HIMM: An HMM-Based Interactive Map-Matching System

Xibo Zhou[1(✉)], Ye Ding[2], Haoyu Tan[2], Qiong Luo[1], and Lionel M. Ni[3]

[1] Department of Computer Science and Engineering,
The Hong Kong University of Science and Technology,
Kowloon, Hong Kong
{xzhouaa,luo}@ust.hk
[2] Guangzhou HKUST Fok Ying Tung Research Institute,
The Hong Kong University of Science and Technology,
Kowloon, Hong Kong
{yeding,haoyutan}@ust.hk
[3] University of Macau, Zhuhai, China
ni@umac.mo

Abstract. Due to the inaccuracy of GPS devices, the location error of raw GPS points can be up to several hundred meters. Many applications using GPS-based vehicle location data require map-matching to pre-process GPS points by aligning them to a road network. However, existing map-matching algorithms can be limited in accuracy due to various factors including low sampling rates, abnormal GPS points, and dense road networks. In this paper, we propose the design and implementation of HIMM, an **H**MM-based **I**nteractive **M**ap-**M**atching system that produces accurate map-matching results through human interaction. The main idea is to involve human annotations in the matching process of some elaborately selected error-prone points and to let the system automatically adjust the matching of the remaining points. We use both real-world and synthetic datasets to evaluate the system. The results show that HIMM can significantly reduce human annotation costs comparing to the baseline methods.

Keywords: Map-matching · Interactive system · Trajectory

1 Introduction

With the ubiquity of location sensing technologies in a wide range of location-based devices such as vehicle GPS navigators and mobile phones, large amounts of trajectory data have been collected from different sources. These data have been utilized by various location-based services such as route recommendation, traffic control, and location-based social networks. A trajectory consists of a sequence of location points with latitudes, longitudes, and time-stamps. In practice, the location information of a trajectory are imprecise due to measurement noises and sampling errors [5]. It is therefore necessary to perform *map-matching* [5] by aligning the observed location points to the road networks

S. Candan et al. (Eds.): DASFAA 2017, Part II, LNCS 10178, pp. 3–18, 2017.
DOI: 10.1007/978-3-319-55699-4_1

in a digital map so that these position data can be sufficiently accurate for trajectory-based applications.

The fundamental difficulty of map-matching is that raw trajectory data typically do not consist the actual paths of moving objects, especially when the location information is collected passively. Without the ground truth, it is difficult to train or evaluate any map-matching algorithms. Hence, in order to collect the ground truth, it is necessary to involve human contributions such as driving a vehicle along the road network and collecting the raw trajectory data along with the corresponding path manually. However, the cost of such methods is quite high. Hence, in this paper, we propose an interactive system called HIMM to process the raw trajectories to reduce the cost of generating the ground truth.

The main process of the system is to interactively select raw trajectory points for human annotators to match, and the challenge is in how to facilitate the interactive map-matching process. A naïve approach would be to simply throw all the sample points on a trajectory onto the road network, and then ask the annotator to drag each point to the correct road segment. However, it is tedious and at times challenging for the annotator to find a proper candidate road segment for each point of the trajectory. A more effective and user-friendly approach would be the following: (1) generate an initial path using certain map-matching algorithms, and display the path on the digital map along with the original trajectory; (2) ask the annotator to drag each mismatched point to the correct road segment. Furthermore, given the feedbacks of an annotator, the interactive system could keep updating the path in display after each human annotation. Unfortunately, to the best of our knowledge, none of the existing map-matching algorithms is able to utilize the feedbacks of annotators.

In this paper, we propose a novel interactive map-matching algorithm that takes the feedbacks of annotators to improve the matching result. Although such an interactive system can help an annotator to easily adjust a single point, the total annotation cost of a trajectory may still be high, because in order to pick and confirm the exact points that are mismatched, the annotator may check a large portion of the points, which could be up to the entire trajectory. As a result, it is desirable that the interactive map-matching system provides some guidance recommending potentially mismatched points for the annotator to check. Such a strategy of posing queries to the annotator can reduce the annotation cost, which is a key research issue in active learning and crowd-sourcing [12]. However, due to the complexity of map-matching algorithms as well as the input trajectories and the road network, existing query selection strategies are not suitable for the interactive map-matching task. Therefore, we design efficient strategies to pose queries for the interactive map-matching algorithm.

The contributions of this paper lie in the following aspects: (1) we propose a novel system framework for interactive map-matching. It is a general framework that combines human efforts with algorithms in an iterative manner to achieve high map-matching accuracy; (2) we design a new HMM (Hidden Markov Model)-based map-matching algorithm that can take an arbitrary number of human annotations into consideration. To the best of our knowledge, this is the first map-matching algorithm whose accuracy can be largely enhanced by the input of human knowledge; (3) we propose different query selection strategies to

effectively reduce the number of points that are required to be manually annotated. Compared with traditional approaches, our query selection strategies can reduce the number of queries by up to 44%; and (4) we use both real world and synthetic trajectory datasets to perform experiments and analyze the empirical results. The results demonstrate that HIMM can significantly reduce human annotation cost.

In the remainder of this paper, we first discuss related work in Sect. 2, and then introduce the problem definitions and the framework of our system in Sect. 3. The map-matching algorithms and the query selection strategies are described in Sects. 4 and 5, respectively. We evaluate our system in Sect. 6, and conclude the paper in Sect. 7.

2 Related Work

Map-Matching. Existing map-matching algorithms can be categorized into three types: geometric algorithms, topological algorithms, and statistical algorithms. Geometric algorithms [6] utilize spatial information to find local matches for each point of the trajectory, thus the accuracy is highly affected by the measurement noises. Topological algorithms [2] consider both the connectivity and contiguity of the road network as well as the geometric information, but the accuracy is reduced when the sampling rate of the trajectory is low. Statistical algorithms make use of advanced statistical models such as Kalman filter [10], particle filter [7], and HMM [8,9,14], to find the global optimal path for the trajectory. These algorithms are less sensitive to measurement noise and sampling rate, but the time complexity is high. To the best of our knowledge, none of the existing map-matching algorithms takes feedbacks from human annotators to improve accuracy or provides interactive mechanisms to facilitate map-matching. More details are shown in Sect. 4.2.

Active Learning. Many learning tasks face a situation where unlabeled data are easy to obtain but annotation is costly [12]. Active learning aims to minimize the annotation cost by querying the most informative instances in the unlabeled dataset. Although the scope of active learning is broad, most of the methods are not suitable to the map-matching problem. For example, graph-based active learning [1] focuses on using graph-based metrics to define the informativeness of instances and querying the most informative instances. These approaches utilize link information with node-specific features or partial network structures to improve the classification accuracy. Different from graph-based active learning, the points of a trajectory in this paper are not part of the graph, but have certain mapping relations with the edges of the graph. Another example is the active learning algorithms for structured prediction tasks [13], which ignore the annotation cost of a single structured object, but query the instances with the highest joint uncertainty or utility. However, the map-matching task for a single trajectory is costly, which cannot be disregarded. To the best of our knowledge, none of the existing active learning frameworks is designed for interactive map-matching.

3 Overview

3.1 Preliminary

Definition 1 (Road Segment). *A road segment e is a directed polyline between two road intersections v_i and v_j, and there is no other road intersection on e. We denote $v_i \in e$ and $v_j \in e$.*

Definition 2 (Road Network). *A road network is a weighted directed graph $G = (V, E)$, where V is a set of road intersections (or vertices), and E is a set of road segments (or edges). The weight of a road segment is represented by its properties.*

A moving object is only allowed to travel on the road segments within the road network.

Definition 3 (Trajectory). *A trajectory T is a sequence of location points sampled from the GPS device of a moving object, denoted as $T = (p_1, p_2, \cdots, p_n)$. We say $p_i \in T$ for $i = 1, \cdots, n$ and $|T| = n$.*

A location point is represented by its latitude and longitude. The sampled location points on a trajectory may not be the actual locations of the moving object due to measurement inaccuracy.

Definition 4 (Path). *A path $P = (e_1, e_2, \cdots, e_n)$ is a sequence of road segments where e_i and e_{i+1} are connected for $i = 1, 2, \cdots, n-1$. Two road segments e_i and e_j are connected if there exists some intersection v such that $v \in e_i$ and $v \in e_j$.*

Definition 5 (Match). *Given a trajectory T and a road network $G = (V, E)$, a match $m_{i,j} = \langle p_i, e_j \rangle$ where $p_i \in T$ and $e_j \in E$ specifies point p_i was sampled when the object was moving on road segment e_j.*

Definition 6 (Map-Matching Query). *Given a trajectory T and a road network $G = (V, E)$, a map-matching query $Q(T, G)$ finds a path P, such that each point $p_i \in T$ is matched to exactly one road segment $e_j \in E$. The resulting set of matches is denoted as $M = \{\langle p_1, e_{j_1} \rangle, \cdots, \langle p_n, e_{j_n} \rangle\}$.*

Definition 7 (Interactive Map-Matching Query). *Given a trajectory T, a road network $G = (V, E)$, and a set of matches M' conducted by the annotator, an interactive map-matching query $Q(T, G, M')$ finds a new path P, such that each point $p_i \in T$ is matched to exactly one road segment $e_j \in E$. The resulting set of matches is denoted as M, where $M' \subseteq M$.*

3.2 Framework

Figure 1 shows the workflow of *HIMM*, our interactive map-matching system. First, an annotator requests a trajectory T to perform the map-matching task.

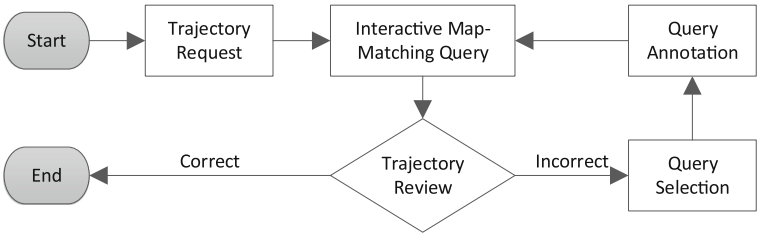


Fig. 1. The workflow of HIMM.

If T is not map-matched before, HIMM automatically generates a path for T using our interactive map-matching algorithm with $M' = \emptyset$. Then, HIMM plots the trajectory T along with the path onto the digital map for the annotator to review. In each iteration, if the annotator considers that the trajectory is not correctly map-matched, an unlabeled point p is selected for the annotator to review using our query selection strategy. If the annotator considers that p is not correctly matched, the annotator marks a correct match for p; otherwise, the annotator leaves the match as is. During the task, HIMM maintains a set of matches M' that are specified by the annotator. After receiving the feedbacks from the annotator, HIMM adds the match of p to M', and then performs an interactive map-matching query with M' to complete the iteration. Finally, if the annotator considers that all points are correctly map-matched, the map-matching task for the trajectory T terminates.

HIMM contains two major components shown in Fig. 1: (1) an interactive map-matching algorithm that takes the feedbacks of the annotator and automatically adjusts the map-matching results; and (2) a query selection strategy that recommends potentially mismatched points for the annotator to review. The details are introduced in Sects. 4 and 5, respectively.

4 Interactive Map-Matching

4.1 Map-Matching Model

As shown in Fig. 2, we model a map-matching query as a hidden Markov model, which is one of the most suitable models in this area [9,15]. Given a map-matching query $Q(T, G)$, trajectory T represents an observation sequence, where each point $p_i \in T$ is an observation, and each candidate road segment $e_{i,j} \in E$ represents a hidden state of p_i. The total number of points of a trajectory is denoted by n, where $n = |T|$, and the total number of hidden states of each point p_i is denoted by r, where $r = |E|$.

For each point p_i, each state $e_{i,j}$ has an *emission probability* denoted as $\Pr(e_{i,j}|p_i)$, which represents the likelihood of p_i being observed if the vehicle is on road segment $e_{i,j}$. A higher emission probability is associated to p_i if $e_{i,j}$ is closer to p_i, and the emission probability follows a Gaussian distribution of positioning measurement noise [9]:

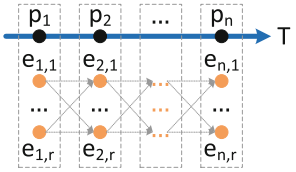


Fig. 2. The map-matching model of HIMM.

$$\Pr(e_{i,j}|p_i) = \frac{1}{\sqrt{2\pi}\delta} \mathrm{e}^{-\frac{1}{2}\left(\frac{\mathrm{pdist}(e_{i,j},p_i)}{\delta}\right)^2} \tag{1}$$

where δ is the standard deviation of the positioning measurement noise, and $\mathrm{pdist}(e_{i,j}, p_i)$ is the *minimum perpendicular distance* [3] between $e_{i,j}$ and p_i.

For each pair of consecutive points (p_i, p_{i+1}), each pair of candidate states $(e_{i,j_i}, e_{i+1,j_{i+1}})$ associated with them has a *transition probability* denoted as $\Pr(e_{i,j_i}, e_{i+1,j_{i+1}}|p_i, p_{i+1})$, which represents the likelihood for a vehicle moving from $e_{i,j}$ to $e_{i+1,j_{i+1}}$. e_{i,j_i} and $e_{i+1,j_{i+1}}$ are more likely to be matched to p_i and p_{i+1}, respectively, if the driving distance along e_{i,j_i} and $e_{i+1,j_{i+1}}$ from p_i to p_{i+1} is closer to the great circle distance between p_i and p_{i+1}; and the transition probability follows an exponential distribution:

$$\Pr(e_{i,j_i}, e_{i+1,j_{i+1}}|p_i, p_{i+1}) = \frac{1}{\beta} \mathrm{e}^{-\frac{|\mathrm{cdist}(p_i,p_{i+1}) - \mathrm{route}(p_i,p_{i+1})|}{\beta}} \tag{2}$$

where β is the rate parameter [9], $\mathrm{cdist}(p_i, p_{i+1})$ is the great circle distance between p_i and p_{i+1}, and $\mathrm{route}(p_i, p_{i+1})$ is the driving distance along e_{i,j_i} and $e_{i+1,j_{i+1}}$ from p_i to p_{i+1}.

4.2 Interactive Map-Matching Algorithm

Recall that a map-matching query $Q(T, G)$ finds a path P, such that each point $p_i \in T$ is matched to exactly one road segment $e_j \in E$. Hence, the objective of the map-matching algorithm is to find a sequence of hidden states $P = (e_{1,j_1}, e_{2,j_2}, \cdots, e_{n,j_n})$ with the maximum joint probability $\Pr(P)$, where:

$$\Pr(P) = \prod_{i=1}^{n} \Pr(e_{i,j_i}|p_i) \times \prod_{i=1}^{n-1} \Pr(e_{i,j_i}, e_{i+1,j_{i+1}}|p_i, p_{i+1}) \tag{3}$$

Traditional hidden Markov model uses the Viterbi algorithm [11] to find the optimal solution, denoted as P^*. The Viterbi algorithm uses dynamic programming to quickly find the state sequence that maximizes $\Pr(P^*)$ in a recursive manner. Hence, if the annotator specifies a match $\langle p_i, e_{i,k} \rangle$ where $e_{i,k} \notin P^*$, the Viterbi algorithm will ignore such feedback of the annotator. Therefore, the traditional Viterbi algorithm cannot utilize the feedbacks of an annotator.

We propose an interactive map-matching algorithm based on the Viterbi algorithm to utilize the feedbacks of the annotator. The recursive formulation (a.k.a., *forward formulation* [11]) is defined as:

$$C(i,j) = \begin{cases} O(i,j) & \langle p_i, e_{i,k} \rangle \in M', k = j \\ 0 & \exists \langle p_i, e_{i,k} \rangle \in M', k \neq j \\ \Pr(e_{i,j}|p_i) \times O(i,j) & \nexists \langle p_i, e_{i,k} \rangle \in M' \end{cases} \tag{4}$$

where $1 \leq k \leq r$, and:

$$O(i,j) = \begin{cases} 1 & i = 1 \\ \max_{1 \leq k \leq r} C(i-1,k) \Pr(e_{i-1,k}, e_{i,j}|p_{i-1}, p_i) & i > 1 \end{cases} \tag{5}$$

In Formula (4), $C(i,j)$ represents the highest value of the probabilities of state sequences $P_i = (e_{1,j_1}, e_{2,j_2}, \cdots, e_{i,j_i})$ for the first i observations $T_i = (p_1, p_2, \cdots, p_i)$ that have $e_{i,j}$ as the final state.

The recursion terminates when the last observation is processed. The optimal state sequence that results in $C(i,j)$ can be retrieved reversely from the last hidden state that results in the maximum $C(i,j)$ in each step through the following formulation (a.k.a. *backward formulation* [11]):

$$e_{i,j_i} = \begin{cases} \arg\max\limits_{1 \leq j \leq r} C(i,j) & i = n \\ \arg\max\limits_{1 \leq j \leq r} \dfrac{C(i+1,j)}{\Pr(e_{i,j}, e_{i+1,j_{i+1}}|p_i, p_{i+1})} & 1 \leq i < n \end{cases} \tag{6}$$

Similar to the Viterbi algorithm, our interactive map-matching algorithm uses dynamic programming [11] to quickly find the optimal state sequence in a recursive manner. When the algorithm calculates the local optimal probability $C(i,j)$ for each hidden state $e_{i,j}$ as the final state for the first i observations $T_i = (p_1, p_2, \cdots, p_i)$, it first checks whether p_i is manually matched by the annotator. If so, the algorithm prunes all candidate hidden states of p_i except $e_{i,j}$, which is chosen by the annotator (i.e., $\langle p_i, e_{i,j} \rangle \in M'$) by modifying the emission probability $\Pr(e_{i,j}|p_i)$ to 1, and all other emission probabilities $\Pr(e_{i,k}|p_i)$ to 0, where $1 \leq k \leq r$ and $k \neq j$. Otherwise, the emission probability $\Pr(e_{i,j}|p_i)$ is set via Formula (1). This way, all $C(i,k)$ where $1 \leq k \leq r$ and $k \neq j$ are 0, and the backward formulation is guaranteed to select the state sequence that contains $e_{i,j}$ with respect to $C(i,j)$.

5 Query Selection Strategy

A good query selection strategy is critical to effectively guide the annotator by picking the points that are likely to be mis-matched. In this section, we propose four query selection strategies for comparison.

5.1 Distance-Based Strategy

A commonly used query selection strategy in active learning is uncertainty sampling. The basic idea is to query the instance whose label is the least certain. In the map-matching problem, the most straight-forward factor that reflects the uncertainty of a trajectory point p_i is the distance distribution from p_i to its candidate road segment set E_i. Consider the example shown in Fig. 3(a), p_1 and p_3 are clearly closer to e_1 and e_7, respectively. Thus, there is no need to check p_1 or p_3 since their labels are almost certain. However, the distances between p_2 and $e_2/e_3/e_6$ are quite similar, which makes p_2 the most uncertain point to be labeled.

A general strategy of uncertainty measurement in information theory is Shannon entropy [12], which represents the average amount of information generated by a probability distribution. As described in Sect. 4.1, the emission probability distribution of the candidate state set E_i of p_i reflects the distance distribution from p_i to each $e_{ij} \in E_i$. Thus, a distance-based strategy defines the uncertainty $H(p_i)$ of p_i as the Shannon entropy of the emission probability distribution of the candidate state set E_i of p_i. More specifically,

$$H(p_i) = -\sum_{j=1}^{r} \Pr(e_{ij}|p_i) \log(\Pr(e_{ij}|p_i)) \tag{7}$$

where r is the number of candidate states of p_i.

Based on Formula (7), given a trajectory T, in each iteration of the interactive map-matching process, the next point recommended for the annotator is:

$$p' = \arg\max_{p_i \in T} H(p_i) \tag{8}$$

After the annotator checks p_i, the system modifies $H(p_i)$ to -∞, so that p_i will not be checked again until the interactive map-matching process terminates. Hence, the time complexity of the distance-based strategy is $O(1)$ with respect to the number of points on the trajectory n.

5.2 Confidence-Based Strategy

In the distance-based strategy, the emission probability distribution of a trajectory point p_i only considers the local information of each point on the trajectory. Consider the example shown in Fig. 3(b). The distances between p_2 and e_2/e_4 are similar, whereas p_3 is closer to e_5 than e_3. According to the distance-based strategy, p_2 has a higher uncertainty than p_3. However, if we consider the entire trajectory, p_3 is the point that mostly likely to be checked, because the optimal path differs a lot if p_3 is matched to e_3 or e_5. On the other hand, p_2 is not likely to be matched to e_4 considering the topological information $((e_1, e_4, e_8, e_5, e_3)$ vs. $(e_1, e_2, e_3))$. Hence, in order to utilize the connectivity and contiguity of the road segments along each trajectory point, we define the *confidence* $\Pr(\langle p_k, e_{k,l} \rangle)$ for

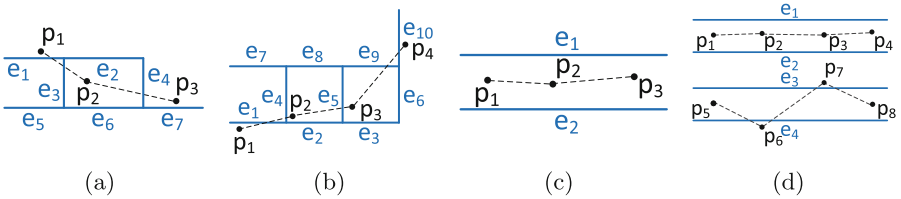


Fig. 3. Examples of the query selection strategy scenarios.

each candidate match $\langle p_k, e_{k,l}\rangle$ of p_k, and evaluate the uncertainty of p_k according to the Shannon entropy of the confidence distribution among all candidate matches for p_k. More specifically, given a trajectory T and a match $\langle p_k, e_{k,l}\rangle$, we generate a set of matches $M_{k,l}$ and a path $P_{k,l}$ by applying $Q(T, G, \{\langle p_k, e_{k,l}\rangle\})$ using the interactive map-matching algorithm. Thus,

$$\Pr(\langle p_k, e_{k,l}\rangle) = \Pr(P_{k,l}) = \prod_{i=1}^{n} \Pr(e_{i,j_i}|p_i) \times \prod_{i=1}^{n-1} \Pr(e_{i,j_i}, e_{i+1,j_{i+1}}|p_i, p_{i+1}) \quad (9)$$

where each $e_{i,j_i} \in P_{k,l}$.

Based on Formula (9), the uncertainty $H(p_i)$ of p_i is:

$$H(p_i) = -\sum_{j=1}^{r} \Pr(\langle p_i, e_{i,j}\rangle) \log(\Pr(\langle p_i, e_{i,j}\rangle)) \quad (10)$$

where r is the number of candidate states of p_i.

The next point recommended in each iteration is similar to the distance-based strategy with the uncertainty computed in Formula (10). The time complexity is $O(1)$ with respect to the number of points on the trajectory n.

5.3 Dynamic Confidence-Based Strategy

In the confidence-based strategy, the confidence for each candidate match of a point is defined under the assumption that all the other points of the trajectory are not map-matched. Consider the example shown in Fig. 3(c), where p_1, p_2 and p_3 are close to both e_1 and e_2. According to the confidence-based strategy, the probabilities of the paths that pass either e_1 or e_2 are similar, thus the uncertainty of p_1, p_2 and p_3 are similar. If there is another point of the trajectory that is wrongly matched but has a lower uncertainty, p_1, p_2 and p_3 will all be checked. However, if we have confirmed that p_2 is matched to e_2, the labels of p_1 and p_3 are no longer uncertain. Hence, the confidence-based method cannot prune the case when the match of a point is constrained by other points of the trajectory.

To deal with such situations, we propose a dynamic confidence-based strategy utilizing the set of labeled matches M' maintained by the system. More specifically, given a trajectory T, a set of labeled matches M', and a match

$\langle p_k, e_{k,l}\rangle$, we generate a set of matches $M_{k,l}$ and a path $P_{k,l}$ by applying $Q(T, G, M' \cup \{\langle p_k, e_{k,l}\rangle\})$ using the interactive map-matching algorithm. The confidence measure $\Pr(\langle p_k, e_{k,l}\rangle)$ is then defined as Formula (9), and the uncertainty $H(p_i)$ of p_i is defined as Formula (10).

In each iteration, since the set of labeled matches M' is updated, the uncertainty of each point should be re-calculated via Formula (10). Hence, the time complexity of the dynamic confidence-based strategy $O(n \times r)$, where n is the number of points on the trajectory, and r is the number of hidden states of each point. In practice, we restrict r in order to ensure high efficiency, which will be explained in Sect. 6.1. The next point recommended in each iteration is similar as the confidence-based strategy.

5.4 Stability-Based Strategy

Another strategy of uncertainty measurement is *stability*. Based on our observation, if the match of a point is frequently influenced by other points on the trajectory (i.e., the match of this point is not *stable*), the uncertainty of this point is often high. Consider the example shown in Fig. 3(d), p_1 has a similar probability to be matched to e_1 or e_2, and p_5 has a similar probability to be matched to e_3 or e_4. Therefore, the uncertainty of p_1 and p_5 are similar. However, since p_6 is much closer to e_3 while e_7 is much closer to e_4, there must be a large measurement noise for either of these two points. Hence, the match of p_5 is very unstable if either p_6 or p_7 is removed from the trajectory. On the contrary, p_2, p_3, p_4 are all slightly closer to e_1. Thus the match of p_1 is much more stable than p_5. In this case, since p_1 and p_5 belong to the same trajectory, the priority of checking p_5 is higher than p_1, because there is a higher probability that there exists a major measurement noise for the points around p_5.

In order to define the stability of a point, we first define the *influence* between two points. Given a trajectory $T = (p_1, p_2, \cdots, p_n)$ and two points $p_a, p_b \in T$, we first generate a set of matches M for T using our interactive map-matching algorithm. Next we generate another trajectory T_b omitting p_b, so that $T_b = (p_1, p_2, \cdots, p_{b-1}, p_{b+1}, \cdots, p_n)$, and then similarly obtain M_b for T_b. Suppose $\langle p_a, e_a\rangle \in M$, we denote that p_a is influenced by p_b as $p_a \prec p_b$ if and only if $\langle p_a, e_a\rangle \notin M_b$. Given a point $p_i \in T$, the stability $H(p_i)$ of p_i is:

$$S(p_i) = |D_{p_i}| \tag{11}$$

where D_{p_i} is the set of points that have no influence on p_i. More specifically, $D_{p_i} = \{p_1, p_2, ..., p_k\}$, where $p_j \in T$ and $p_i \not\prec p_j$ for all $p_j \in D_{p_i}$.

Based on Formula (11), given a trajectory T, in each iteration of the interactive map-matching process, the next point recommended is:

$$p' = \arg\min_{p_i \in T} S(p_i) \tag{12}$$

The stability-based strategy is efficient for selecting problematic points. However, since $S(p_i)$ is defined as a cardinality rather than a probability, it is possible

that the points of a trajectory have the same $S(p_i)$. In this case, the stability-based method is not able to determine an efficient order for these points. To deal with this case, we dynamically switch to dynamic confidence-based strategy in each iteration if two points have the same $S(p_i)$. Hence, the time complexity of the stability-based strategy is between $O(n)$ and $O(n \times r)$. Similar to dynamic confidence-based strategy, we restrict r to ensure high efficiency.

6 Evaluation

6.1 Experiment Setup

Experiment Environment. The experiments are conducted on a Linux server with a CPU of Intel Core i5-4590 and 8 GB memory. The operating system is Ubuntu 14.04, and the code is written in Python 2.7.6.

Road Network. In our experiments, the road network data is provided by the government of a large city in China and consisted of 25,613 intersections and 36,451 road segments. There are no direction information thus all the road segments are bi-directional.

Synthetic Trajectory Data. We build a trajectory generator to generate synthetic trajectories with the following parameters: (1) the number of points n_p on the trajectory, (2) the number of road segments n_e covered by the trajectory, and (3) the standard deviation δ of the positioning *measurement noise*.

The trajectory generator selects a starting point $p_1 \in e_1$ and an ending point $p_{n_p} \in e_{n_e}$ from an n_e-hop random path $P = (e_1, e_2, \ldots, e_{n_e})$, and then computes the *distance interval* Δ between two consecutive points:

$$\Delta = \frac{\text{route}(p_1, p_{n_p})}{n_p - 1} \tag{13}$$

where $\text{route}(p_1, p_{n_p})$ is the driving distance between p_1 and p_{n_p}. Starting from p_1, the trajectory generator derives the locations of the remaining points along P such that $\text{route}(p_1, p_{i+1}) = \text{route}(p_1, p_i) + \Delta$. Finally, the generator adds a Gaussian distributed measurement noise to each point with δ so that $T = (\delta(p_1), \delta(p_2), \ldots, \delta(p_{n_p}))$.

In our experiments, we study the impact of different parameters including: (1) the *number of points* on a trajectory; (2) the *initial accuracy* of a trajectory in terms of the number of points that are matched to the correct road segments by comparing $Q(T, G, \emptyset)$ with P; (3) the *sampling rate* of a trajectory which is represented by n_e given a fixed average driving speed; and 4) the standard deviation of the *measurement noise*.

Real Trajectory Data. Our real world trajectory data contains 154 million records of 15,231 vehicles for 26 days [4].

Evaluation Metrics. To evaluate the effectiveness of our interactive map-matching algorithm, we simulate the work flow of HIMM with the assumption that each query annotation is correct and the time of trajectory review is trivial. In order to reduce the response time, for each point on the trajectory, we empirically reserve the top 10 nearest road segments as its candidate set. In our experiments, it is sufficient to produce a high percentage of the correct matches within this range. The correct road segment is also added into the candidate set in case it is excluded.

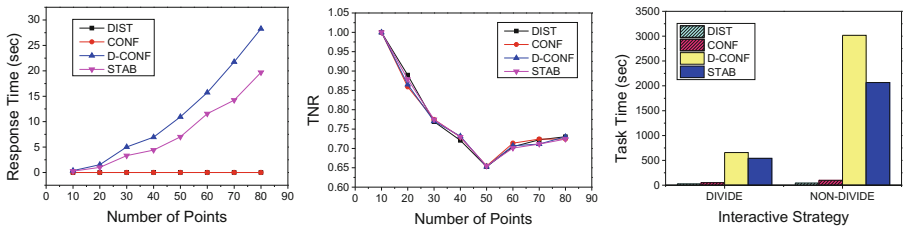
We use two metrics to evaluate the efficiency of our query selection strategies: (1) the response time for a query selection strategy; and (2) the execution time of an interactive map-matching task for a single trajectory. In addition, we define three metrics to evaluate the effectiveness of our query selection strategies. Given a trajectory T along with the initial path generated by HIMM, the number of mis-matched points is denoted as $\zeta(T)$. After the map-matching task for T is terminated, the total number of points that are reviewed by the annotator is denoted as $\eta(T)$, and the total number of points that are corrected by the annotator is denoted as $\psi(T)$. The three metrics are defined as follows: (1) ***cost ratio*** $\texttt{CR} = \eta(T)/|T|$ representing the review cost of the interactive map-matching task; (2) ***selection accuracy*** $\texttt{SA} = \psi(T)/\eta(T)$ representing the accuracy of selecting mis-matched points; and (3) ***true negative rate*** $\texttt{TNR} = \psi(T)/\zeta(T)$ representing the ratio of the corrections conducted by the annotator rather than the interactive map-matching algorithm. A lower `CR` indicates fewer iterations for a map-matching task; a higher `SA` indicates a higher rate of selecting mis-matched points; and a lower `TNR` indicates more points are automatically corrected by our interactive map-matching algorithm.

6.2 Experiment Results

The experiments are conducted on both synthetic and real data. For each data set, we apply all the query selection strategies proposed in this paper: *distance-based strategy* (`DIST`), *confidence-based strategy* (`CONF`), *dynamic confidence-based strategy* (`D-CONF`), and *stability-based strategy* (`STAB`); as well as two baselines: *sequential strategy* (`SEQ`) and *random strategy* (`RAND`), where the annotator checks each point along the trajectory in a sequential and random order, respectively.

Efficiency. In order to evaluate the scalability of our query selection strategies, we conduct experiments on 8 groups of trajectories whose numbers of points range from 10 to 80. The initial accuracy is fixed within 60–70%. The sampling rate and measurement noise are fixed to 1.5 min and 101.04 m, respectively.

Figure 4(a) shows the impact of number of points on response time. Consistent with the analysis in Sects. 5.3 and 5.4, the response time of `D-CONF` and `STAB` increases linearly with the number of points. Figure 4(b) shows the impact of number of points on `TNR`, where `TNR` drops at first when the number of points grows, but rises after the number of points reaches 50. Therefore,



(a) Response Time (b) True Negative Rate (c) Task Execution Time

Fig. 4. Performance of query selection strategies.

HIMM achieves the best performance when the number of points on the query selection trajectory is 50. Hence, for long trajectories, the most efficient strategy is to divide them into sub-trajectories with 50 points each, and perform an interactive map-matching task for each sub-trajectory. In order to show the effectiveness of this dividing strategy, we generate a group of trajectories consisting of 100 points each, and then compare the average execution time of the interactive map-matching task for each trajectory with or without using the dividing strategy. As a result, Fig. 4(c) shows that dividing long trajectories into sub-trajectories significantly reduces the task time.

Effectiveness on Synthetic Data. To study the impact of initial accuracy, we generate 5 groups of trajectories with 5 categories of initial accuracy: 50–60%, 60–70%, 70–80%, 80–90%, and 90–100%. The sampling rate and measurement noise are fixed to 1.5 min and 101.04 m respectively. To study the impact of sampling rate, we generate 3 groups of trajectories with 3 categories of sampling rates: 0.5, 1.5, and 4.5 min. The initial accuracy and measurement noise are fixed to 70–80% and 11.23 m respectively. To study the impact of measurement noise, we generate 3 groups of trajectories with 3 categories of measurement noises: 11.23, 33.68, and 101.04 m. The initial accuracy and sampling rate are fixed to 60–70% and 0.5 min respectively.

The experiment results on synthetic trajectory data are shown in Fig. 5. In general, the performance (in terms of `CR` and `SA`) of our query selection strategies (`DIST`, `CONF`, `D-CONF`, and `STAB`) achieve a much higher efficiency than the two baseline strategies (`SEQ` and `RAND`). Among our query selection strategies, the performance of `D-CONF` is better than the two global strategies (`DIST` and `CONF`), and `STAB` outperforms the other three strategies. Compared with the two baseline strategies, `CR` reduced by our query selection strategies is up to 44%, and `SA` is improved up to 24%.

Moreover, the `TNR` results in Fig. 5 show that the percentage of mis-matched points that are automatically corrected by the interactive map-matching algorithm during human annotation is up to 59%, which indicates a significant reduction of the annotation cost.

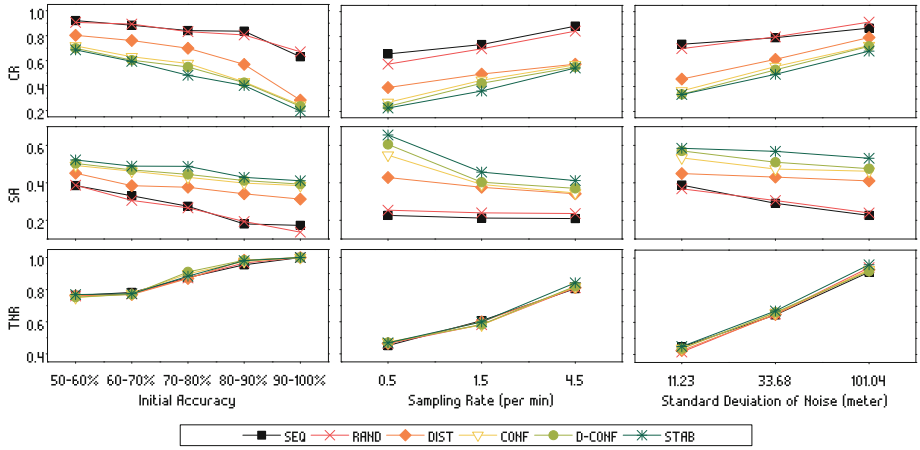
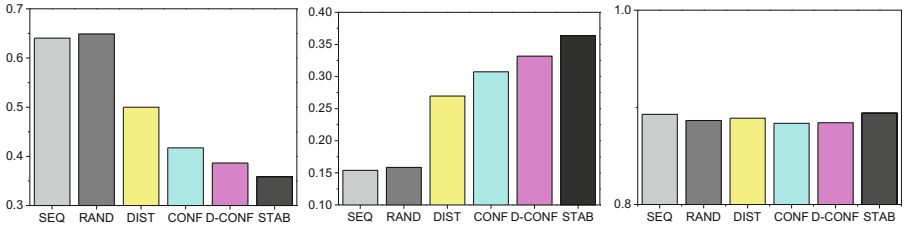


Fig. 5. Effectiveness of query selection strategies on the synthetic trajectory data.

Next, we discuss the impact of each parameter on the performance of our query selection strategies. Firstly, we observe that the gaps of `CR` and `SA` between baseline strategies and our query selection strategies enlarge when initial accuracy grows. This indicates that our query selection strategies are more effective in picking out wrongly matched points when the initial accuracy is high. Meanwhile, `TNR` decreases when the initial accuracy falls for all the query selection strategies. This indicates that the ratio of the automatic corrections triggered by human annotation rises when the initial accuracy is low, which also saves the annotation cost. In conclusion, HIMM can reduce $\eta(T)$ no matter the initial map-matching accuracy is low or high.

Secondly, we observe that a larger sampling rate or measurement noise will hurt the performance both in `CR` and `SA` for all the query selection strategies. However, compared with the baseline strategies, our query selection strategies are more sensitive to sampling rate, but less sensitive to measurement noise. This is because a larger sampling rate reduces the topological correlations between points, thus the advantage of our query selection strategies is less effective. In contrast, a larger measurement noise only increases the deviation of each point within its local area rather than the topological information, thus the advantage of our query selection strategies remains. As a result, our query selection strategies outperform the baseline strategies in most of the cases.

Effectiveness on Real Data. Since the cost of a manual map-matching task is very high, due to the limit of time, we manually processed 200 trajectories with 50 points each. We use HIMM to annotate these trajectories, and record the resulting paths as the ground truth. For the experiments on real trajectory data, based on our statistics, the initial accuracy is 89% on average; the sampling rate ranges from 30 s to 5 min; and the measurement noise is around 33.68 m.



(a) Cost Ratio (b) Selection Accuracy (c) True Negative Rate

Fig. 6. Effectiveness of query selection strategies on the real trajectory data.

The experiment results on real trajectory data are shown in Fig. 6. It is clear that the effectiveness of our query selection strategies are much higher than the baseline strategies in terms of both `CR` and `SA`. Similar to the experiments on synthetic trajectory data, the performance of `STAB` is the best, which reduces 29% of `CR` and improves 21% of `SA` compared with baseline strategies. Moreover, 12% mis-matched points are automatically corrected by the interactive map-matching algorithm, which is a satisfactory result for such a high initial accuracy.

In general, the performance of HIMM on the real trajectory data is similar to that on the synthetic trajectory data, which indicates that HIMM achieves a satisfactory performance on a wide range of trajectories, and significantly reduces the annotation cost.

7 Conclusion

In this paper, we propose an interactive map-matching system called HIMM for the annotators to perform effective interactive map-matching tasks. We design and implement an interactive map-matching algorithm that can be improved by manual annotations, and propose four different query selection strategies to reduce the costs of interactive map-matching tasks. We conduct intensive experiments on both synthetic and real trajectory data. The results show that our query selection strategies achieve a satisfactory performance. In this paper, we only consider single annotators for interactive map-matching tasks, and it could be further discussed when multiple annotators and crowd-sourcing are introduced, which will be our future work.

Acknowledgments. This work is supported in part by NSFC Grant 61300030 and the National Key Basic Research and Development Program of China (973) Grant 2014CB340304.

References

1. Bilgic, M., Mihalkova, L., Getoor, L.: Active learning for networked data. In: Proceedings of ICML, pp. 79–86 (2010)
2. Brakatsoulas, S., Pfoser, D., Salas, R., Wenk, C.: On map-matching vehicle tracking data. In: Proceedings of VLDB, pp. 853–864. VLDB Endowment (2005)
3. Ding, Y., Liu, S., Pu, J., Ni, L.M.: HUNTS: a trajectory recommendation system for effective and efficient hunting of taxi passengers. In: Proceedings of MDM, vol. 1, pp. 107–116. IEEE (2013)
4. Ding, Y., Zheng, J., Tan, H., Luo, W., Ni, L.M.: Inferring road type in crowdsourced map services. In: Bhowmick, S.S., Dyreson, C.E., Jensen, C.S., Lee, M.L., Muliantara, A., Thalheim, B. (eds.) DASFAA 2014. LNCS, vol. 8422, pp. 392–406. Springer, Cham (2014). doi:10.1007/978-3-319-05813-9_26
5. Jagadeesh, G., Srikanthan, T., Zhang, X.: A map matching method for GPS based real-time vehicle location. J. Navig. **57**(03), 429–440 (2004)
6. Karimi, H.A., Conahan, T., Roongpiboonsopit, D.: A methodology for predicting performances of map-matching algorithms. In: Carswell, J.D., Tezuka, T. (eds.) W2GIS 2006. LNCS, vol. 4295, pp. 202–213. Springer, Heidelberg (2006). doi:10.1007/11935148_19
7. Liao, L., Patterson, D.J., Fox, D., Kautz, H.: Learning and inferring transportation routines. Artif. Intell. **171**(5), 311–331 (2007)
8. Lou, Y., Zhang, C., Zheng, Y., Xie, X., Wang, W., Huang, Y.: Map-matching for low-sampling-rate GPS trajectories. In: Proceedings of SIGSPATIAL, pp. 352–361. ACM (2009)
9. Newson, P., Krumm, J.: Hidden Markov map matching through noise and sparseness. In: Proceedings of SIGSPATIAL, pp. 336–343. ACM (2009)
10. Pink, O., Hummel, B.: A statistical approach to map matching using road network geometry, topology and vehicular motion constraints. In: Proceedings of ITSC, pp. 862–867. IEEE (2008)
11. Rabiner, L.R.: A tutorial on hidden Markov models and selected applications in speech recognition. Proc. IEEE **77**(2), 257–286 (1989)
12. Settles, B.: Active learning literature survey, vol. 52, no. 55–66, p. 11. University of Wisconsin, Madison (2010)
13. Settles, B., Craven, M.: An analysis of active learning strategies for sequence labeling tasks. In: Proceedings of Empirical Methods in Natural Language Processing, pp. 1070–1079. Association for Computational Linguistics (2008)
14. Wang, G., Zimmermann, R.: Eddy: an error-bounded delay-bounded real-time map matching algorithm using HMM and online Viterbi decoder. In: Proceedings of SIGSPATIAL, pp. 33–42. ACM (2014)
15. Xue, A.Y., Qi, J., Xie, X., Zhang, R., Huang, J., Li, Y.: Solving the data sparsity problem in destination prediction. VLDB J. **24**(2), 219–243 (2015)