On Efficient Map-matching According to Intersections You Pass By

Yaguang Li[1,3], Chengfei Liu[2], Kuien Liu[1], Jiajie Xu[1], Fengcheng He[1,3], and Zhiming Ding[1]

[1] Institute of Software, Chinese Academy of Sciences, Beijing 100190, China
{yaguang,kuien,jiajie,fengcheng,zhiming}@nfs.iscas.ac.cn
[2] FICT, Swinburne University of Technology, Australia
cliu@swin.edu.au
[3] University of Chinese Academy of Sciences, Beijing 100049, China

Abstract. Map-matching is a hot research topic as it is essential for Moving Object Database and Intelligent Transport Systems. However, existing map-matching techniques cannot satisfy the increasing requirement of applications with massive trajectory data, e.g., traffic flow analysis and route planning. To handle this problem, we propose an efficient map-matching algorithm called *Passby*. Instead of matching every single GPS point, we concentrate on those close to intersections and avoid the computation of map-matching on intermediate GPS points. Meanwhile, this efficient method also increases the uncertainty for determining the real route of the moving object due to less availability of trajectory information. To provide accurate matching results in ambiguous situations, e.g., road intersections and parallel paths, we further propose *Passby**. It is based on the multi-hypothesis technique and manages to maintain a small but complete set of possible solutions and eventually choose the one with the highest probability. The results of experiments performed on real datasets demonstrate that Passby* is efficient while maintaining the high accuracy.

Keywords: Efficient Map-matching, Multi-Hypothesis Technique

1 Introduction

The past few years have seen a dramatic increase of GPS-embedded and handheld navigation devices. These devices allow to record accurately the spatial displacement of moving objects. Given their low set-up cost, large volumes of data can be easily generated. In order to provide a range of Intelligent Transport Systems services, e.g., traffic flow analysis [1], route planning [2], hot route finder [3], we need to analyze large amounts of trajectory data. An essential pre-processing step that matches the trajectories to the road network is needed. This technique is commonly referred as map-matching.

However, existing map-matching techniques cannot fully satisfy the increasing requirement of applications with massive trajectory data. Some local/increment map-matching methods [4, 5] are fast but generate inconsistent matching results since they ignore the correlation between subsequent points and topological information of the road network. Meanwhile, more advanced map-matching methods, e.g., global approach [6, 7] and statistical approach [8, 9], manage to be quite accurate, but suffer from high computational complexity.

To handle this problem, we propose an efficient map-matching algorithm called Passby. Instead of matching every single GPS point, we concentrate on those close to intersections and avoid the computation of map-matching on intermediate GPS points. In the circumstances of constrained road network, if a moving object passes by the start vertex of edge e at t_i, and the end vertex of e at a reasonable time t_j, the object is then supposed to be on e at the time interval between t_i and t_j. Figure 1 illustrates an example of such case, i.e., we just have to match these thick blue points close to intersections instead of analyzing every single point. While cutting the cost of computing on intermediate GPS points

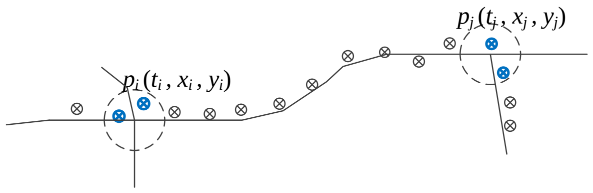


Fig. 1: An example of Passby.

makes map-matching more efficient, nevertheless it is not at no cost. The uncertainty for determining the real route of the moving object also arises due to less availability of trajectory information. The challenge of this work is to keep providing accurate matching results in ambiguous situations, e.g., road intersections and parallel paths. To handle this challenge, we propose another algorithm called Passby*, which is based on the multi-hypothesis technique, to maintain all the possible solutions in situations of ambiguity and eventually choose the one with the highest probability. Meanwhile, several efficient approaches for hypothesis management are developed to further improve the performance of Passby*.

To summarize, this paper makes the following contributions:

- We propose two efficient map-matching algorithms, Passby and Passby*. They avoid large parts of computation by only performing map-matching on GPS points close to intersections and inferring the remaining ones.
- We develop a set of novel approaches for hypothesis creation, evaluation and management which make the algorithm both efficient and accurate.
- We present a solution to edge simplification and edge measurement error. As Passby* mainly bases on positions of intersections and road connectivity, it is robust when geometric information of the edge is not available or not accurate enough.
- We conduct experimental study on real dataset and demonstrate the efficiency and accuracy of the proposed algorithms.

The rest of the paper is organized as follows. Section 2 discusses the related work. Section 3 gives the problem statement. The proposed algorithm is described in Section 4. The results of experiments and the analysis are presented in Section 5. Finally, conclusions and future work are summarized.

2 Related Work

In this section, we present a brief overview of previous studies on map-matching and the multi-hypothesis technique.

2.1 Map-matching Techniques

Map-matching is commonly referred as the process of aligning a sequence of observed positions with the road network. Approaches for map-matching algorithms can be categorized into three groups [6]: local/incremental method [4, 5, 10], global method [6, 7, 11], and statistical method [8, 9]. The local/incremental method tries to find local match of geometries, and performs matching based on the previous matching result of a point. The global method aims to find a global optimal match for the entire trajectory and a path in the road network. The algorithm in [11] is based on Frechét distance and its variant. [6] proposes an algorithm that takes temporal information into consideration and manages to get a high accuracy for low-sampling-rate GPS trajectories. Statistical method is also widely used. A method based on the Bayesian classifier is presented in [9]. [8] proposes an algorithm that takes advantage of the rich information extracted from the historical trajectories to infer the real routes.

Many advanced techniques are also integrated with map-matching [12]. [13] proposes a fuzzy logical based method that is able to take noisy, imprecise input and yield crisp output. [7] proposes a method based on the Hidden Markov Model which can deal with the inaccuracies of the location measurement systems.

Most existing map-matching algorithms have to perform map-matching on every single trajectory point which is not always necessary. Instead of matching every single GPS point, we concentrate on those close to intersections and infer the results for the remaining points to avoid extra computation.

2.2 Multi-Hypothesis Technique

The Multi-Hypothesis Technique(MHT) is a method to track multiple targets under the clutter environment using a likelihood function [14]. To realize a map matching method using the MHT, pseudo-measurements are generated utilizing adjacent roads of GPS position and the MHT is reformulated as a single target problem [15]. The main advantage of multi-hypothesis map-matching over a mono-hypothesis approach is that it maintains all the possible solutions in situations of ambiguity and eventually chooses the one with the highest probability. Two map-matching methods using the MHT are proposed in [15] and [16]. These previous works are more focused on the accuracy than the efficiency which makes them unsuitable for processing massive trajectory data. Moreover, these methods have to use extra information from the dead reckoning device, e.g., heading, acceleration and speed, which is not always available.

3 Problem Statement

We have a standard road network representation in which intersections are represented by vertices and roads are represented by directed edges and the geometric detail of roads are described by polylines.

Definition 1 (Road Network). *The road network G is defined as:*

$$G = (V, E) \tag{1}$$

where V is the set of vertices while E is the set of directed edges. A vertex of G is defined as: $v = (vid, lat, lng)$, where vid is the identifier of v while lat and lng denote the latitude and longitude of v. An edge of G is defined as: $e = (eid, geo, len, v_{from}, v_{to})$, where eid is the identifier of e, len is the length of e, v_{from} and v_{to} represent the start and the end vertices of e, geo is a polyline representing the geometry of e.

In the most simplified road network described in [17], geo is not available which will pose a great challenge to map-matching algorithms. For robustness evaluation, we will also conduct experiments on the most simplified road network.

Definition 2 (Path). *A path P is defined as a sequence of edges:*

$$P = (e_i)_{i=1}^{n} \; n \geq 1 \;\wedge\; \forall 1 \leq i < n \;\; e_{i+1}.start = e_i.end \tag{2}$$

The start vertex and the end vertex of P are defined as:

$$P.start = e_1.start \qquad P.end = e_n.end$$

Definition 3 (Trajectory). *The trajectory of a moving object is defined as a sequence of points with timestamps:*

$$T = (p_i)_{i=1}^{n} = (t_i, lat_i, lng_i)_{i=1}^{n}$$

We use $T[i]$ to represent the ith point of T and $\overrightarrow{p_i, p_{i+1}}$ to represent the ith trajectory segment of T. $\overrightarrow{p_i, p_{i+1}}$ is defined as the directed line segment formed by p_i and p_{i+1}.

Definition 4 (Matching Result). *Given a trajectory $T = (p_i)_{i=1}^{n}$, the matching result R is defined as:*

$$R = (p_i, e_i)_{i=1}^{n} \tag{3}$$

where e_i denotes the edge that p_i be matched to.

Definition 5 (Matching Accuracy). *Given the matching result $R = (p_i, e_i)_{i=1}^{n}$ and the ground truth $R_t = (p_i, \hat{e}_i)_{i=1}^{n}$, the accuracy A is defined as:*

$$A = \frac{\sum_{i=1}^{n} \theta_i}{n} \qquad \theta_i = \begin{cases} 1 & \text{if } R.e_i = R_t.\hat{e}_i \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Problem Statement The **Map-matching** problem can be described as, given a road network G and a trajectory T, how to efficiently find a matching result R, with the goal of maximizing the matching accuracy A.

4 Proposed Algorithms

In this section, we first illustrate the key observations of Passby followed by the algorithm description and implementation, then we describe several novel approaches for the multi-hypothesis model in detail including hypothesis creation, evaluation and management.

Observation 1. *In most cases, if a moving object passes by both the start and the end intersections of an edge within a reasonable period of time, the object is supposed to be on that edge during the time interval.*

According to Observation 1, we can improve the efficiency of map-matching algorithm by avoiding the computation on intermediate GPS points.

Observation 2. *The representation of vertices is generally more accurate than edges especially when edges are simplified.*

The most commonly used representation of vertex and edge in a road network is point and polyline. Sometimes, the road network is simplified [17, 18], the deviations between real edges and polylines can be up to hundreds of meters due to edge simplification or edge measurement error while the representation of vertices tends to be more accurate.

4.1 Passby: A Baseline Approach

Based on above observations, we design a map-matching algorithm called Passby, which mainly depends on positions of intersections and road connectivity. In particular, this algorithm does not need either the heading information or the instantaneous speed of the vehicle from the dead reckoning device.

One of the key processes in Passby is to estimate how likely the moving object passes by a certain vertex. We measure it using the passby probability.

Definition 6 (Passby Probability). *The passby probability is defined as the likelihood that the moving object passes by the vertex v.*

$$F_p(T, v) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{(\lambda-\mu)^2}{2\sigma^2}} \qquad \lambda = d(T, v) = \min_{0<i<|T|-1} d(\overrightarrow{p_i, p_{i+1}}, v)$$

where λ is the distance between the trajectory T and the vertex v, i.e., the minimum distance between the segments of T and v. We use a zero-mean normal distribution with a standard deviation of 6.4m for the training dataset.

Algorithm 1 gives an overview of Passby. The function **init**(G, T, i) takes three arguments: the road network G, the trajectory T and the start index i, to find the first passed vertex v_c that $d(T, v_c)$ is less than a threshold σ_p, e.g., 25m, and the index of corresponding trajectory segment i_c. This can be efficiently calculated by building indices on GPS points and vertices. The function **pass**(T, v, i) takes three arguments: the trajectory T, the vertex v, and start index of trajectory segment i, and returns the index of the nearest trajectory segment from v, or 0 if no valid trajectory segment is found.

First, the algorithm performs the initialization to get the first passed vertex v_c and the corresponding index of the trajectory segment i_c. (line 1). Then the

Algorithm 1 Passby(G, T)

```
Input:Road network $G$, trajectory $T = (p_i)_{i=1}^n = (t_i, lat_i, lng_i)_{i=1}^n$
 1: $(v_c, i_c) \leftarrow$ init$(G, T, 1)$
 2: while $i_c \leq n$ do
 3:     $S_c \leftarrow \emptyset,\ S_e \leftarrow \{e | e.start = v_c\}$
 4:     for each $e \in S_e$ do
 5:         $i \leftarrow$ pass$(T, e.end, i_c)$
 6:         if $i > 0$ then
 7:             $S_c \leftarrow S_c \cup \{(i, e)\}$
 8:     if $S_c \neq \emptyset$ then
 9:         $t \leftarrow \arg\min_{t \in S_c}(t.i)$
10:         match $\{p_{i_c+1}, \ldots, p_{t.i}\}$ to edge $t.e$
11:         $i_c \leftarrow t.i,\ v_c \leftarrow t.e.end$
12:     else
13:         $(v_c, i'_c) \leftarrow$ init$(G, T, i_c)$
14:         match $\{p_{i_c+1}, \ldots, p_{i'_c}\}$ to nearest edges
15:         $i_c \leftarrow i'_c$
```

algorithm repeats the following process until the end of trajectory: 1) it retrieves all the candidate edges (line 3); 2) it filters out infeasible edges using passby probability (line 4 - line 7); 3) if valid edges exist, the edge $t.e$ with the smallest index is selected (line 8 - line 9), then the algorithm matches the points between p_{i_c+1} and $p_{t.i}$ to $t.e$ (line 10), otherwise a re-initialization will be made to find the next vertex v_c and corresponding index (line 13). The intermediate points will be matched to their nearest edges (line 14). As illustrated in Figure 1, the algorithm needs only to match the initial and the final points in most cases and is consequently very efficient.

4.2 Passby*: A Multi-Hypothesis Based Approach

The Passby algorithm works well in the premise that every passed vertices can be found and the one closest to the moving object is the real passed vertex. However, this assumption may not always hold, e.g., several vertices can be quite close to the moving object at the same time while the real passed vertex might not be close enough to be found. To generate stable matching results in ambiguous situations, we propose another algorithm called Passby*, which is integrated with the multi-hypothesis technique to maintain all the possible solutions in situations of ambiguity and eventually choose the one with the highest probability.

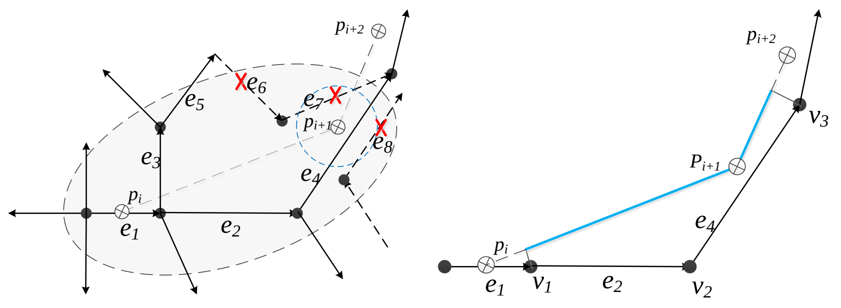
4.2.1 Hypothesis Creation

The traditional hypothesis creation method usually generates new hypotheses at each step, which will result in an exponential computational complexity. To be efficient, we redesigned the rule of hypothesis creation.

Algorithm 2 shows the pseudo-code of candidate creation process. Function **createCands**(G, T, sp) takes three arguments: the road network G, the trajectory T, and the path information structure $sp : (v, i, f, ptr)$ which contains information about the candidate path, where v is the vertex, i is the index of the trajectory segment that is nearest to the v, f is the score of the path and ptr is the pointer to the parent structure, which is used to get the whole path.

Algorithm 2 createCands(G, T, sp)

```
Input: Road network $G$, trajectory $T = (t_i, lat_i, lng_i)_{i=1}^{n}$, path structure $sp$
Output: A set of candidate path structures: $S_{sp}$
1: $S_{sp} \leftarrow \emptyset$
2: $S_v \leftarrow \{v | d(v, T[sp.i]) + d(v, T[sp.i+1]) \leq 2a\}$
3: $S_P \leftarrow$ buildPath$(G, S_v, sp)$
4: for each $P \in S_P$ do
5:     $i \leftarrow$ pass$(T, P.end, sp.i)$
6:     if $i > 0$ then
7:         $f \leftarrow sp.f \cdot F_p(T, P.end) \cdot F_t(T, P)$
8:         $S_{sp} \leftarrow S_{sp} \cup \{(P.end, i, f, sp\}$
9: return $S_{sp}$
```



(a) (b)

Fig. 2: Examples of hypothesis creation.

Take Figure 2(a) for example, the algorithm first retrieves all the edges whose start vertices lie within the error ellipse that has a major axis of $2a$ (line 2). e_6 is not a valid edge as its start vertex falls outside the ellipse. e_7, e_8 won't be considered either as they are not connected with valid edges. Then the algorithm builds candidate paths in line 3. These paths start from $sp.v$ and are extended with candidate edges based on road connectivity. Finally, the algorithm finds these candidate paths whose end vertices are considered passed by the moving object, and calculates their scores using passby probability and vertex transition probability (line 4 - line 8).

Definition 7 (Vertex Transition Probability). *The vertex transition probability is used to measure the likelihood that the moving object transfers between* *vertices, and is defined as:*

$$F_t(T,P) = \exp(-\alpha_t \frac{\sum_{e\in P} \text{len}(e)}{\sum_{e\in P} \text{len}(\text{proj}(e,T))}) \tag{5}$$

where α_t is the scaling factor, len(e) returns the length of e, and proj(e, T) returns the projection of e on T. Figure 2(b) shows the candidate path $\{e_2, e_4\}$, the thick blue line represents the projection of the path on T. The efficiency of this hypothesis creation method will be discussed in Section 4.2.3.

4.2.2 Solutions to Ambiguous Situations

In practice, there exist many ambiguous situations, e.g., Y-junctions [12] and parallel paths, which pose a challenge to map-matching.

Definition 8 (Ambiguous Situation). *An ambiguous situation is defined as the situation in which there exists a path P in the candidate set, the end vertices of more than one of P's candidate edges are close to the trajectory.*

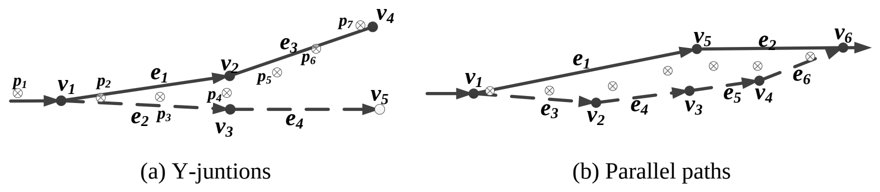


Fig. 3: Examples of ambiguous situations.

Figure 3(a) shows a case of Y-junctions problem with vertices represented by solid circles are considered possibly passed by the moving object while empty ones not, e.g., v_5. It is difficult to determine whether p_2, p_3, p_4 should be matched to e_1 or to e_2. [11] describes a solution by performing certain steps of look-ahead. However, this method is quite time-consuming especially when the number of look-ahead is large [10].

Observation 3. *In most cases, the deviation between the wrong path and the real route of the moving object tends to increase with the elapse of time.*

This simple observation is helpful to solve the Y-junctions problem. We use the example in Figure 3(a) to explain. Let P be the path to be extended, as both v_2 and v_3 satisfy the criteria, two new paths $P_1 = P \cup \{e_1\}$ and $P_2 = P \cup \{e_2\}$ will be created. As stated in Observation 3, the deviation between the real route and the wrong path P_2 increases, there is no GPS points near the candidate edges of P_2, thus it will not be extended any more, i.e., this hypothesis will be automatically discarded. However, Observation 3 will not hold in another kind of ambiguous situation called Parallel Paths.

Definition 9 (Parallel Paths). *Parallel Paths is defined as a set of paths S_{pp} satisfying the following two conditions:*

- *The maximum Hausdorff distance [19] between any two paths in S_P is lower than $2\sigma_p$.*
$$\max_{P_i, P_j \in S_{pp}} h(P_i, P_j) < 2\sigma_p$$
- *All of these paths have identical start and end vertices.*

$$\forall P_i, P_j \in S_{pp} \quad P_i.start = P_j.start \wedge P_i.end = P_j.end$$

Figure 3(b) gives an example of parallel paths. Both of these two paths start from v_1 and end at v_6. As these two paths come quite close to each other, all the vertices in the two paths can be possibly passed by the moving object, which makes it difficult to find the real path. After analyzing the relation between the cost of a path, i.e., the minimum time required to pass it, and the actual time used to go through it, we found Observation 4.

Observation 4. *The cost of the real path tends to be similar to the actual time used to go through it.*

Although the lengths of the parallel paths might be nearly the same, the differences between their costs are relatively large, e.g., the highway and the service road nearby. Therefore, we are likely to find the real path by further considering the cost similarity.

Definition 10 (Cost Similarity). *Given a path P and a trajectory T, the cost similarity $F_c(T, P)$ is defined as follows:*

$$F_c(T, P) = \prod_{e \in P} \exp(-\alpha_c |c_e - \hat{c}_e|) = \exp(\sum_{e \in P} -\alpha_c |c_e - \hat{c}_e|) \tag{6}$$

$$c_e = T[i].t - T[j].t \qquad \hat{c}_e = \frac{\text{len}(e)}{s_e}$$

where c_e is the actual time used to pass e, $\hat{c}_e$ is the cost the e, α_c is the scaling factor for cost similarity. i and j are the indices of trajectory segment that pass the start vertex and the end vertex of e, len(e) is the length of e, s_e is the speed limitation of e.

4.2.3 Management of Candidate Paths

The candidate management is mainly used to reduce the candidate size while preserving all possible solutions. It consists of two aspects: pruning and confirmation. Pruning is the process of eliminating infeasible candidate paths while confirmation is the process of confirming a candidate path as the real path.

Pruning happens in the following conditions: 1)The ratio of the score of a hypothesis to the largest score is lower than a threshold σ_A, which usually happens at Y-junctions. 2) In the case of parallel paths, all the parallel paths should be merged into a single one.

Confirmation happens in the following conditions: 1) The ratio of the largest score to the next largest one exceeds a threshold σ_B. 2) There exists only one hypothesis. σ_A and σ_B are two thresholds used to control the candidate size.

Next, we will show that the number of hypotheses will only increases in ambiguous situations. In order to increase the number of hypotheses, at least two sub-paths must be created from a path. Besides, one of the prerequisites of hypothesis creation is that the end vertex of the path must be close enough to the trajectory, i.e., the ambiguous situation.

4.2.4 Overview of Passby*

Figure 4 gives a description of the procedure of the algorithm. Figure 4(a) shows the road network and the GPS points, while Figure 4(b) demonstrates the searching process. As mentioned before, vertices represented by solid circles in figures are considered possibly passed by the moving object while empty ones not. The red cross denotes a clip of the search graph, with corresponding path being removed from the candidate set.

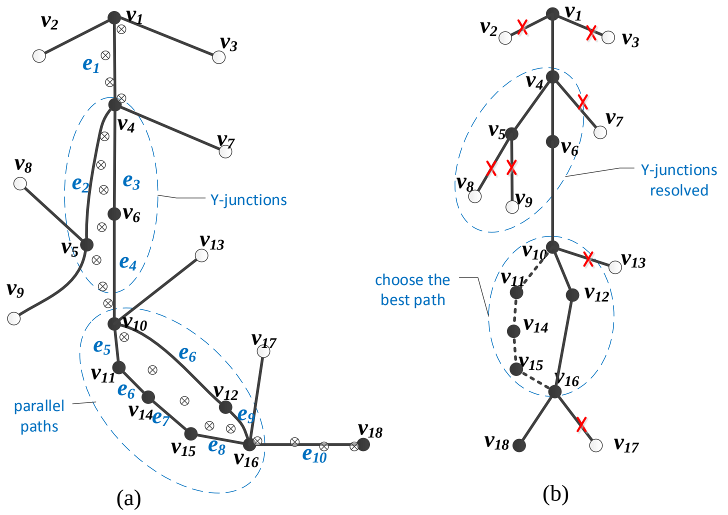


Fig. 4: An example of the Passby* algorithm.

Algorithm 3 outlines the framework of Passby*. S_O, S_P, S_{pp} are all sets of path information structures, i.e., $sp : (v, i, f, ptr)$. Function **getInitialCands**(G, T) is used to get the initial vertices and evaluate them with passby probability. Function **createCands**(G, T, sp) is used to get candidate paths with the method proposed in Section 4.2.1. Function **getBestPath**(S_{pp}, T) is used to get the set of parallel paths and further evaluate them with cost similarity to get the best path.

Algorithm 3 Passby*(G, T)

Input:Road network G, trajectory $T = (t_i, lat_i, lng_i)_{i=1}^{n}$

Output:The matching result: R

```
 1: $S_O \leftarrow \emptyset$; $sp_c \leftarrow (null, 1, 1, null)$ //$(v, i, f, ptr)$
 2: while $sp_c.i \le n$ do
 3:     if $S_O = \emptyset$ then
 4:         $S_O \leftarrow$ getInitialCands$(G, T, sp_c.i)$
 5:     while $S_O \ne \emptyset \wedge sp_c.i \le n$ do
 6:         $sp_c \leftarrow$ first$(S_O)$; $S_O \leftarrow S_O \backslash \{sp_c\}$;
 7:         $S_P \leftarrow$ createCands$(G, T, sp_c)$ //Section 4.2.1
 8:         for each $sp$ in $S_P$ do
 9:             $S_{pp} \leftarrow \{sp' | sp' \in S_O \wedge sp'.i = sp.i \wedge sp'v = sp.v\}$
10:             if $S_{pp} \ne \emptyset$ then
11:                 $sp_{best} \leftarrow$ getBestPath$(S_{pp}, T)$ //Section 4.2.2
12:                 $S_O \leftarrow S_O \backslash S_{pp} \cup \{sp_{best}\}$
13:             else
14:                 $S_O \leftarrow S_O \cup \{sp\}$
15:         $S_O \leftarrow$ pruneConfirm$(S_O)$ //Section 4.2.3
16: $sp_c \leftarrow \arg\max_{sp \in S_O}(sp.f)$
17: $R \leftarrow$ getMatchResult$(sp_c)$
18: return $R$
```

Take Figure 4(a) for example, the algorithm first gets initial paths in line 4, and then pop the first item from S_O, i.e., the item with smallest i, as sp_c (line 5). Hypothesis creation method proposed in Section 4.2.1 are used to generate candidate paths (line 6). These candidate paths are processed in line 7 to line 13. Line 8 indicates the case of parallel paths, and the best candidate is selected using method in Section 4.2.2 (line 10). The pruning and confirmation are performed in line 14. Finally, the algorithm finds the path with the maximum score in S_O (line 15), i.e., $\{e_1, e_3, e_4, e_6, e_9, e_{10}\}$, and generates the matching result (line 16).

4.3 Theoretical Analysis

Now we will analyze the complexity of the **Passby*** algorithm. Let n be the length of the trajectory T, l be the maximum number of edges in the error ellipse used in the hypothesis generation process, and k be the maximum number of candidates.

The time complexity of the function **createCands**, **pruneConform** and **getMatchResult** are $O(l\log(l))$, $O(k)$ and $O(n)$. In the worst-case, hypothesis creation might be performed on every single trajectory point, so the Passby* algorithm has the time complexity of $O(nkl\log(l) + nk + n) = O(nkl\log(l))$. In practice, k is usually quite small, thus the time complexity of Passby* is close to $O(nl\log(l))$. Furthermore, as indicated by Figure 1, large part of computation is avoid, so the constant factor is actually quite small, this is also confirmed by the experiment.

5 Experiments

In this section, we first present the experimental settings, then we evaluate the efficiency and accuracy of the proposed algorithms, finally we show their performances on the most simplified road network.

5.1 Dataset and Experimental Setup

In our experiment, we use the road network and trajectory data provided by ACM SIGSPATIAL Cup 2012 [20] , which is a GIS-focused algorithm competition hosted by ACM SIGSPATIAL. The road network graph contains 535,452 vertices and 1,283,540 road segments. Ground truth files are included in trajectory data file, which are used in results verification. These algorithms have been implemented with C++ on Visual Studio express platform, the results of the experiments are taken on a computer with Intel Core2 Dual CPU T8100 2.1 GHz and 3 GB RAM.

Baseline Algorithms We compare proposed algorithms with the incremental algorithm: IMM05 [11], and the Hidden Markov Model based algorithm: NK09 [7]. IMM05 performs matching based on three aspects: the distance, the direction and a point's previous matching result, and is known to have a low time complexity. NK09 is well-known for its high accuracy and the ability to deal with noise.

For IMM05, we use the suggested settings in [11]: $\mu_d = 10, \alpha = 0.17, n_d = 1.4, \mu_a = 10$, $n_\alpha = 4$. For NK09, we use the following assignment : $\sigma = 4.1$, and $\beta = 5$. Preprocessing and optimization suggested in [7] are made to NK09. To further speed up the algorithm, roads that are more than 50 meters away from the GPS point are eliminated from the candidate set. For Passby and Passby*, we use : $\sigma = 6.4, \sigma_p = 25, \alpha_t = 65, \alpha_c = 4, \sigma_A = 0.1, \sigma_B = 3$.

5.2 Matching efficiency with different sampling interval

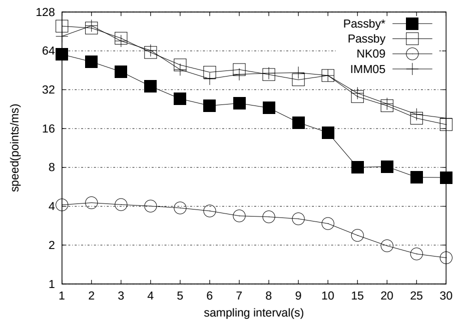


Fig. 5: Efficiency.

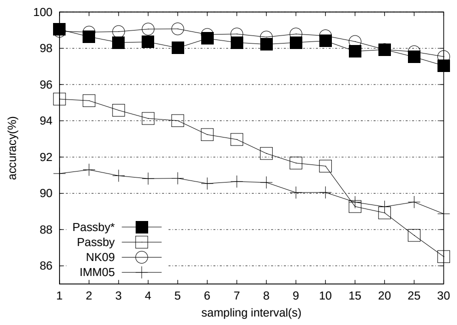


Fig. 6: Accuracy.

As shown in Figure 5, Passby* is 4∼10 times faster than the NK09, and is even comparable to IMM05 when sampling interval is small. In addition, the matching speed of Passby is a little faster than IMM05 . With the increase of the sampling interval, the speeds of Passby and Passby* begin to decrease due to the fact that less computation can be avoided.

The following reasons may contribute to the high efficiency of Passby*:

- Computation of map-matching on a large part of GPS points is avoided. In most cases, Passby* needs only to match the initial and the final points. Moreover, there is less shortest path calculation in Passby* than NK09 which is quite time-consuming.
- The refined hypothesis management method is efficient. With the hypothesis creation method proposed in Section 4.2.1, the number of hypotheses is usually quite small.

5.3 Matching accuracy with different sampling interval

As shown in Figure 6, the matching accuracy of Passby* is quite comparable to NK09, and much higher than IMM05. Meanwhile, the accuracies of all these algorithms decrease with the increase of sampling interval as the deviation between the real route of the moving object and the trajectory represented by polyline becomes larger.

In particular, the matching accuracy of Passby decreases significantly with the increase of sampling interval. This is mainly because Passby simply choose the nearest vertex as the real passed one which is less accurate when the distances between the trajectory and passed vertices become larger. In contrast, Passby* maintains all the possible paths, and eventually chooses the best one which makes it less sensitive to the increase of sampling interval. Several reasons may contribute to the high accuracy of Passby*:

- Passby* evaluates candidate path in terms of both temporal and spatial factors, e.g, passby probability, vertex transition probability and cost similarity.
- The refined multi-hypothesis model is used to effectively maintain a set of candidate paths and eventually choose the best one.
- Passby* is less dependent on the geometric detail of the edge, e.g., it needs no projection to the edge, which makes it robust to edge measurement errors.

5.4 Matching accuracy on the most simplified road network

In the most simplified road network, as mentioned in Definition 1, the edge detail, i.e., *geo*, is omitted. This will significantly reduce the size of the digital map. However, the distance between the trajectory point and the simplified edge becomes much larger, and it sometimes can be up to hundreds of meters. Figure 7 gives an example of edge simplification. To evaluate the robustness of the algorithms to edge simplification, we test the algorithms on the most simplified road network.

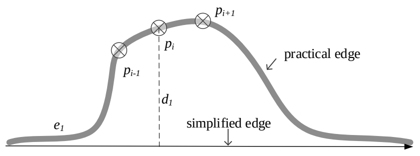


Fig. 7: Edge simplification.

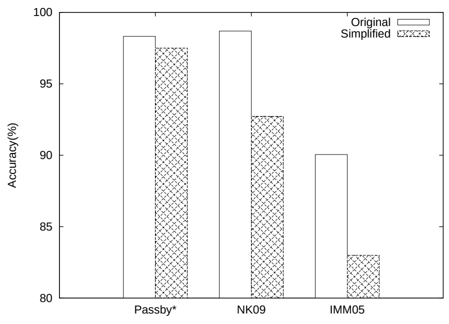


Fig. 8: Matching accuracy on the most simplified road network.

Figure 8 illustrates the matching accuracies on both the original and the most simplified road network when the sampling interval is 10s. The accuracy of Passby* merely changes while the accuracies of other two algorithms suffer from significant decreases. This is because Passby* matches the trajectory mainly based on intersections and is less dependent on edge detail information.

6 Conclusion and Future Work

In this paper, we investigate the problem of how to improve the efficiency of map-matching, and propose two novel algorithms called Passby and Passby*. They perform map-matching according to intersections that the moving object passes by. Taking advantage of the proposed candidate generation and evaluation methods, Passby* manages to efficiently provide accurate matching results in ambiguous situations. In addition, Passby* is less dependent on the edge detail information, and is robust to edge measurement error and edge simplification. We conduct experimental study on real dataset to evaluate the performance of proposed algorithms. The results of the experiments indicate that Passby* is both efficient and accurate.

In the future work, we plan to further improve the algorithm to better process GPS data of low sampling rate.

Acknowledgements

This work was supported by the National Natural Science Foundation of China (Nos. 61202064, 91124001), the Strategic Priority Research Program of the Chinese Academy of Sciences (No. XDA06020600), and the Key Research Program of the Chinese Academy of Sciences (NO. KGZD-EW-102-3-3).

References

1. Liu, K., Deng, K., Ding, Z., Li, M., Zhou, X.: Moir/mt: Monitoring large-scale road network traffic in real-time. PVLDB **2**(2) (2009) 1538–1541

2. Gonzalez, H., Han, J., Li, X., Myslinska, M., Sondag, J.P.: Adaptive fastest path computation on a road network: a traffic mining approach. In: VLDB. (2007) 794–805
3. Li, X., Han, J., Lee, J.G., Gonzalez, H.: Traffic density-based discovery of hot routes in road networks. In: SSTD, Berlin, Heidelberg (2007) 441–459
4. White, C.E., Bernstein, D., Kornhauser, A.L.: Some map matching algorithms for personal navigation assistants. Transportation Research Part C: Emerging Technologies **8**(1-6) (2000) 91–108
5. Greenfeld, J.S.: Matching gps observations to locations on a digital map. In: Transportation Research Board. Meeting, Washington, D.C. (2002)
6. Lou, Y., Zhang, C., Zheng, Y., Xie, X., Wang, W., Huang, Y.: Map-matching for low-sampling-rate gps trajectories. In: GIS, Seattle, Washington (2009) 352–361
7. Newson, P., Krumm, J.: Hidden markov map matching through noise and sparseness. In: GIS, Seattle, WA, USA (2009) 336–343
8. Zheng, K., Zheng, Y., Xie, X., Zhou, X.: Reducing uncertainty of low-sampling-rate trajectories. In: ICDE, Washington, DC, USA (2012) 1144–1155
9. Pink, O., Hummel, B.: A statistical approach to map matching using road network geometry, topology and vehicular motion constraints. In: ITSC,IEEE. (2008) 862 –867
10. Wenk, C., Salas, R., Pfoser, D.: Addressing the need for map-matching speed: Localizing globalb curve-matching algorithms. In: SSDBM, Washington, DC, USA (2006) 379–388
11. Brakatsoulas, S., Pfoser, D., Salas, R., Wenk, C.: On map-matching vehicle tracking data. In: VLDB, Trondheim, Norway (2005) 853–864
12. 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 **15**(5) (2007) 312–328
13. Syed, S., Cannon, M.: Fuzzy logic based-map matching algorithm for vehicle navigation system in urban canyons. In: National Technical Meeting of The Institute of Navigation, San Diego, CA (2004) 982–993
14. Reid, D.: An algorithm for tracking multiple targets. Automatic Control, IEEE Transactions on **24**(6) (1979) 843 – 854
15. Pyo, J.S., Shin, D.H., Sung, T.K.: Development of a map matching method using the multiple hypothesis technique. In: Intelligent Transportation Systems, 2001. Proceedings. 2001 IEEE, Oakland, CA, USA (2001) 23 –27
16. Abdallah, F., Nassreddine, G., Denoeux, T.: A multiple-hypothesis map-matching method suitable for weighted and box-shaped state estimation for localization. IEEE Transactions on Intelligent Transportation Systems **12**(4) (2011) 1495–1510
17. Liu, K., Li, Y., He, F., Xu, J., Ding, Z.: Effective map-matching on the most simplified road network. In: GIS, Redondo Beach, CA, USA (2012) 609–612
18. Zhou, J., Golledge, R.: A three-step general map matching method in the gis environment: travel/transportation study perspective. International Journal of Geographical Information System **8**(3) (2006) 243–260
19. Alt, H., Guibas, L.: Discrete geometric shapes: Matching, interpolation, and approximation. In: Handbook of Computational Geometry, Amsterdam (1999) 121–153
20. ACM SIGSPATIAL Cup 2012: Training data sets. (2012) Available from http://depts.washington.edu/giscup/home.