggered (line 12 to line 13). The detailed descriptions of rollback mechanism will be presented in the next sub-section.

5.3. Rollback mechanism

To deal with abnormal situations during our incremental route inference process, our rollback mechanism is designed with two parts. The first part is to examine a set of rollback conditions for detecting the abnormal situations so as to trigger the rollback operation. The second part is a provision of rollback correction operation that corrects the already inferred partial route. The general principle of rollback operation is to rollback to the previous inference states and re-doing the inference.

Rollback condition detection: The first task is to determine when it is necessary to trigger rollback. To this end, we design a set of rollback conditions and if any of rollback conditions is met, the rollback correction is then performed. Specifically, we consider three types of rollback conditions: *no route* condition, *high cost* condition and *abnormal driving* condition. The *no route* condition occurs when there is no reachable local route between the previous and current GPS points. The *high cost* condition is triggered when all the candidate local routes between the previous and current GPS points are of high cost values. The *abnormal driving* condition defines the situation where the inferred local route exhibits some abnormal driving patterns of a vehicle, such as unnecessary U-turn

or detour. The detailed definitions of these rollback conditions are described below. Note that it is possible to include more rollback conditions as necessary to enhance the inference accuracy of our INC-RB in future.

Rollback condition 1 (No route) : No reachable candidate local route is found between g_{i-1} and g_i. This implies that the previous inference has led to a dead end situation where the previous inferred partial route must be altered.

Rollback condition 2 (High cost) : The MGC distances from the line connecting g_{i-1} and g_i to all the candidate local routes between g_{i-1} and g_i exceed the distance threshold η, and at the same time, the costs of all these local routes exceed certain cost threshold ξ (Note that for each candidate local route we measure the maximum MGC distance between the route and the line of the consecutive corresponding GPS points). This condition implies that the local routes have deviated from the line connecting g_{i-1} and g_i which provides a rough indication of actual route direction. Meanwhile, all these local routes have very low transition probabilities (i.e., corresponding to high costs) to match the actual route. Note that the values of threshold η and ξ are dynamically determined during the online inference process (see section 6.2 for details).

Rollback condition 3 (Abnormal driving) : The newly inferred local route between g_{i-1} and g_i contains too many large angle turns which can be considered to be abnormal from driving behavior's perspective. This condition is evaluated based on the calculation of turning rate as follows:

$$tr_i = \frac{\sum \chi(\theta_i^k/(0.5\pi))}{snum(l_i^*) - 1} \tag{6}$$

where θ_i^k is the angle of the turn k on the inferred local route l_i^*, $\chi(x)$ is a defined step function which returns 1 if $x \geq 1$ and 0 otherwise, the function $snum(l_i^*)$ returns the number of segments in the inferred local route l_i^*. In the case that $snum(l_i^*) = 1$, tr_i is set to 0. The rollback condition 3 is triggered when tr_i is greater than a threshold τ which is dynamically determined during the online inference process (see section 6.2 for details).

Rollback correction: Once one of rollback conditions is met, the second

task is to correct the inferred partial route L_i by rolling back to the matching results of the previous GPS points and search for the alternative feasible routes. We first rollback to the previous level (i.e., $i-1$). If a new feasible route extending to g_i can be found, we replace the old route and stop rolling back further. Otherwise, we iteratively rollback to further levels up to the rollback depth *dep*. The process of rollback correction is shown in Algorithm 2.

Algorithm 2: Rollback

Input: Inferred partial route L_i, Rollback Depth *dep*, Ranking tables $< RT_{i-dep}, ..., RT_{i-1} >$

Output: Updated inferred partial route L_i;

```
for k = 1 to dep do
    L'_{i-k} ← < l*_1, ..., l*_{i-k} >;
    for each alternative local route l'_{i-k} in RT_{i-k} do
        Replace l*_{i-k} in L'_{i-k} with the alternative local route l'_{i-k};
        Re-generate a new L'_i that starts from l'_{i-k};
        Re-generate new ranking tables < RT'_{i-k+1}, ..., RT'_i >;
        if sc(L'_i) > sc(L_i) according to Eq. 7 then
            L_i ← L'_i;    // L_i is updated by L'_i
            < RT_{i-k+1}, ..., RT_i > ← < RT'_{i-k+1}, ..., RT'_i >;
    if L_i has been updated then
        return L_i;
return L_i;
```

In Algorithm 2, the rollback level k ($k \in [1, dep]$) indicates the current level where rollback correction is performed. At each rollback level, we first initialize L'_{i-k} with L_{i-k} (that is, $< l^*_1, ..., l^*_{i-k} >$, a subset of the existing inferred local route in L_i). Then, we retrieve the alternative local routes stored in the ranking table RT_{i-k} which were previously generated as shown in Algorithm 1. We replace the old l^*_{i-k} with each alternative local route l'_{i-k} and re-generate L'_i according to the same procedure as described in Algorithm 1.

To compare the original solution L_i and the new solution L'_i, the joint score

of the inferred local routes is calculated as follows:

$$sc(L_i) = \prod_{j=i-k}^{j=i} sc(l_j^*) \tag{7}$$

where $sc(l_j^*)$ is the selection score of the inferred local route l_j^* ($j \in [i-k, i]$). If $sc(L_i^{'})$ is higher than $sc(L_i)$, the inferred partial route L_i will be corrected by $L_i^{'}$. The original ranking tables $< RT_{i-k+1}, ..., RT_i >$ are also updated accordingly. After examining all the alternative local routes, the rollback process will be terminated if L_i has been corrected. Otherwise, the rollback process will proceed to the next level. If L_i is not updated through the rollback correction process, the algorithm will return the existing L_i.

6. Experiments and results

This section shows the evaluation of the proposed INC-RB in terms of both the inference accuracy and output latency. We first describe the real world dataset for evaluation and other online map matching algorithms for comparison. We then present the parameter settings used in our algorithm as well as the compared algorithms. Next, we describe the evaluation metrics being used. Lastly, we show the experiment results, the detailed analysis of our rollback mechanism and a discussion of the limitations of our INC-RB.

6.1. Dataset and compared algorithms

A large-scale real dataset consisting of 100 GPS trajectories over 100 cities (one GPS trajectory per city) around the world [21] is used for evaluation. The dataset contains up to 247,251 GPS points with a sampling rate of 1Hz and the length of the trajectories varies from 5km to 100 km. In addition, the trajectories are labeled with a set of different features that may post difficulties for map matching. These features are as follows and the number of trajectories with different features are shown in Table 1.

- u-turns: the vehicle turned 180 degree and reversed the direction of travel

- hives: many GPS points are packed in a small area
- loops: the trajectories of the vehicle are in circles
- gaps: there exist temporal gaps in the GPS trajectory
- severe congruence issues: situations where the map and the GPS trajectory are incongruent or dissimilar

Table 1: Number of trajectories with different features

loops	gaps	u-turns	hives	congruence-issues	no-tags
24	73	25	3	20	19

The GPS trajectories under different sampling rates are generated by sub-sampling from the original dataset. Specifically, we use the sampling rates of 60s, 120s and 180s for our evaluation. We evaluate and compare the proposed INC-RB with three state-of-the-art online map matching algorithms as described below. The selection of compared algorithms is based on two considerations. Firstly, the algorithms show the recent advancements that improve from the classical HMM-based model [8] in terms of inference accuracy and latency. Secondly, similar to our INC-RB, the algorithms leverage only a minimum set of information (i.e., GPS trajectories and road network topology data) for route inference. Therefore, the algorithms (e.g., [18, 35]) that require additional information such as vehicle's velocity, free-flow travel time, road class, etc., are not selected for comparison.

- OLMM [23]: The algorithm is a HMM-based online map matching algorithm that uses an online learning method to dynamically tune the key parameter in map matching model. The online matching method utilizes a small piece of trajectory data and their matching result to tune the parameter value in the subsequent inference. The matching process in OLMM adopts the online Viterbi algorithm proposed in [8] to compute the partial inferred route based on the detection of convergence point.

- Heuristic Optimization (HEU) [17]: The algorithm is a HMM-based online map matching algorithm that proposes several heuristic optimization for pruning low probability states, reducing the shortest path search space and detection of convergence conditions respectively. By adopting these optimization techniques, both computation speed and output latency are improved from the original HMM-based model [8]. In HEU, the partial inferred route can be generated before the convergence point is found.

- Feature-based method (FEA) [39]: In this algorithm, a set of features that considers both GPS observations and human factors is proposed for estimating the cost of candidate routes during the matching process. The algorithm offers both offline and online versions.

6.2. Parameter settings

To have a fair comparison, the standard deviation of GPS measurements σ is initially set to 10 and the maximum number of candidate segments for each GPS point is set to 10 for all the compared algorithms and our INC-RB. For OLMM, the parameter σ is dynamically adjusted from its initial value during the inference according to its online learning method [23]. For HEU, we use the same settings as described in [17]. The parameters for pruning unlikely candidate segments and candidate local routes are set to 100 and 1.1 respectively. The threshold value, which is used for the early output of the partially inferred route to decrease the decoding delay is set to 0.9. For FEA algorithm, the balance factor ω which controls the transition angle is set to 100 the same as in [39].

For our INC-RB, the scaling factor α, which is used to determine the transition score is set to 0.1 according to the analysis of the ground truth data. The rollback depth *dep* is set 2 according to our analysis that will be described in section 6.4.4. For the parameters η, ξ and τ in our rollback conditions, the values of these parameters are dynamically determined during the online inference process. Specifically, given the partial inferred route $L_{i-1} :< l_1^*, l_2^*, ..., l_{i-1}^* >$, the values of η, ξ and τ are dynamically updated based on the average of the corresponding

values of these inferred local routes (i.e., partially inferred route from l_1^* to l_{i-1}^*). For example, to determine η, as maximum MGC distance of each inferred local routes (l_1^* to l_{i-1}^*) to their corresponding lines connecting GPS points have been obtained, then, η is determined as the average value of these distance values. Similarly, ξ is determined as the average value of the costs of these inferred local routes from l_1^* to l_{i-1}^*. For τ, it is also determined as the average value of the turning rates from l_1^* to l_{i-1}^*.

6.3. Evaluation metrics

In our experiments, the proposed INC-RB and the above algorithms are evaluated and compared from the three aspects: inference accuracy, output latency and execution efficiency. In addition, the effectiveness of rollback mechanism in INC-RB is also evaluated. The evaluation metrics used in these experiments are described as follows.

Inference accuracy: To evaluate how the inferred route matches with the real route, we adopt four evaluation metrics similar to the ones in [39]. These include route mismatch ratio (RMF), map matching precision, recall and f-score. The route mismatch fraction (RMF) was initially proposed by Newson and Krumm [27] and has been used in the accuracy evaluation for both offline (e.g., [27]) and online map matching (e.g., [39]). RMF computes the ratio of the total mismatched distance over the total length of the real path. Specifically, RMF is measured as follows:

$$RMF = \frac{L_+ + L_-}{L_{\text{real}}} \tag{8}$$

where L_+ is the total length of the mismatched road segments in the inferred route, L_- is the total length of the mismatched road segments in the real route and L_{real} is the total length of the real route. A smaller RMF value indicates a higher inference accuracy.

The map matching precision is measured as the ratio between the total length of the matched road segments and the total length of the inferred route. The map matching recall is measured as the ratio between the total length of the

matched road segments and the total length of the real route. These two metrics are therefore formulated as follows:

$$precision = \frac{L_{\text{matched}}}{L_{\text{inferred}}} \tag{9}$$

$$recall = \frac{L_{\text{matched}}}{L_{\text{real}}} \tag{10}$$

where L_{matched} represents the total length of the matched road segments and L_{inferred} represents total length of the inferred route.

The F-score, which is a harmonic mean of precision and recall, can then be defined as:

$$F\text{-}score = 2 \times \frac{precision \times recall}{precision + recall} \tag{11}$$

Output latency: Due to the real-time requirement in many GPS-enabled applications, output latency is another important aspect for assessing the performance of online map matching algorithm. In general, the output latency measures the time lag between the time a GPS observation is received and the time when the inferred route up to that GPS point is produced by an online map matching algorithm. Such output latency can be further decomposed into *client-server transmission time*, *algorithm execution time* and *decoding delay*. The client-server transmission time in a networked environment is the time required to receive the GPS observation data from a remote client and return the matching results back to the client. The algorithm execution time is the time needed for executing a given map matching algorithm and it often depends on the complexity of the designed algorithm and the processing power of running machine. The decoding delay is the inherent delay that is due to the localizing strategy [8] used by a given online map matching algorithm. In this work, all the GPS data and the tested algorithms are handled in a local machine. Therefore, the output latency T_{lat} in our experiment does not include client-server transmission time (i.e., T_{lat} = *algorithm execution time*+*decoding delay*).

The decoding delay is the most influential component in the overall output latency and it varies based on different localizing strategies for online decoding as follows.

- Fixed sliding window (FSW): for algorithms adopting FSW strategy (e.g., FEA), the partial inferred route can be always obtained through moving the window forward one step. Therefore, the decoding delay is expected to be constant for all the GPS observations under a given window size, but it increases with the increase of window size.

- Variable sliding window (VSW): for algorithms adopting VSW strategy (e.g., HEU), the sliding window can shrink and expand depending on the detection of convergence point. Therefore, the decoding delay can vary for different GPS observations, but the overall decoding delay is not affected by the change of window size.

- Bounded variable sliding window (BVSW): for algorithms adopting BVSW strategy (e.g., OLMM), the decoding latency is the same as the one in VSW when the window size does not exceed the defined bound and as the one in FSW when the window size exceeds the defined bound.

For our INC-RB algorithm, we do not adopt any of the above localizing strategies and we use an opportunistic approach to infer the route up to the current GPS point. Therefore, we do not have decoding delay in the most cases except the time when rollback mechanism is triggered. Therefore, we consider the algorithm execution time as well as the time required to perform rollback mechanism to measure the output latency for our algorithm.

Execution efficiency: To evaluate the execution efficiency, we measure the algorithm execution time of a given online matching algorithm at each GPS point and the average value over all the GPS points is recoded as the average execution time. The average execution time of INC-RB, OLMM, HEU and FEA are measured with Dell Precision Tower 5810 with Intel(R) Xeon(R) CPU E5-1650 V4@3.60GHz and 32 GB RAM operated with a 64-bit windows 10 system.

Performance of rollback mechanism: The design of rollback mechanism plays a key role in our algorithm to ensure the accuracy of the route inference

while achieving minimal output latency. Therefore, it is important to understand to what extent such rollback mechanism is triggered during the entire inference process and whether the rollback correction can work once it is triggered. To this end, we propose two evaluation metrics, rollback frequency and rollback improvement rate. The rollback frequency (RF) counts the number of times that rollback correction is triggered during the inference process of an entire GPS trajectory where GPS points are incrementally added (i.e., in an online manner). Thus, the rollback frequency can be calculated as follows:

$$RF = \frac{1}{N^{\mathbb{T}}} \sum_{i=1}^{N^{\mathbb{T}}} \frac{N_i^{\mathrm{rb}}}{n_i - 1} \tag{12}$$

where $N^{\mathbb{T}}$ denotes the number of all trajectories being tested, N_i^{rb} denotes the total number of rollback correction being triggered for a given trajectory i, n_i is the number of the GPS points in the trajectory i. A small RF indicates a small percentage of rollback corrections are triggered, which means less overhead and faster inference.

The rollback improvement rate (RI) measures how the accuracy of inference results can be improved by the rollback mechanisms when comparing with the alpha version of our INC-RB with rollback mechanisms being disabled. Here, we use RMF (i.e., smaller is better) to measure the accuracy and the rollback improvement rate RB can then be calculated as follows:

$$RI = \frac{1}{N^{\mathbb{RT}}} \sum_{j=1}^{N^{\mathbb{RT}}} \frac{RMF_j^- - RMF_j^+}{RMF_j^-} \tag{13}$$

where $N^{\mathbb{RT}}$ denotes the number of trajectories among all trajectories being tested where the rollback operation(s) are triggered, RMF_j^+ is the RMF value of a given trajectory j with rollback mechanism is enabled and RMF_j^- is the RMF value of a given trajectory j with rollback mechanism is disabled. A positive RI indicates the rollback operation can improve the overall accuracy.

6.4. Experimental results and analysis

6.4.1. Inference accuracy results

in Table 2, we show the average inference accuracy in terms of RMF, precision, recall and F-score over the total 100 GPS trajectories (i.e., one GPS trajectory per city) under different sampling rate. It can be observed that our INC-RB outperforms the compared algorithms in the most cases especially under the low-sampling rates. In terms of RMF, INC-RB achieves up to 18.5% and 32.6% compared with OLMM, 15.8 % and 24.8% compared with HEU and 14.4% and 13.3% improvement compared with FEA under the sampling rates of 120s and 180s. INC-RB can also obtain improved precision, recall and f-score under all the three sampling rates except for the case of precision under 120s sampling rate.

Table 2: Statistics on inference accuracy under different sampling rates (sr)

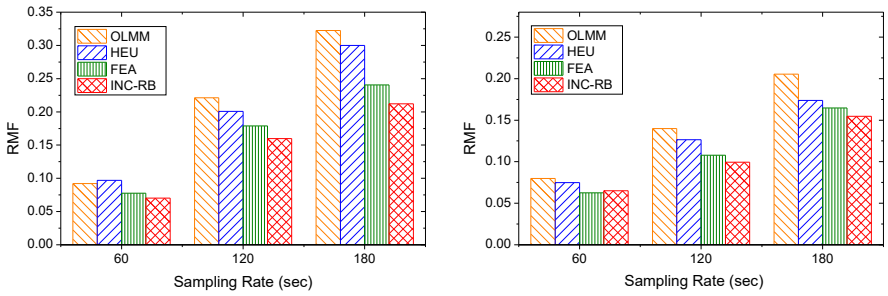
Metrics \ **Algorithms**		OLMM	HEU	FEA	INC-RB
RMF	(sr = 60s)	0.079	0.066	0.065	**0.064**
	(sr = 120s)	0.124	0.120	0.118	**0.101**
	(sr = 180s)	0.184	0.165	0.143	**0.124**
Precision	(sr = 60s)	0.946	0.968	0.970	**0.973**
	(sr = 120s)	0.937	0.953	**0.964**	0.957
	(sr = 180s)	0.907	0.913	0.910	**0.943**
Recall	(sr = 60s)	0.926	0.918	0.920	**0.934**
	(sr = 120s)	0.889	0.921	0.918	**0.938**
	(sr = 180s)	0.862	0.887	0.916	**0.917**
F-score	(sr = 60s)	0.932	0.941	0.953	**0.956**
	(sr = 120s)	0.924	0.932	0.935	**0.946**
	(sr = 180s)	0.892	0.909	0.914	**0.922**

Among the compared algorithms, FEA can achieves the best inference accuracy. This is mainly because that unlike OLMM and HEU which take shortest path as the candidate local route between two candidate segments, FEA considers several human behavior features to determine the path that can reflect real driver's route choices. Therefore, it can yield the more natural route when determining the local routes. Our INC-RB also adopts the similar features to derive the transition probability of candidate local routes. However, our INC-RB introduces the rollback mechanism to further ensure the accuracy of inference.

Next, we investigate the inference accuracy of INC-RB for trajectories under different features. The average RMF of the trajectories belonging to each feature are shown in Fig. 4. It can be seen that our INC-RB can also obtain good performance for the trajectories under different features, even for the challenging features such as u-turns and gaps. For the trajectories with hive which indicates a large volume of GPS points are packed in a small area and the trajectories with congruence issues (CI) which indicates the map and the GPS trajectories are dissimilar, these trajectories may potentially contain many noises. It can been observed that our INC-RB achieves a much better RMF value compared to the other three algorithms for the trajectories with hive and CI. The trajectories with no-tags are those with the high-quality GPS data. Therefore, all the algorithms can achieve relatively small RMF values for these trajectories.

6.4.2. Output latency results

Figure 5 shows the output latency with the increase of window size of localizing strategies under different sampling rates. It can be seen that the output delay of FEA generally increases proportionally with window size because FSW is used as the localizing strategy. By using BVSW, the average output latency of OLMM remains unchanged once the window size reaches 6 for the sampling rates of 60s and 120s and 4 for the sampling rate of 180s. This means BVSW used in OLMM has no significant advantages at the window size of 4 and above. As HEU adopts localizing strategy that functions similar to VSW, the output latency remains constant under different window size. HEU specifically



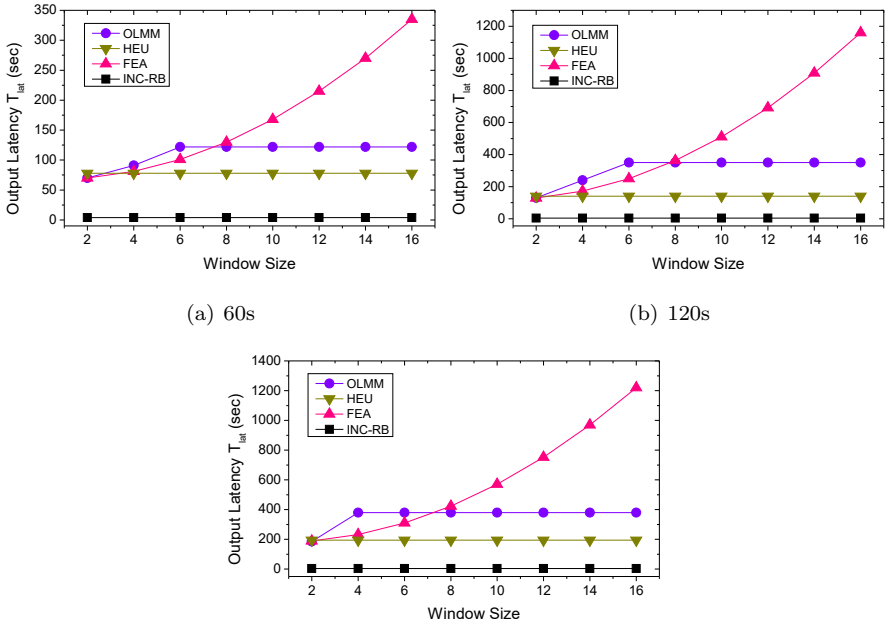
(a) Loops (b) Gaps

(c) U-turns (d) Hive

(e) CI (f) No-tags

Figure 4: RMF comparison with different trajectory features

introduces the heuristic optimization to reduce the decoding delay. Therefore, the latency keeps lower compared with OLMM and FEA. Our proposed INC-RB achieves the best output latency and it is also not affected by the window size. This is because we use the opportunistic approach in which the delay only occurs when the rollback mechanism is triggered.

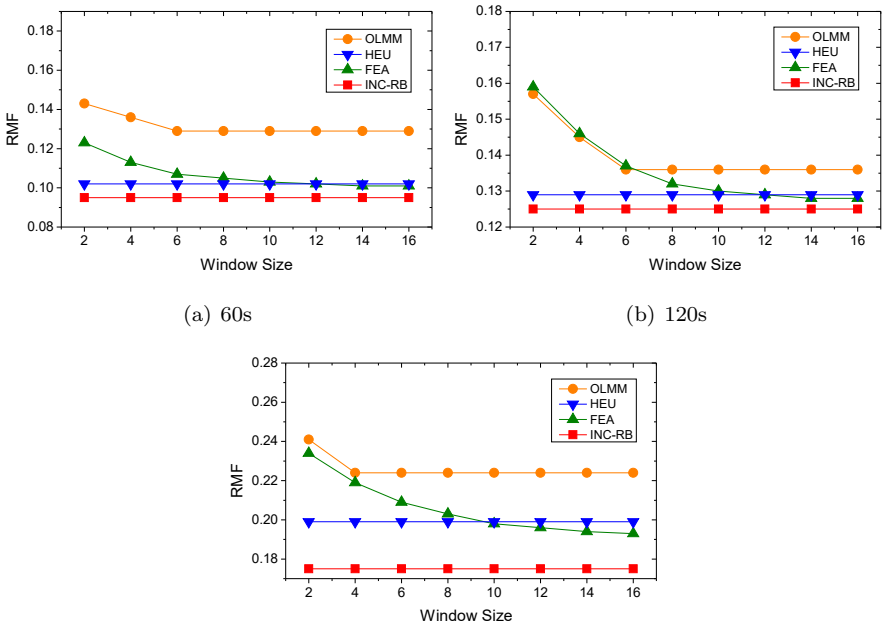


(a) 60s (b) 120s

(c) 180s

Figure 5: Average output latency with the increase of window size under different sampling rates

In conjunction with Figure 5, Figure 6 shows the corresponding average RMF of each algorithm with the increase of window size. For OLMM and FEA, smaller RMF can be obtained when the window size increases from 2 to 6. When the window size is above 6, RMF value of OLMM remains unchanged which lies in the reason that the inferred partial route has been updated before the window bound is reached. For FEA, RMF value deceases slowly when the window size is larger than 8. For HEU and our INC-RB, the RMF value is not affected by the window size. Regardless of window size, our INC-RB can achieve



(a) 60s (b) 120s

(c) 180s

Figure 6: Average RMF with the increase of window size under different sampling rates

the lowest RMF value for all the sampling rates. Compared to HEU which has the shortest output latency among the compared algorithms, our INC-RB reduces the output latency by 92.67%, 96.12% and 97.51% for the sampling rates of 60s, 120s, and 180s respectively.

6.4.3. Execution time results

The average algorithm execution time is measured for evaluating the execution efficiency of each testing algorithm. Specifically, we select all 3 GPS trajectories of hive feature and randomly select 5 GPS trajectories of each of other features. The execution time results over the 28 GPS trajectories are shown in Table 3. We also conducted the independent two sample t-test which shows that the average execution of INC-RB is significantly shorter than other three algorithms.

Among the other three algorithms, the average execution time of HEU is the shortest because the algorithms is optimized to reduce the total number of

Table 3: Average algorithm execution time (in seconds) over 28 cities

Sampling rate (sr)	60s	120s	180s
OLMM	12.06 ^{T}	12.54^T	13.15^T
HEU	8.46^T	9.26^T	9.74^T
FEA	11.75^T	12.12^T	12.66^T
INC-RB	**5.19**	**5.43**	**5.62**

T: significantly better (t-test)

route search by pruning unlikely candidate segments. Compared with HEU, the proposed INC-RB still can achieve 38.7%, 41.3% and 42.2% improvements in terms of the average execution time under the three sampling rates respectively. The performances of OLMM and FEA in terms of average execution time are similar as the size of search space for determining the local routes are similar in these two algorithms. Our INC-RB outperforms the other compared algorithms mainly because that its opportunistic inference strategy requires a smaller search space and is capable of producing a fast inference up to the current GPS point.

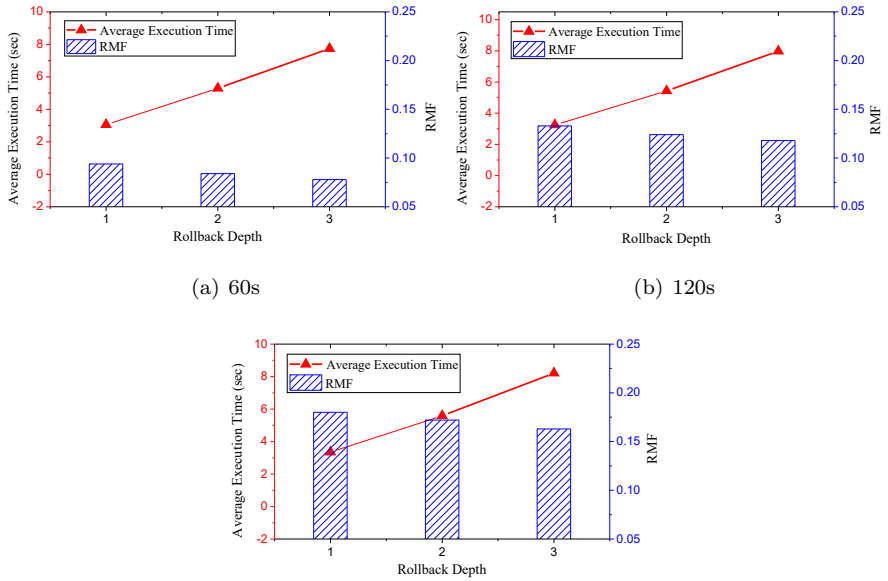
It should be also noted that the execution time of our INC-RB reported here also improves from our previous work [13]. This is mainly because that we have optimized the threshold values, such as η and ξ, in the rollback conditions so as to reduce the number of times that rollback needs to be triggered.

6.4.4. Analysis of rollback mechanism

In this section, we first present the analysis on how the setting of rollback depth *dep* parameter affects the inference accuracy and execution time. Then, we present the evaluations based on two metrics: rollback frequency (RF) and rollback improvement rate (RI) to examine the performance of rollback mechanism.

In our INC-RB, the parameter of rollback depth *dep* affects the maximum

number of rollback correction operations that can be executed. Therefore, a larger value of *dep* generally implies a higher computational cost, even though it may help to ensure the success of the rollback correction. To analyze how how *dep* affects the algorithm performance, we evaluate the RMF inference accuracy as well as the average execution time under the settings of different *dep* values. Fig 7 shows the corresponding results with *dep*'s value being set from 1 to 3.



(a) 60s

(b) 120s

(c) 180s

Figure 7: Impact of rollback depth *dep* under different sampling rates

As shown in Figure 7, the average execution time increases almost linearly with the increase of *dep* value. For the sampling rate of 60s, there is a slight improvement on the inference accuracy (i.e., the decrease of RMF value) with the increase of *dep* value. For the sampling rate of 120s and 180s, the inference accuracy decreases when $dep = 3$. This may be because that the distance between GPS points increases as the sampling interval is larger. In such situation, the rollback operation requires an evaluation of more alternative routes, which may bring additional noises. Therefore, we find that the setting of $dep = 2$ can achieve a good balance between the inference accuracy and efficiency for our

current dataset.

Table 4: Rollback frequency (RF) and improvement rate (RI) under different sampling rates (sr)

Metrics \ Features		Loops	Gaps	U-turn	Hive	CI	No-tags
RF	(sr = 60s)	6.2%	6.0%	8.2%	5.0%	11.0%	5.3%
	(sr = 120s)	8.4%	9.6%	10.3%	6.8%	13.2%	6.6%
	(sr = 180s)	8.5%	10.6%	12.1%	9.4%	12.7%	7.5%
RI	(sr = 60s)	8.5%	9.5%	5.1%	13.3%	11.3%	7.2%
	(sr = 120s)	11.7%	8.2%	8.2%	10.2%	10.4%	5.9%
	(sr = 180s)	12.5%	17.9%	7.6%	18.5%	7.2%	6.4%

The average rollback frequency (RF) and rollback improvement rate (RI) of the trajectories being tested with different features under different sampling rates are shown in Table 4. It can be seen that RF is kept in a low level (less than 15 %) under all the sampling rates, which means the opportunistic inference is successful in the most of times and rollback corrections are only invoked occasionally. It can be also observed that RF generally increases from sampling rate of 60s to 180s. This is as expected because the low-sampling rate generally introduces more uncertainties and it is thus more likely to have some false inferences that need to be corrected. RI measures how the inference accuracy in terms of RMF can be improved by the rollback mechanisms compared with rollback mechanism being disabled. It can be seen that RI values generally increase when the sampling rate becomes low (e.g., 180s). This means the rollback mechanism plays a more role when dealing with low-sampling data. RI values also vary for the trajectories of different features. For the trajectories with features of loops, gaps and hive that may contain more noisy data, RI values are larger compared with the ones for the trajectories with other features. This indicates the rollback mechanism is especially helpful to ensure the inference accuracy when noisy data are presented.

6.4.5. Discussions

Although our proposed algorithm can achieve a good balance between inference accuracy and output latency, there exist some limitations and the room for improvements. Firstly, given the current dataset being used, our approach assumes a minimum set of information (the information on GPS points and topological structure of road network) are available for route inference. This potentially limits the factors we can consider when deriving the transition cost for local route search. In fact, as suggested in [18], driver's behaviors models are proven to be useful for estimating the travelling route. However, an in-depth analysis of driver behaviors often relies on additional information such as vehicles speed and heading directions. In future, we will consider to conduct data collection experiments by ourselves so as to obtain additional information and develop more advanced inference model for local route search in our algorithm. Secondly, in our current rollback conditions, the abnormal driving condition only considers the abnormality exists in driver's turning behavior. If additional information such as driver's velocity becomes available, we can also extend this condition with the detection of more types of abnormal driving behaviors, such as overspeeding. This will be useful to further ensure the robustness of our opportunistic approach.

7. Conclusions

In this paper, we proposed an incremental route inference algorithm with rollback for online map matching. To minimize output latency, the proposed algorithm adopts an opportunistic approach that can infer the route up to the current GPS point. Furthermore, a rollback mechanism is proposed to correct the already inferred partial route when some abnormal situations happen. We evaluated our proposed INC-RB with a large-scale real-world dataset and compared it with three state-of-the-art online map matching algorithms OLMM, HEU and FEA. The results show that the performance of our algorithm is better or at least comparable to the compared algorithms in terms of inference

accuracy. More importantly, our algorithm has much shorter output latency, which is critical for some real-time applications, such as traffic sensing, navigation assistance and route recommendation.

Research on online map matching is still an ongoing research area where many challenges need to be tackled. There are several future directions. Firstly, the inference accuracy of our algorithm may be further enhanced by considering the driving preferences (e.g., distance-shortest vs. time-shortest) of different persons in our local route search process. Secondly, how to extend the proposed algorithm to pedestrian or multimodal map matching problem can be explored. Lastly, to fully utilize the results of map matching for useful applications (e.g., personalized route recommendation), it is also important to understand driver's intentions from travelled routes. Thus, the study on discovering the semantics of the matched routes (e.g., [36]) can be a further step of research.

Acknowledgements

This work is supported by National Natural Science Foundation of China (Grant No. 61872282), Natural Science Basic Research Plan in Shaanxi Province of China (Program No. 2019JM-031) and the Open Project Program of State Key Laboratory of Virtual Reality Technology and Systems, Beihang University (No. VRLAB2019C04).

References

[1] Aly, H., Youssef, M.. semMatch: Road semantics-based accurate map matching for challenging positioning data. In: Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems. 2015. p. 1–10.

[2] Atia, M.M., Hilal, A.R., Stellings, C., Hartwell, E., Toonstra, J., Miners, W.B., Basir, O.A.. A low-cost lane-determination system using GNSS/IMU fusion and HMM-based multistage map matching. IEEE Transactions on Intelligent Transportation Systems 2017;18(11):3027–3037.

[3] Bernstein, D., Kornhauser, A.. An introduction to map matching for personal navigation assistants 1998;.

[4] Boots, B., Okabe, A., Sugihara, K.. Spatial tessellations. Geographical information systems 1999;:503526.

[5] Brakatsoulas, S., Pfoser, D., Salas, R., Wenk, C.. On map-matching vehicle tracking data. In: Proceedings of the 31st International Conference on Very Large Data Bases. 2005. p. 853–864.

[6] Chen, C., Ding, Y., Xie, X., Zhang, S.. A three-stage online map-matching algorithm by fully using vehicle heading direction. Journal of Ambient Intelligence and Humanized Computing 2018;9(5):1623–1633.

[7] El Najjar, M.E., Bonnifait, P.. A road-matching method for precise vehicle localization using belief theory and kalman filtering. Autonomous Robots 2005;19(2):173–191.

[8] Goh, C.Y., Dauwels, J., Mitrovic, N., Asif, M.T., Oran, A., Jaillet, P.. Online map-matching based on hidden markov model for real-time traffic sensing applications. In: Preceedings of the 15th International IEEE Conference on Intelligent Transportation Systems. 2012. p. 776–781.

[9] Gong, Y.J., Chen, E., Zhang, X., Ni, L.M., Zhang, J.. Antmapper: An ant colony-based map matching approach for trajectory-based applications. IEEE Transactions on Intelligent Transportation Systems 2018;19(2):390–401.

[10] Greenfeld, J.S.. Matching GPS observations to locations on a digital map. In: 81th Annual Meeting of the Transportation Research Board. volume 1; 2002. p. 164–173.

[11] Hashemi, M., Karimi, H.A.. A critical review of real-time map-matching algorithms: Current issues and future directions. Computers, Environment and Urban Systems 2014;48:153–165.

[12] Hashemi, M., Karimi, H.A.. A weight-based map-matching algorithm for vehicle navigation in complex urban networks. Journal of Intelligent Transportation Systems 2016;20(6):573–590.

[13] Hou, X., Luo, L., Cai, W., Hanai, M.. Fast online map matching for recovering travelling routes from low-sampling GPS data. In: Proceedings of 2018 IEEE International Conference on Ubiquitous Intelligence and Computing. 2018. p. 917–924.

[14] Hsueh, Y.L., Chen, H.C.. Map matching for low-sampling-rate GPS trajectories by exploring real-time moving directions. Information Sciences 2018;433:55–69.

[15] Hu, G., Shao, J., Liu, F., Wang, Y., Shen, H.T.. IF-matching: Towards accurate map-matching with information fusion. IEEE Transactions on Knowledge and Data Engineering 2017;29(1):114–127.

[16] Hunter, T., Abbeel, P., Bayen, A.. The path inference filter: model-based low-latency map matching of probe vehicle data. IEEE Transactions on Intelligent Transportation Systems 2014;15(2):507–529.

[17] Jagadeesh, G.R., Srikanthan, T.. Heuristic optimizations for high-speed low-latency online map matching with probabilistic sequence models. In: 19th IEEE International Conference on Intelligent Transportation Systems. 2016. p. 2565–2570.

[18] Jagadeesh, G.R., Srikanthan, T.. Online map-matching of noisy and sparse location data with hidden markov and route choice models. Proceedings of the 18th International Conference on Intelligent Transportation Systems 2017;18(9):2423–2434.

[19] Kang, W., Li, S., Chen, W., Lei, K., Wang, T.. Online map-matching algorithm using object motion laws. In: Proceedings of 2017 IEEE 3rd International Conference on Big Data Security on Cloud. 2017. p. 249–254.

[20] Kim, J.. Node based map matching algorithm for car navigation system. In: 29th International Symposium on Automotive Technology & Automation. 1996. .

[21] Kubička, M., Cela, A., Moulin, P., Mounier, H., Niculescu, S.I.. Dataset for testing and training of map-matching algorithms. In: Intelligent Vehicles Symposium. 2015. p. 1088–1093.

[22] Kubicka, M., Cela, A., Mounier, H., Niculescu, S.I.. Comparative study and application-oriented classification of vehicular map-matching methods. IEEE Intelligent Transportation Systems Magazine 2018;10(2):150–166.

[23] Liang, B., Wang, T., Li, S., Chen, W., Li, H., Lei, K.. Online learning for accurate real-time map matching. In: Proceedings of the 20th Pacific-Asia Conference on Knowledge Discovery and Data Mining. 2016. p. 67–78.

[24] Liu, X., Liu, K., Li, M., Lu, F.. A ST-CRF map-matching method for low-frequency floating car data. IEEE Transactions on Intelligent Transportation Systems 2017;18(5):1241–1254.

[25] 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. 2009. p. 352–361.

[26] Nassreddine, G., Abdallah, F., Denoeux, T.. Map matching algorithm using interval analysis and dempster-shafer theory. In: 2009 IEEE Intelligent Vehicles Symposium. 2009. p. 494–499.

[27] Newson, P., Krumm, J.. Hidden markov map matching through noise and sparseness. In: Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. 2009. p. 336–343.

[28] Obradovic, D., Lenz, H., Schupfner, M.. Fusion of map and sensor data in a modern car navigation system. Journal of VLSI signal processing systems for signal, image and video technology 2006;45(1-2):111–122.

[29] Osogami, T., Raymond, R.. Map matching with inverse reinforcement learning. In: Proceedings of the 23rd International Joint Conference on Artificial Intelligence. 2013. p. 2547–2553.

[30] Phuyal, B.P.. Method and use of aggregated dead reckoning sensor and GPS data for map matching. In: 15th International Technical Meeting of the Satellite Division of The Institute of Navigation. 2002. .

[31] Quddus, M.A., Noland, R.B., Ochieng, W.Y.. A high accuracy fuzzy logic based map matching algorithm for road transport. Journal of Intelligent Transportation Systems 2006;10(3):103–115.

[32] Quddus, M.A., Ochieng, W.Y., Noland, R.B.. Current map-matching algorithms for transport applications: State-of-the art and future research directions. Transportation research part c: Emerging technologies 2007;15(5):312–328.

[33] Quddus, M.A., Ochieng, W.Y., Zhao, L., Noland, R.B.. A general map matching algorithm for transport telematics applications. GPS solutions 2003;7(3):157–167.

[34] Rappos, E., Robert, S., Cudré-Mauroux, P.. A force-directed approach for offline GPS trajectory map matching. In: Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. 2018. p. 319–328.

[35] Taguchi, S., Koide, S., Yoshimura, T.. Online map matching with route prediction. IEEE Transactions on Intelligent Transportation Systems 2019;20(1):338–347.

[36] Wan, C., Zhu, Y., Yu, J., Shen, Y.. Smopat: Mining semantic mobility patterns from trajectories of private vehicles. Information Sciences 2018;429:12–25.

[37] Wang, G., Zimmermann, R.. Eddy: an error-bounded delay-bounded real-time map matching algorithm using hmm and online viterbi decoder. In: Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. 2014. p. 33–42.

[38] White, C.E., Bernstein, D., Kornhauser, A.L.. Some map matching algorithms for personal navigation assistants. Transportation Research Part C: Emerging Technologies 2000;8(1):91–108.

[39] Yin, Y., Shah, R.R., Wang, G., Zimmermann, R.. Feature-based map matching for low-sampling-rate GPS trajectories. ACM Transactions on Spatial Algorithms and Systems 2018;4(2):4.

[40] Yin, Y., Shah, R.R., Zimmermann, R.. A general feature-based map matching framework with trajectory simplification. In: Proceedings of the 7th ACM SIGSPATIAL International Workshop on GeoStreaming. 2016. p. 1–10.

[41] Yuan, J., Zheng, Y., Zhang, C., Xie, X., Sun, G.Z.. An interactive-voting based map matching algorithm. In: Proceedings of the 11th International Conference on Mobile Data Management. 2010. p. 43–52.