Effective Map-matching on the Most Simplified Road Network

Kuien Liu [†] Yaguang Li [†§] Fengcheng He [†§] Jiajie Xu [†] Zhiming Ding [†]
[†] Institute of Software, Chinese Academy of Sciences, Beijing 100190, China
[§] University of Chinese Academy of Sciences, Beijing 100049, China
{kuien,yaguang,fengcheng,jiajie,zhiming}@nfs.iscas.ac.cn

ABSTRACT

The effectiveness of map-matching algorithms highly depends on the accuracy and correctness of underlying road networks.In practice, the storage capacity of certain hardware, e.g. mobile devices and embedded systems, is sometimes insufficient to maintain a fat digital map for map-matching. Unfortunately, most existing map-matching approaches consider little about this problem. They only apply to environments with information-rich maps, but turns out to be unacceptable for map-matching on simplified road maps. In this paper, we propose a novel map-matching algorithm called *Passby* to work on most simplified road networks. The storage size of digital road map in disk or memory can be greatly reduced after the simplification. Even under the most simplified situation, i.e., each road segment only consists of a couple of junction points and omits any other information of it, the experimental results on real dataset show that our Passby algorithm significantly maintain high matching accuracy. Benefiting from the small size of map, simple index structure and heuristic filter strategy, Passby improves matching accuracy as well as efficiency.

Categories and Subject Descriptors

H.2.8 [**Database Applications**]: Spatial databases and GIS

General Terms

Algorithms, Performance

Keywords

GPS, Map-matching, Simplified Road Network

1. INTRODUCTION

Map matching aims at relating a sequence of (possibly imprecise) location measurements to a spatial road network and identifying the correct road segment on which the vehicle is travelling. It is a fundamental pre-processing step for many applications, such as moving object management [4], vehicle navigation [8], traffic conditions monitoring [5] and geographical social network.

The effectiveness of map-matching algorithms highly depends on the accuracy and correctness of underlying road networks. There are a number of studies on map-matching [2, 3, 7, 6]. Most of them assume that the digital road network data used for map-matching should be of a large scale and well featured in order to generate matching outputs with fewer errors. In practice there exists many cases that lack of storage capacity to maintain a fat digital map for map-matching, e.g., mobile devices or embedded systems. A practical solution for such cases is to simplify the road network to reduce its storage size. However, effective map-matching on the simplified road network is challenging because of the incompleteness of topology information.

Unfortunately, most existing map-matching approaches is not applicable to simplified maps, and produce unacceptable matching accuracy. After simplification, many features become no longer available for map-matching. For example, large curves and roundabouts might be represented by a relatively short line segment (or two sequential points). With simplified road network, many commonly used criterions for map-matching may not be useful any more. Figure 1 illustrates an example of matching error on simplified road network, in which roads e_1 and e_2 are simplified as directed lines. Given a vehicle moving along road e_1, and reported the current GPS position p_i, take the geometric measures of both deviation distance ($d_2 \ll d_2$) and orientation similarity ($\theta_2 \ll \theta_1$) into account, it is easy to match p_i to the incorrect road e_2 rather than the true road e_1.

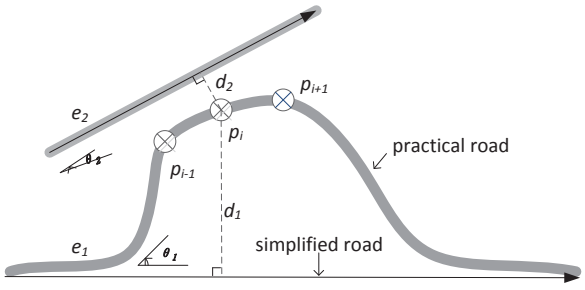

Figure 1: An example of simplified road network.

In addition, as the dependency between current or neighboring positions and nearby roads decreases, less information can be used to deduce the precise road network loca-

tions of the object. The problem is aggravated when the GPS data is sampled in low frequency, or there are many road links around these points.

1.1 Most Simplified Road Network

In this paper we refer to simplified road networks as simple graph with no more attributes, i.e., a graph consists only intersection coordinates and topological relation. In a classic road network, each road segment includes three parts of features: 1) start/end junctions, 2) a route (usually be expressed by a polyline) from start junction to end junction, and 3) design parameters, such as speed limitation, turn restrictions at junctions, roadway classification (such as one-way, two-way), width of the carriageway, number of lanes, and overpass and underpass information etc. Under the most simplified situation, each road segment only consists of a couple of junction points (part 1) and omits any other information of it (parts 2, 3). Table 1 illustrate the difference of edge representation between original road map and the most simplified road map.

Table 1: Original road edge vs. most simplified one

Type	Attributes
Original	$id, name, type, length, speed\ limit$ $width, start, p_1, p_2, ..., p_n, end, etc.$
Simplified	$id, start, end$

In this paper, we focus on the most simplified road network. Below lists necessary principles for problem statement.

Definition 1. A **trajectory** T is a sequence of GPS points $\{p_1, p_2, ..., p_n\}$, where each GPS point p_i consists of a triple $< t, x, y >$ and $p_{i-1}.t < p_i.t$.

Definition 2. A **most simplified road intersection** v is a geometrical point that is associated with only an id $v.id$, a geometrical position $v.lat$ and $v.lng$.

Definition 3. A **most simplified road segment** e is a directed edge that is associated with only an id $e.id$, a starting point $e.start$ and an ending point $e.end$.

Definition 4. A **most simplified road network** G is a directed graph G(V, E), where V is a set of vertices representing the intersections and terminal points of the road segments, and E is a set of edges representing the most simplified road segments.

Now the problem of map-matching is defined as:
Given a trajectory T *and a most simplified road network* G(V, E)*, find the edge e from* G *that matches p(p ∈ T).*

In this paper, we propose a novel map-matching algorithm called *Passby* for simplified road networks. Our main contributions are listed as below:

- We study the problem of map-matching on the most simplified road network. To the best of our knowledge, this paper is the first work in this field.
- We propose the Passby algorithm to deal with map simplification. Its basic idea is to efficiently determine whether the trajectory passes by a simplified road segment no matter what practical road looks like.
- We perform experiments on real dataset, which verifies the feasibility and effectiveness of Passby algorithm .

2. PROPOSED METHOD: PASSBY

The major difference of map-matching on simplified road networks and on original ones is how we measure the similarity between trajectory and road segments. Traditional measures in current existing approaches do not apply for the context of simplification maps. For example, the distance proximity and orientation similarity, δ and θ in Figure 1, seem to be rather misleading and unhelpful for road edge determination, because road edges (e.g., e_1) in simplified map may be quite different from those in practice. We need effective methods to decrease the uncertainty in simplified map.

An interesting observation is that a vehicle's motion is much more certain when it passes by a road junction. In other words, whether a vehicle's position can be matched to a road edge may be partly deduced by finding out whether this vehicle passed by (one or both) intersections of this road. Following this idea, this paper proposes the *Passby* algorithm aiming at the most simplified road network.

Algorithm 1 The **Passby** Algorithm

```
1: input: a trajectory and a simplified road network
2: repeat
3:   if p_{i-1} and p_i pass junction e.start then
4:     if p_j and p_{j+1} pass junction e.end then
5:       match points {p_i, ..., p_j} to edge e;
6:       next loop starts from p_{j+1};
7:     else
8:       get candidate edges traversing from p_{i-1} to p_i;
9:       match p_i to e in weighted ranking way;
10:      next loop starts from p_{i+1};
11:    end if
12:  end if
13:  match remaining points with existing algorithm;
14: until ALL points are processed
```

As Algorithm 1 shows, Passby carries out the map-matching process with three cases. The first case is that the trajectory passes by both junctions of a road, then mid-points between two junctions can be matched to the same road without individual check. It is really useful for GPS in high sampling rate ($\geq$ 0.1Hz). The second case is that the trajectory only passes by the starting junction of a road, then we generate the candidate edges that next point may move on. It is more common for low sampling trajectories. The last case is the remaining points (including cold-start points), whose passing-by semantics are not that intuitive, we employ current existing methods to match them, e.g., the incremental algorithms.

The index structures used in Passby are simple but effective. We only need to build one index on road junctions (rather than on road edge in traditional approaches) and the other one on trajectory with point index, e.g., the grid which is also well suited for parallel processing further. Next, we will briefly introduce the major components of Passby: ① how to determine the passing-by (in line 3 and line 4) and ② how to choose the candidates (line 8).

2.1 How to measure a vehicle passing by a road junction

Given two successive GPS points p_{i-1}, p_i and a candidate edge e, the question of whether p_{i-1}, p_i passing by $e.start$ (line 3 in Algorithm 1) can be evaluated by weighted formula with following four measures:

- d_p: the projection distance from point p_i to edge e;
- θ_i: the intersection angle between line $\overrightarrow{p_{i-1}, p_i}$ and e;
- d_t: the traversing distance projected from junction $e.start$ to line $\overrightarrow{p_{i-1}, p_i}$;
- θ_t: the traversing angle between line $\overrightarrow{e.start, p_i}$ and $\overrightarrow{p_{i-1}, p_i}$.

As the road edge e may be simplified a lot, we do not know its actual length with any certainty. Furthermore, temporal/speed constraints of the trajectories are hard to enforce in the map-matching process. Figure 2 illustrates an example of these four measures. For simplified roads, smaller traversing distance and angle give better proximity. Finally, the combined passing-by score may be computed as the weighted sum of individual measures.

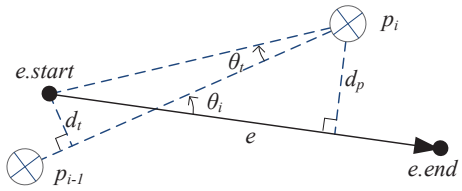

Figure 2: Four measures used in Passby.

2.2 How to choose the road candidates a vehicle move next

Existing geometric methods (in Euclidean space), e.g., the error ellipse [2], are not always suitable for map-matching in simplified road network, during to two reasons: 1) the map simplification would cause loss-of-answer at times; 2) the error may be accumulated from previous steps, e.g., the Y-junction problem [7]. With these considerations, we employ a topological and Euclidean hybrid method to choose the road candidates for matching next point (p_i in Figure 3).

The traversing range between last matched point and current point is constrained by two factors. One is the maximum moving range (r in Figure 3) around current point. The other is the maximum accumulative length of traversing path (e.g., $r+|e_1|$) starting from last matched point. We call the former as Euclidean range and the latter topological range. Figure 3 gives us an example on this. The region intersected by both ranges significantly restricts the number of candidate edges (in bold black).

The topological range is stepwise generated. As the example in Figure 3 shows, we first look backward, and locate the upstream junctions where the vehicle comes from (e.g., $e_1.start$ for p_{i-1}). The step helps to partly solve the Y-junction problem [7]. Then, from these upstream junctions, we look forward along the road network topology till out of the reasonable traversing range, and keep all the intermediate edges into candidate set. Once the candidate region is constructed, we can rank them using weighted measures listed in section 2.1 and finally get the matching result.

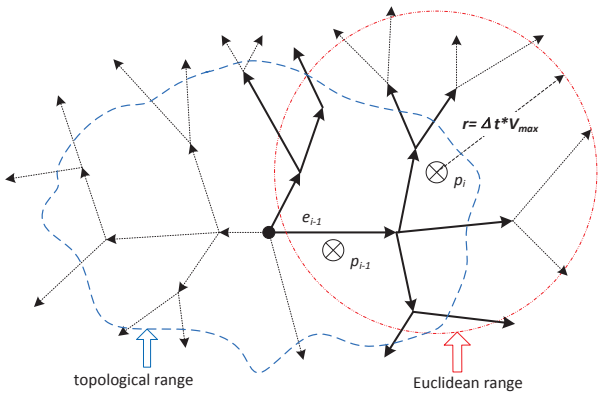

Figure 3: Candidate region construction in Passby.

2.3 Some supplementary mechanisms

This section gives out several supplementary mechanisms to fasten the map-matching process while guaranteeing its correctness.

2.3.1 *Fast Angle Calculation*

Angle calculation is the foundational function for all of the measures in section 2.1, e.g., the traversing distance can be calculated by $d_t = \sin\theta_t \times |\overrightarrow{e.start, p_i}|$. To speed up the calculation, we import arctan lookup table and sin lookup table which are organized in hash containers. With this design, it is quite easy to get the angle for a given vector. Then we can get the intersection angle of two vectors by subtraction.

2.3.2 *Parallelized Matching Process*

To make best use of the advanced multi-core technologies in modern computers, Passby algorithm is designed following the idea of parallelism. First, the digital map data is loaded concurrently, each thread takes charge of an (near) equivalent part of map data. To avoid extensive collision in exclusive operations and time cost in mutual locks, we create a global memory pool and share it with every thread. Furthermore, we delay the maintenance of topological relationship (e.g., indegree/outdegree of each junction) from loading step to matching step, and perform these operations on need. In the same way, we implement the index building and map-matching process.

2.3.3 *Outlier Identification*

The approach of candidate graph construction in section 2.2 is a little sensitive to outliers and depends on correctness of pre-matching result which may be incorrect. For example, a vehicle is at a standstill for a short time, two sequentially matched road edges are not reachable under reasonable constraints, cold-start points may not be unique (e.g., passing through the tunnel is treat as a new trip). By identifying the outlier, we can further improve the accuracy of Passby.

3. EXPERIMENTAL RESULTS

In this section we first present the experimental settings, then we report the overall effectiveness of Passby algorithm on simplified road network in terms of map size and running time, and finally we study the matching accuracy of Passby

on trajectories in different sampling rates.

In our experiments, we use the road network and data sets provided by ACM SIGSPATIAL Cup 2012 [1], which is a GIS-focused algorithm competition hosted by ACM SIGSPATIAL. The network graph contains 535,452 vertices and 1,283,540 road segments. The data set used in our experiments contains 20 files: 10 trajectory files (14,436 GPS points in total) are used for testing and each file contains a GPS trace route of an individual trip, while the other 10 files are used as the ground truth in results verification. Our implementation is written in C++ on Visual Studio express platform, experimental results were taken on a computer with Intel Core2 Dual CPU T8100 2.1 GHz and 3 GB RAM.

To evaluate our experimental results, we compare Passby with incremental algorithm, which has been widely used as its low time-complexity [6]. The incremental algorithm tries to find local match of geometries, and performs matching based on a point's previous matching result. Comparing with other approaches, such as global methods or statistical methods, local/incremental algorithm achieves better performance with a little lower matching accuracy. It is similar to our motive and objective in Passby.

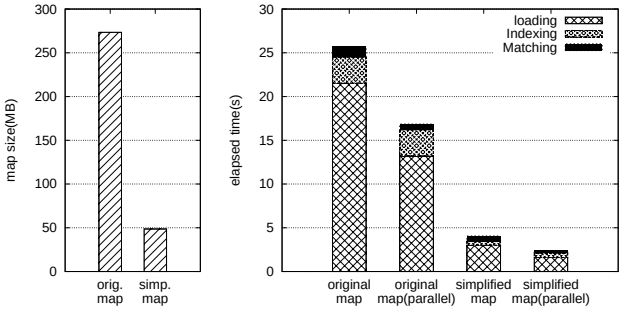

Figure 4: Effectiveness of Passby on simplified map.

As shown in Figure 4, the map size after simplification reduces to 17.7% as the original one, while the overall elapsed time reduces to 15.8%(14.1%) for sequential(parallel) mode. It demonstrates the effectiveness of our work on simplified road network, i.e., the requirement of memory capacity for map-matching is reduced as well as computing power.

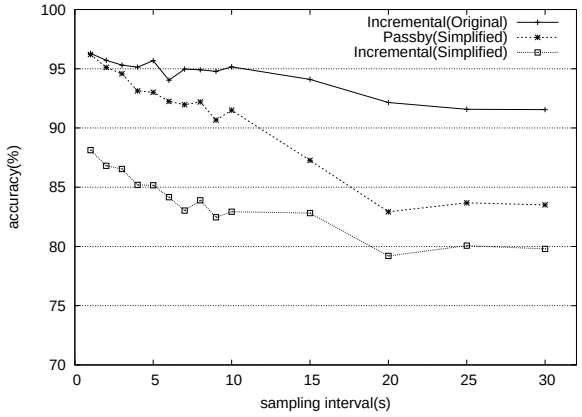

Figure 5: Accuracy w.r.t different sampling rates.

Figure 5 demonstrates the change of matching accuracy w.r.t to different sampling rates extracted from real data. It can be seen clearly that our Passby algorithm outperforms the incremental algorithm on simplified road network, for example, the matching accuracy of Passby algorithm is up to 8% better than the incremental algorithm for the frequent sampling cases (rate over 0.1Hz). We also notice that, as the sampling rate decreases, the accuracy of Passby drops in the same trend as incremental algorithm, leading to near consistent 4%. This implies that traversing measures become less effective when neighboring GPS points are far away from each other.

4. CONCLUSIONS

In this paper, we propose a new offline map-matching algorithm called *Passby* to match GPS data onto a simplified digital map. The experiment results demonstrate that our Passby algorithm achieves exciting effects compared to the incremental algorithm. Meanwhile, benefiting from the small size of map, simple index structures and heuristic filter strategy, *Passby* improves matching accuracy as well as efficiency.

5. ACKNOWLEDGMENTS

We wish to thank the SIGSPATIAL CUP organizers for their efforts and help during the match. This work was supported by the National Natural Science Foundation of China (Nos.91124001, 60970030, 61003028 and 61202064) and the Strategic Priority Research Program of the Chinese Academy of Sciences (No. XDA06020600).

6. REFERENCES

[1] Acm sigspatial cup 2012 training data sets, Feb. 2012. Available from http://depts.washington.edu/giscup.

[2] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk. On map-matching vehicle tracking data. In *VLDB*, pages 853–864, Trondheim, Norway, 2005.

[3] Z. Ding and K. Deng. Collecting and managing network-matched trajectories of moving objects in databases. In *DEXA (1)*, pages 270–279, 2011.

[4] Z. Ding and R. H. Güting. Uncertainty management for network constrained moving objects. In *DEXA*, pages 411–421, Zaragoza, Spain, 2004.

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

[6] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang. Map-matching for low-sampling-rate gps trajectories. In *ACM SIGSPATIAL GIS*, pages 352–361, Seattle, Washington, 2009.

[7] M. A. Quddus, W. Y. Ochieng, and R. B. Noland. Current map-matching algorithms for transport applications: State-of-the art and future research directions. *Transportation Research Part C: Emerging Technologies*, 15(5):312–328, 2007.

[8] J. Xu, L. Guo, Z. Ding, X. Sun, and C. Liu. Traffic aware route planning in dynamic road networks. In *DASFAA (1)*, pages 576–591, 2012.