HyMU: A Hybrid Map Updating Framework

Tao Wang, Jiali Mao, and Cheqing Jin(✉)

School of Data Science and Engineering,
School of Computer Science and Software Engineering,
East China Normal University, Shanghai, China
{toy_king,jlmao1231}@stu.ecnu.edu.cn, cqjin@sei.ecnu.edu.cn

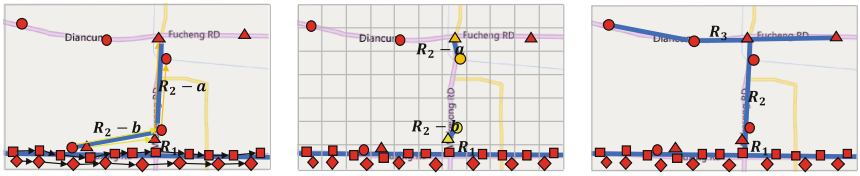
Abstract. Accurate digital map plays an important role in mobile navigation. Due to the ineffective updating mechanism, existing map updating methods cannot guarantee completeness and validity of the map. The common problems of them involve huge computation and low precision. More importantly, they scarcely consider inferring new roads on sparse unmatched trajectories. In this paper, we first address the issue of finding new roads in sparse trajectory area. On the basis of sliding window model, we propose a two-phase hybrid framework to update the digital map with inferred roads, called HyMU, which takes full advantage of line-based and point-based strategies. Through inferring road candidates for consecutive time windows and merging the candidates to form missing roads, HyMU can even discover new roads in sparse trajectory area. Therefore, HyMU has high recall and precision on trajectory data of different density and sampling rate. Experimental results on real data sets show that our proposal is both effective and efficient as compared to other congeneric approaches.

1 Introduction

With the widespread use of onboard navigators and smart phones, the accuracy of navigation map has aroused universal concern. An inaccurate road map with disconnected and misaligned roads may make the experienced drivers get lost and even cause traffic accidents. Essentially, the accuracy and completeness of a digital map depend on whether road information is updated timely and effectively. However, such a task is difficult to achieve due to two factors, one is the rapid development of road construction, and the other is the ineffective map updating mechanism. Specifically, rapid construction of roads has increased great difficulties to timely update of digital map. Massive amount of roads all over the world change every year. According to the reports by the Ministry of Transport of China[1], 4,500 km of new expressways will be built in China, and 29 road projects will be pushed forward in Shanghai in 2016. While at the same time, existing techniques cannot guarantee the timeliness of map updating. The commercial map companies update digital map by periodically conducting geological survey of the entire road network. To cut down the overall cost, survey

[1] http://www.chinahighway.com/.

S. Candan et al. (Eds.): DASFAA 2017, Part II, LNCS 10178, pp. 19–33, 2017.
DOI: 10.1007/978-3-319-55699-4_2



(a) Line-based method (b) Point-based method (c) Hybrid method

Fig. 1. An example of point-based and line-based method

period is quite long and thus the map updating rate lags far behind the construction of new roads. Alternative mechanism is to adopt the crowdsourced map project to generate customized map (e.g. OpenStreetMap), but it largely depends on the geographic data directly provided by volunteers. As a result, the amount of users and even the editing skills of users greatly influence the quality of map updating. As mentioned above, it is desirable to devise a low-cost but high reliable map updating mechanism.

Huge amount of trajectory data of vehicles can be applied to update the map. Recently, a few researches have been done in map updating with trajectories [13,16,18], and they can be grouped into two classes: line-based strategy (e.g., CrowdAtlas [16]) and point-based strategy (e.g., Glue [18] and COBWEB [13]). To be specific, the former is to infer the missing roads for a given map based on clustering considerable volume of unmatched trajectory segments, and the latter on massive unmatched trajectory points. These methods still face a series of problems, such as high computational overhead, low accuracy of inferred roads, and bad timeliness of map updating, etc. Moreover, line-based strategy has poor performance in processing low-sampling data (sampling interval longer than 30 s [18]), because it may infer the roads with false directions when the line segments cross over several roads. Although point-based methods can overcome this issue, they easily infer some short road segments rather than long roads due to the low coverage caused by point-based clustering. As shown in Fig. 1(a), two consecutive sampling points that are located on two roads are connected as a line segment, and accordingly an incorrect road R_2-b is inferred by line-based strategy. Though point-based strategy solves the above deficiencies, it infers two short road segments, R_2-a and R_2-b, instead of a long road that covers them, as illustrated in Fig. 1(b). Thus, the inferred roads in Fig. 1(a) and (b) are incorrect. To improve the inferring accuracy and obtain the ideal result in Fig. 1(c), it necessitates a hybrid framework to integrate virtues of both line-based and point-based strategies.

Furthermore, the aforementioned map updating mechanisms focus on discovering the missing roads on trajectory data of dense areas. They usually define a threshold of minimum clustering quantity standard, and cluster unmatched trajectory line segments (or points) to infer new roads only when satisfying a specific threshold. Hence, for the top road region with sparse positional points in Fig. 1(a) and (b), both line-based and point-based strategies cannot infer the

road R_3 in Fig. 1(c). Actually, we can obtain two insights from the observation of trajectories. When the new roads first come into service, relatively few vehicles will drive along them and thus the track data are more sparse than that of normal roads. Distinct from noisy data, sparse trajectories appear on such roads in many days, i.e., the amount of trajectories will not increase tremendously in a short time period. If simply lowering the threshold of aforementioned methods, noisy data may also be clustered and some incorrect roads can be inferred. Thus, both methods are not tailored to inferring new roads in sparse trajectory area. Given the two insights above, on the basis of sliding window model, we propose a two-phase road inferring framework, including candidate generation and missing roads inferring, called HyMU. Additionally, we employ a hybrid scheme to enhance the accuracy of map updating by integrating line-based and point-based strategies. Specifically, the contributions of this paper are summarized below.

- We first address the issue of new roads inferring on sparse trajectory data to improve the overall inferring precision.
- Based on the sliding window model, we take full advantage of line and point-based strategies, and propose a two-phase hybrid framework to update the map, called HyMU.
- We compare our proposal with other congeneric approaches by conducting substantial experiments on real data sets. Experimental results show that HyMU method has good inferring performance on trajectory data under different sampling rate and density.

The remainder of this paper is organized as follows. Section 2 reviews the most related work. In Sect. 3, the preliminary concepts are introduced and the problem is defined formally. In Sect. 4, we outline and analytically study the details of HyMU framework. In Sect. 5, a series of experiments are conducted on real datasets to evaluate our proposal. Finally, we briefly conclude this article in Sect. 6.

2 Related Work

In this section, we briefly conduct a systematic review over the related work in two relevant areas: map inference and map updating.

2.1 Map Inference

Based on the track data set or satellite images, map inference aims to infer the entire road map. Image processing technology is mainly applied to infer map from satellite images [10,12]. But it is costly to obtain high-resolution satellite images data. Therefore, most researches on map inference are based on the trajectories of vehicles and they can be divided into three classes: K-means [1,5,11], KDE algorithm [2,4], and trace merging algorithm [3,8]. Nevertheless, most approaches have poor performance in handling trajectory data with excessive random noise, nonuniform distribution, and uneven sampling rate. Additionally, they are too time-consuming to fit online map inference.

2.2 Map Updating

Compared with time overhead of inferring the whole map, simply adding or modifying the roads for a given map is more realistic. Map updating methods are to discover missing roads to update a given map based on unmatched trajectories, including CrowdAtlas [16], Glue [18] and COBWEB [13]. CrowdAtlas consists of four stages: trajectory clustering, centerline fitting, connection and iteration. When the number of unmatched trajectory segments reaches the specified quantity criterion, CrowdAtlas implements clustering and polyline fitting functions to generate the centerlines that represent new roads. However, CrowdAtlas may infer new roads with false directions when unmatched trajectory segments cross over two or more roads, as illustrated in Fig. 1(a). In addition, CrowdAtlas obtains poor accuracy when dealing with low sampling rate data. To improve the precision of inferred roads on low sampling rate trajectory data, Glue clusters the unmatched trajectory points to infer new roads. Similarly, COBWEB organizes the GPS points using a Cobweb data structure and reduces the vertices and edges from Cobweb to generate Road-Tree, and finally finishes map updating. Nevertheless, both Glue and COBWEB cluster unmatched trajectory points and easily infer incomplete road segments instead of intrinsically long roads, as illustrated in Fig. 1(b). Moreover, all the above mentioned approaches cannot infer the new roads based on sparse trajectory data. Thus, it necessitates devising a map updating mechanism with high precision and noise tolerance to online infer the new roads on trajectory data of various sampling rate and density.

3 Problem Definition

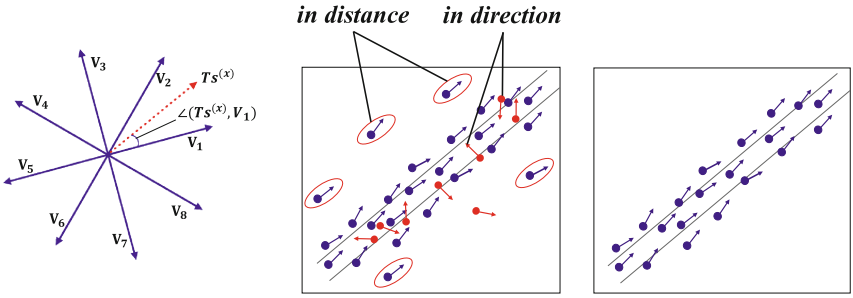
In this section, we introduce preliminary concepts, and formally define the problem of map updating upon trajectory data.

A complete digital map contains road type, geometry, turn restriction, speed limit, etc. We aim to find the roads that have not been marked on the map. The road network G that corresponds to the map is defined as follows.

Definition 1 (Road Network). *A road network is denoted by a graph* $G = (V, E)$*, where* V *is a set of vertexes and* E *refers to a set of edges. Each edge* $e \in E$ *represents a road segment.*

To infer the missing roads, we need to cope with the continuously arrived trajectories. The trajectory of an object that consists of a series of points is defined below.

Definition 2 (Trajectory). *The trajectory of a moving object, denoted as* Tr*, consists of a sequence of points,* (p_1, t_1)*,* $(p_2, t_2), \cdots$*, where* p_i *is the position at* t_i*. Such records arrive in chronological order, i.e.,* $\forall i < j$*,* $t_i < t_j$*. A trajectory segment is a line segment between two adjacent trajectory points, which is denoted as* $Ts = (p_i, p_{i+1})$*.*



(a) Denoising in direction (b) Noisy data (c) After denoising

Fig. 2. An example of denoising in distance and direction (Color figure online)

Trajectory data are collected in real-time with massive scale. In order to describe the portions of trajectories in different time periods, we employ the sliding window model, and a trajectory in a time window is denoted as Tw. Given a window size N, the window range at timestamp t_0 is $(t_0, t_0 + N)$. Hereafter, we infer the missing road candidates based on the trajectories in each time window.

Due to different resolutions of various GPS-enabled equipments and city canyon surrounded by high-rise buildings, trajectory data are noisy. According to our observation, noisy data often behave abnormally in direction or distance relative to its neighborhood. The neighborhood of a trajectory segment $Ts^{(x)}$ is defined as follows.

Definition 3 (Trajectory Segment Neighborhood). *Given a trajectory segment $Ts^{(x)}$, a distance threshold th_{dis}, and a set of trajectory segments TS, if we denote $dist(Ts^{(x)}, Ts^{(y)})$ as the shortest Euclidean distance between any two points in two line segments, the neighborhood of $Ts^{(x)}$ is defined as follows:*

$$Nd(Ts^{(x)}) = \{Ts^{(y)} \in TS | dist(Ts^{(x)}, Ts^{(y)}) \leq th_{dis}\}$$

Correspondingly, the neighborhood of a trajectory point p_i is denoted as $Nd(p_i)$, which represents the set of points that their distances to p_i are within a distance threshold th_{dis}. Subsequently, we define noisy trajectory segment as below.

Definition 4 (Noisy Trajectory Segment). *Given a trajectory segment $Ts^{(x)}$ and the directions' distribution of its surrounding segments $U(Ts^{(x)})$, $Ts^{(x)}$ is noisy if $Nd(Ts^{(x)})$ is empty or the direction of $Ts^{(x)}$ does not tally with the top-k most popular directions of its surrounding segments.*

We take the starting point of $Ts^{(x)}$ as center and the length of $Ts^{(x)}$ as radius of a circle, and generate a region. For example, in Fig. 2(a), we divide the region into 8 pieces representing 8 sector [6]. The distribution is represented as below.

$$U(Ts^{(x)}) = (C_1, C_2, C_3, C_4, C_5, C_6, C_7, C_8)$$

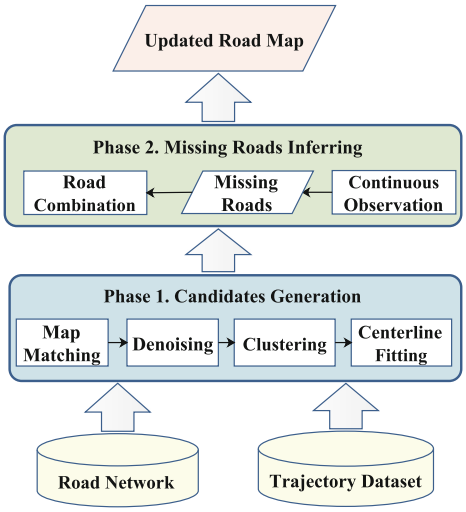


Fig. 3. The framework of HyMU

where C_i records the number of trajectory segments of $Nd(Ts^{(x)})$ that belong to the *ith* direction. Considering that each road usually has at least two lanes with opposite directions, we decide whether $Ts^{(x)}$ is a noisy trajectory segment by calculating whether $Ts^{(x)}$ belongs to the top two most popular directions. For example, in Fig. 2(a), for a trajectory segment $Ts^{(x)}$, we can determine which direction the trajectory segment $Ts^{(x)}$ belongs to according to the angle range between $Ts^{(x)}$ and V_1, denoted as $\angle(Ts^{(x)}, V_1)$.

Besides, each inferred road is represented by a road centerline.

Definition 5 (Road Centerline). *A road centerline, denoted as Rc, is represented by a polyline. It consists of a sequence of continuous positional points,* $(p_1, p_2, \ldots, p_n)$*, where* p_i *is the geographical position.*

Finally, we summarize the problem as below.

Given a road network G and a set of trajectories in different time periods, our goal is to infer the missing roads as early as possible, and then update the road network G by using the inferred missing roads.

4 Framework

In this section, we introduce a novel framework, which is called Hybrid Map Updating (HyMU). HyMU is to identify missing roads based on trajectory data. As shown in Fig. 3, HyMU is mainly composed of two phases: *candidates generation* and *missing roads inferring*. During the first phase, we obtain the unmatched trajectories in each time window by map matching, distance denoising and direction denoising. Then, through clustering and centerline fitting on

Algorithm 1. Candidate Generation

Input: A trajectory set TwS in current time window
Output: Line-based candidate set RC_l and point-based candidate set RC_p

1 $RC_l \leftarrow \emptyset$; $RC_p \leftarrow \emptyset$; //line-based and point-based candidate set
2 $TuS \leftarrow \emptyset$; //unmatched trajectory segment set
3 **foreach** *trajectory* $Tw^{(i)}$ *in* TwS **do**
4 $Tu \leftarrow MapMatching(Tw^{(i)})$; //unmatched trajectory segments
5 $TuS \leftarrow TuS \cup Tu$;
6 **foreach** *trajectory segment* $Ts^{(x)}$ *in* TuS **do**
7 **if** $Ts^{(x)}$ *is noisy trajectory segment* **then**
8 $TuS \leftarrow TuS \setminus \{T_s^{(x)}\}$;
9 $CS_l \leftarrow LClustering(TuS)$; //line-based clustering
10 $CS_p \leftarrow PClustering(TuS)$; //point-based clustering
11 **foreach** *cluster* $CS_l^{(i)}$ *in* CS_l **do**
12 $RC_l \leftarrow RC_l \cup CLFitting(CS_l^{(i)})$; //line-based candidate generation
13 **foreach** *cluster* $CS_p^{(j)}$ *in* CS_p **do**
14 $RC_p \leftarrow RC_p \cup CLFitting(CS_p^{(j)})$; //point-based candidate generation
15 **return** RC_l and RC_p;

the unmatched and denoised trajectories, we derive the road candidates in each time window. During the second phase, we combine the candidates of multiple time windows via continuous observation. When the number of hybrid candidates related to a certain road reaches the threshold k, they will be merged to form a missing road. Finally, through road combination, we update the road network with inferred roads. Note that our hybrid framework integrates the advantages of line-based and point-based strategies, including high coverage and greater precision of inferred roads.

4.1 Candidate Generation

As shown in Algorithm 1, the candidate generation phase involves *map matching* (at lines 3–5), *denoising* (at lines 6–8), *clustering* (at lines 9–10) and *centerline fitting* (at lines 11–14). First, the trajectories in each time-window are matched with the road network to obtain unmatched trajectories. Then, after denoising, the denoised and unmatched trajectories are grouped into clusters using both line-based and point-based clustering methods. Finally, each cluster is fitted into a polyline that represents a road candidate through centerline fitting.

Map Matching. The purpose of map matching is to match the GPS trajectories to the right roads. Commonly used map matching can be divided into two categories: incremental approach [9,15], which aims to select the best matching candidate only on the basis of the preceding observations; global methods [14,17], which is to observe the entire series to select the best candidate. The Fast Viterbi [17], one of the most popular map matching methods, has been

Algorithm 2. LClustering

Input: Trajectory segment set TuS, a threshold th_c
Output: Cluster set CS

$CS \leftarrow \emptyset$; $l \leftarrow 1$;
foreach *unvisited segment* $Ts^{(i)}$ *in* TuS **do**
  Mark $Ts^{(i)}$ as visited;
  if $Nd(Ts^{(i)}) > th_c$ **then**
    $C_l \leftarrow \{Ts^{(i)}\}$; $Q \leftarrow \emptyset$; $Q.enqueue(Ts^{(i)})$;
    while Q *is not empty* **do**
      $Ts^{(x)} \leftarrow Q.dequeue()$;
      foreach *segment* $Ts^{(y)}$ *in* $Nd(Ts^{(x)})$ **do**
        Mark $Ts^{(y)}$ as visited;
        if $Nd(Ts^{(y)}) > th_c$ ***and*** $Ts^{(x)}$ *and* $Ts^{(y)}$ *are similar* **then**
          $Q.enqueue(Ts^{(y)})$;
        if $Ts^{(y)}$ *does not belong to any cluster* **then**
          $C_l \leftarrow C_l \cup \{Ts^{(y)}\}$;
    $CS \leftarrow CS \cup \{C_l\}$; $l \leftarrow l + 1$;
return CS;

adopted by most of map updating methods (e.g. CrowdAtlas and Glue) due to its excellent performance. Likely, the $MapMatching$ function in Algorithm 1 (at line 4) also implements Viterbi, and derives unmatched trajectory segments by selecting candidates with the maximal weight after calculating the candidate positions within a certain radius. Finally, we will obtain a set of unmatched trajectory segments.

Denoising. GPS samples often have a few noisy data of position or direction. To improve the accuracy of inferred missing roads, denoising process is required to reduce the noisy samples. For example, there are a few noisy points (in red) in Fig. 2(b). First, the red circled points can be removed through distance denoising because they are far from most of its surrounding points. Subsequently, as the red track points in Fig. 2(b) are significantly different from most of its surrounding points in directions, they are removed by direction denoising [6]. Specifically, we search the nearby segments of each trajectory segment, and compare the direction of it with its neighboring segments. Then, we identify a noisy trajectory segment according to the significant gap between its direction and most of its surrounding segments' direction. The denoising result after distance and direction denoising is shown in Fig. 2(c).

Clustering. After map matching and denoising, the unmatched trajectory segments need to be clustered to infer the road candidates. To enhance the accuracy of inferred missing roads, we combine both line-based clustering ($LClustering$) and point-based clustering ($PClustering$). The point-based clustering takes two endpoints of all trajectory segments as input, while the line-based cluster takes

trajectory segments as input. In Algorithm 2 (*LClustering*), each trajectory segment is initialized as a cluster once the number of its similar trajectory segments is greater than a specific threshold, i.e., $N_d(p_i) > th_c$ (at lines 2–5). The similar trajectory segment is defined below.

Definition 6 (Similar Trajectory Segment). *Given two trajectory segments* $T_s^{(x)}$ *and* $T_s^{(y)}$*, a distance threshold* th_{dis} *and a direction threshold* th_{dir}*,* $T_s^{(x)}$ *and* $T_s^{(y)}$ *are two similar trajectory segments if the distance between* $Ts^{(x)}$ *and* $Ts^{(y)}$ *is smaller than* th_{dis}*, and the angle between* $Ts^{(x)}$ *and* $Ts^{(y)}$ *is less than* th_{dir}*.*

Then, for each segment $Ts^{(x)}$ in one cluster and each segment $Ts^{(y)}$ in $Nd(Ts^{(x)})$, if they are similar and $Nd(Ts^{(y)}) > th_c$, we add $Ts^{(y)}$ into queue. If $Ts^{(y)}$ does not belong to any cluster, it should also be added into the cluster of $Ts^{(x)}$ (at lines 7–13). The *PClustering* approach also divides the input points into several clusters according to the similar criterion. The directions of two endpoints of a segment can be seen as the direction of the segment. Due to space limitations, we omit the detail of *PClustering*.

Centerline Fitting. The centerline fitting step aims to generate the centerlines to represent road candidates. Since a cluster that consists of the trajectory points or segments may belong to the same road candidate, we need to fit a centerline to represent a road candidate. For the clustering results of former stage, we use the sweeping line method in [7] to realize the centerline fitting process. The *CLFitting* function in Algorithm 1 takes trajectory points or segments as input, and generates the road candidates (at lines 11–14). Finally, we obtain line-based candidates and point-based candidates.

4.2 Missing Roads Inferring

In this phase, we group road candidates belonging to the same road based on two kinds of road candidates, RC_l and RC_p, generated in Algorithm 1. To be specific, if k road candidates (at least one line-based candidate and one line-based candidate) are located on the same road, they are merged to infer a missing road. After that, we connect the inferred roads with existing roads in network. Therefore, the missing roads inferring phase is composed of two steps, including *continuous observation* and *road combination*.

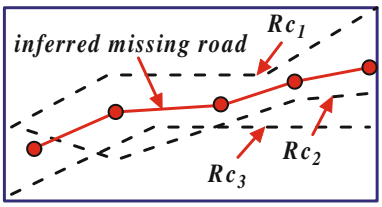


Fig. 4. An example of missing road generated in the MBR

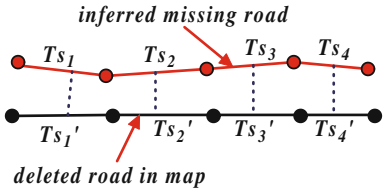


Fig. 5. An example of two similar road centerlines

Algorithm 3. Continuous Observation

Input: Road candidate sets RC_p and RC_l, a threshold k ($k \geqslant 3$)
Output: A missing road set $RS = \{R_1, R_2, ..., R_m\}$

```
1 RS ← ∅; i ← 1;
2 foreach unvisited candidate Rc^(i) in RC_p ∪ RC_l do
3     Z ← Rc^(i) ∪ {Rc^(j) | Rc^(j) ∈ RC_p ∪ RC_l, Rc^(i) and Rc^(j) are similar} ;
4     Mark all road candidates in Z as visited;
5     if |Z| ⩾ k and at least one point-based candidate in Z then
6         R_i ← CLFitting(Z) ;
7         RS ← RS ∪ {R_i}; i ← i + 1;
8 return RS;
```

Continuous Observation. As mentioned in Sect. 1, since the sparse trajectories in one time window may be confused with noise, the road candidates derived from them may imply wrong missing roads. To improve the precision, we propose a continuous observation approach to infer the missing roads based on the candidates of multiple time windows. To be specific, as show in Algorithm 3, we collect the candidates of consecutive time windows so far and take them as input. First, we divide all road candidates according to the roads which they belong to (at line 3). When the number of road candidates exceeds a predefined threshold k ($k \geqslant 3$) and at least one point-based candidate is involved, we can identify a missing road. Next, we fit them into a missing road by invoking $CLFitting$ function (at line 6).

For example, there are three similar road candidates (e.g. the black polyline) co-exist in Fig. 4. As the number of road candidates reaches the predefined threshold (k = 3), we combine them to generate a new road centerline to represent a missing road. To be specific, given two road candidates $Rc^{(x)}$ and $Rc^{(y)}$, if $\exists Ts^{(i)} \in Rc^{(x)}, Ts^{(j)} \in Rc^{(y)}$, and $Ts^{(i)}$ and $Ts^{(j)}$ are similar, we take $Rc^{(y)}$ as one of the similar road candidates of $Rc^{(x)}$ and take them as candidates of the same road. In Fig. 4, if the number of road candidates reaches k ($k = 3$), a missing road will be inferred through centerline fitting. If $k = 4$, we continue to observe the road candidates in the following time windows until the number of road candidates belonging to the same road reaches 4.

Road Combination. After inferring missing roads, we update the existing road network by connecting the inferred roads to the existing neighboring roads. Given an inferred missing road $Rc^{(x)}$, we try to find a road $Rc^{(y)}$ in the road network such that the $Rc^{(y)}$ is close to one of the endpoints of $Rc^{(x)}$ (e.g. smaller than 20 m). If such $Rc^{(y)}$ exists, we update the existing road network by connecting $Rc^{(x)}$ and $Rc^{(y)}$.

5 Experimental Evaluation

We conduct substantial comparison experiments on real data sets to evaluate the performance of HyMU. Specifically, we compare HyMU with line-based method

(CrowdAtlas [16]) and point-based method (Glue [18]) to verify the superiority of HyMU. Our codes, written in Java, are conducted on a PC with 16 GB RAM, Intel Core CPU 3.2 GHz i7 processor, and the operating system is Windows 10.

5.1 Evaluation Method

In order to ensure fairness, we randomly select an area on the existing map and remove some road segments from this region. The goal is to verify whether the deleted road segments can be inferred by different map updating methods. Evaluation criteria includes *Precision*, *Recall* and *F-measure* [8,18]. Let *truth* denote the deleted roads, $inferred$ denote all inferred road segments, and *tp* denote the correctly inferred roads. Accordingly, we use $len(truth)$, $len(inferred)$ and $len(tp)$ to represent the length of all the deleted roads, the inferred roads and the correctly inferred roads respectively. Then, *Precision*, *Recall* and *F-measure* can be calculated as follows.

$$Precision = \frac{len(tp)}{len(inferred)} \qquad Recall = \frac{len(tp)}{len(truth)}$$

$$F\text{-}measure = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

As shown in Fig. 5, the deleted roads and their corresponding inferred missing roads are split into small segments with fixed length. Then, *tp* can be denoted as below.

$$tp = \{si(Ts^{(x)}, Ts^{(y)}) | \forall Ts^{(x)} \in inferred,\ \forall Ts^{(y)} \in truth\}$$

The function $si(Ts^{(x)}, Ts^{(y)})$ returns $Ts^{(x)}$ if $Ts^{(x)}$ and $Ts^{(y)}$ are similar. Otherwise, it returns *null*.

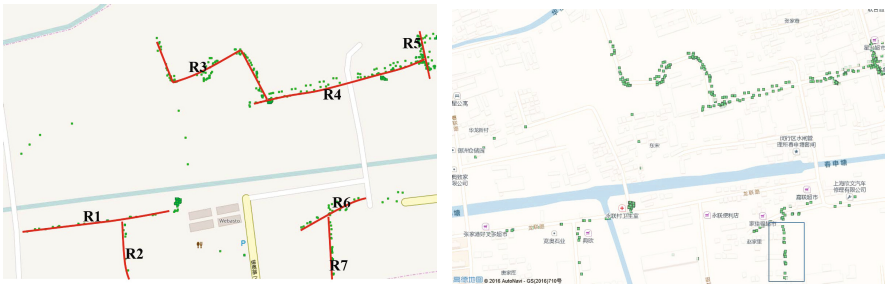
5.2 Data Sets and Map

We use two real data sets to evaluate the effectiveness of HyMU method, including a taxi trajectory data set of 2015 in *ShanghaiOpen Data Apps*[2] (hereafter termed $Taxi2015$) and a high-sampling Shanghai taxi data set in 2013 (hereafter termed $Shanghai2013$). In addition, we choose an open source map $OpenStreetMap(OSM)$[3] as our map data.

$Taxi2015$ contains the GPS logs of taxis from Apr. 1 to Apr. 30, 2015. It involves about 10,000 trajectories every day (about 115 million points). Each GPS log, represented by a sequence of time-stamped points, contains Vehicle ID, Time, Longitude and Latitude, Speed, etc.

$Shanghai2013$ contains the GPS logs of taxis in 2 days (from Oct. 1 to Oct. 2). It involves about 50,000 trajectories every day (about 107 million points). The average sampling rate of the objects is about 60 s. Besides, each GPS log, represented by a sequence of time-stamped points, contains Vehicle ID, Time, Longitude and Latitude, Speed, etc.

[2] http://soda.datashanghai.gov.cn/.
[3] http://wiki.openstreetmap.org/.



(a) OSM Map (b) AutoNaviMap

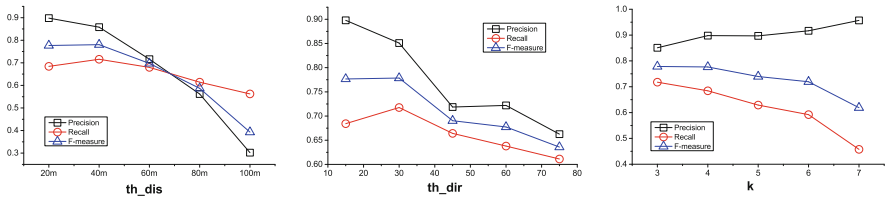
Fig. 6. Visualization result of HyMU on *Taxi*2015

5.3 Effectiveness Evaluation

Results for Taxi2015. We first implement HyMU on *Taxi*2015 to infer about 150 road segments that haven't been described in OSM map. The visualization result is shown in Fig. 6(a), where the red lines represent the missing roads detected by HyMU. As compared to the roads in *AutoNaviMap*[4] (as shown in Fig. 6(b)), we can find that six roads (R_1–R_6) are correctly inferred by HyMU. This verifies the high precision of our proposal. In addition, we infer the road R_7 that is not marked on *AutoNaiveMap*, which further confirms the superiority of HyMU in discovering missing roads on sparse trajectory data.

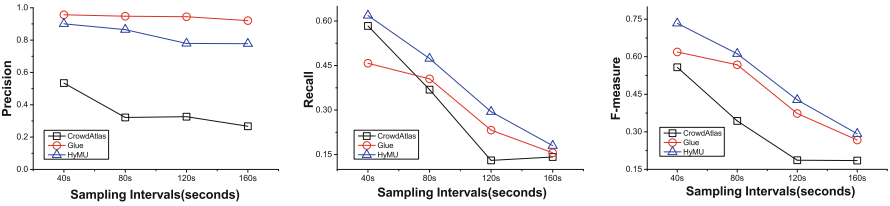
Results for Shanghai2013. We compare HyMU with CrowdAtlas and Glue on *Shanghai*2013 and randomly select a test area consists of 19 road segments (from North Zhang Yang Road, through Wuzhou Avenue and Shenjiang Road, to Jufeng Road). Firstly, to verify the robustness of HyMU, we evaluate sensitivity of parameters (th_{dis}, th_{dir} and k) on *Shanghai*2013, as illustrated in Fig. 7. After tuning them repeatedly, we find that HyMU achieves the best performance on *Shanghai*2013 when $th_{dis} = 20\,m$, $th_{dir} = \frac{\pi}{6}$ and $k = 3$. Secondly, we further evaluate HyMU, CrowdAtlas and Glue by varying the sampling interval from 40 s to 160 s. As shown in Fig. 8, we find that Glue has the best precision because the point-based strategy will not infer the missing roads with wrong direction. But it does not take into account inferring roads on sparse region, which result in a lower recall rate. By contrast, HyMU combines the advantage of line-based and point-based strategies. It attains almost the same precision as Glue, and the highest recall as well as F-measure. Thirdly, we evaluate the performance of HyMU, CrowdAtlas and Glue under various data volume, as shown in Fig. 9. As data volume becomes larger, we observe that the precision, recall and F-measure value of HyMU increases accordingly, and the precision approaches Glue. Hence, HyMU has a good scalability. Additionally, we observe that HyMU has higher recall than other methods in all situations, which demonstrates that it is capable of inferring missing roads on sparse trajectory data.

[4] http://ditu.amap.com/.



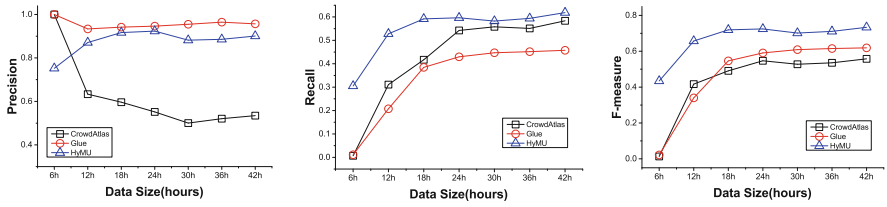
(a) Varying th_{dis} (b) Varying th_{dir} (c) Varying k

Fig. 7. Performance of HyMU under different parameters on *Shanghai*2013



(a) Precision (b) Recall (c) F-measure

Fig. 8. Performance comparison under various sampling intervals on *Shanghai*2013

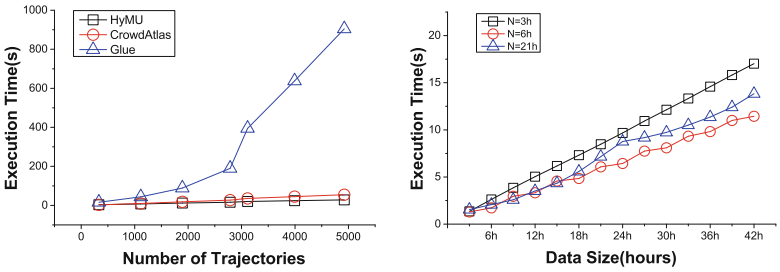


(a) Precision (b) Recall (c) F-measure

Fig. 9. Performance comparison under various data volume on *Shanghai*2013

5.4 Efficiency Evaluation

Next, we assess the efficiency of HyMU by comparison with CrowdAtlas and Glue on *Shanghai*2013. As shown in Fig. 10(a), HyMU run faster than the other two methods with the increase of trajectory data. It indicates that HyMU is more efficient than other map updating methods. GLUE, by contrast, is extremely time-costing, due to the cost on calculating direction of each point. Additionally, we evaluate the efficiency of HyMU by varying the time window size N. Figure 10(b) shows the processing time comparison when N is set to 3 h, 6 h and 21 h respectively. When $N = 6$ h, the execution time is the smallest. This is due to that massive amount of data in a time window requires to be denoised and clustered which is quite time-consuming if the time window size is large. Conversely, when time window size is small, we need to deal with too many road candidates, which is also time-consuming. So the appropriate window size is 6 h



(a) Efficiency comparison (b) Efficiency of HyMU

Fig. 10. Efficiency evaluation

on *Shanghai*2013, and we also use this optimal value to execute effectiveness evaluation on *Shanghai*2013. Consequently, HyMU is efficient and effective to infer the missing roads for a given map.

6 Conclusion

In this paper, we address the issue of inferring missing roads on sparse trajectory data of vehicles. On the basis of sliding window model, we propose a hybrid framework called HyMU to infer the missing roads. HyMU is mainly composed of two phases: road candidates generation and missing roads inferring. Owing to advantages of the hybrid framework, HyMU attains a better performance as compared to the other map updating methods. Substantial experimental results demonstrate the superiority of HyMU especially in dealing with sparse trajectory data. In addition, since there are other forms of road changes in the road network (e.g. blocked roads). Such road changing information is very important in navigation applications. In the future work, we proceed to study how to detect road changes, and provide real-time traffic information to users to enable route planning.

Acknowledgement. Our research is supported by the National Key Research and Development Program of China (2016YFB1000905), NSFC (61370101, 61532021, U1501252, U1401256 and 61402180), Shanghai Knowledge Service Platform Project (No. ZF1213).

References

1. Agamennoni, G., Nieto, J.I., Nebot, E.M.: Robust inference of principal road paths for intelligent transportation systems. IEEE Trans. Intell. Transp. Syst. **12**(1), 298–308 (2011)
2. Biagioni, J., Eriksson, J.: Map inference in the face of noise and disparity. In: SIGSPATIAL, pp. 79–88 (2012)
3. Cao, L., Krumm, J.: From GPS traces to a routable road map. In: GIS, pp. 3–12 (2009)

4. Davies, J.J., Beresford, A.R., Hopper, A.: Scalable, distributed, real-time map generation. IEEE Pervasive Comput. **5**(4), 47–54 (2006)
5. Edelkamp, S., Schrödl, S.: Route planning and map inference with global positioning traces. In: Computer Science in Perspective, Essays Dedicated to Thomas Ottmann, pp. 128–151 (2003)
6. Ge, Y., Xiong, H., Zhou, Z., Ozdemir, H.T., Yu, J., Lee, K.C.: Top-eye: top-k evolving trajectory outlier detection. In: CIKM, pp. 1733–1736 (2010)
7. Lee, J., Han, J., Whang, K.: Trajectory clustering: a partition-and-group framework. In: SIGMOD, pp. 593–604 (2007)
8. Liu, X., Biagioni, J., Eriksson, J., Wang, Y., Forman, G., Zhu, Y.: Mining large-scale, sparse GPS traces for map inference: comparison of approaches. In: KDD, pp. 669–677 (2012)
9. Mazhelis, O.: Using recursive bayesian estimation for matching GPS measurements to imperfect road network data. In: International IEEE Conference on Intelligent Transportation Systems, pp. 1492–1497 (2010)
10. Mokhtarzade, M., Zoej, M.J.V.: Road detection from high-resolution satellite images using artificial neural networks. Int. J. Appl. Earth Obs. Geoinf. **9**(1), 32–40 (2007)
11. Schrödl, S., Wagstaff, K., Rogers, S., Langley, P., Wilson, C.: Mining GPS traces for map refinement. Data Min. Knowl. Discov. **9**(1), 59–87 (2004)
12. Seo, Y., Urmson, C., Wettergreen, D.: Exploiting publicly available cartographic resources for aerial image analysis. In: SIGSPATIAL, pp. 109–118 (2012)
13. Shan, Z., Wu, H., Sun, W., Zheng, B.: COBWEB: a robust map update system using GPS trajectories. In: UbiComp, pp. 927–937 (2015)
14. Thiagarajan, A., Ravindranath, L., Balakrishnan, H., Madden, S., Girod, L.: Accurate, low-energy trajectory mapping for mobile devices. In: NSDI (2011)
15. Velaga, N.R., Quddus, M.A., Bristow, A.L.: Developing an enhanced weight-based topological map-matching algorithm for intelligent transport systems. Transp. Res. Part C Emerg. Technol. **17**(6), 672–683 (2009)
16. Wang, Y., Liu, X., Wei, H., Forman, G., Chen, C., Zhu, Y.: CrowdAtlas: self-updating maps for cloud and personal use. In: MobiSys, pp. 27–40 (2013)
17. Wei, H., Wang, Y., Forman, G., Zhu, Y., Guan, H.: Fast viterbi map matching with tunable weight functions. In: SIGSPATIAL, pp. 613–616 (2012)
18. Wu, H., Tu, C., Sun, W., Zheng, B., Su, H., Wang, W.: GLUE: a parameter-tuning-free map updating system. In: CIKM, pp. 683–692 (2015)