Neurocomputing 444 (2021) 126–135

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

iMatching: An interactive map-matching system [☆]

Ye Ding [a], Xibo Zhou [c], Qing Liao [b,*], Haoyu Tan [c], Qiong Luo [c], Lionel M. Ni [c]

[a] School of Cyberspace Security, Dongguan University of Technology, China
[b] Department of Computer Science and Technology, Harbin Institute of Technology (Shenzhen), China
[c] Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong

ARTICLE INFO

Article history:
Received 20 January 2020
Revised 23 April 2020
Accepted 26 April 2020
Available online 26 January 2021

Keywords:
Map-matching
Active learning
Trajectory analysis
Urban computing

ABSTRACT

Map-matching is a process that aligns location points on a digital map and it is an essential step in location-based services. However, regular map-matching methods cannot archive very high accuracy due to the errors in raw location data and the complexity of road networks. Hence, the final resort for map matching is often through manual annotation, which is human labour intensive. Therefore, we propose *iMatching*, an interactive system for map-matching which greatly reduces annotation cost and achieves a high accuracy through an active learning approach. Specifically, we model the mapping of a sequence of location points to a road network as a hidden Markov model and automatically generate an initial result. Then, we select error-prone points on the trajectory and guide the annotator to review, and possibly correct, the results. Our evaluation on both real-world and synthetic data demonstrates that iMatching has a better performance comparing with the existing methods.

© 2021 Elsevier B.V. All rights reserved.

1. Introduction

In location-based services and vehicle monitoring systems, precise traces of locations (or *trajectory*) of the moving objects are essential. However, the precision of a trajectory is often affected by sampling errors and measurement noises in real life. To solve such imprecision problem, map-matching [2] is proposed to align the location points of a trajectory to correct corresponding road segments. For example, in Fig. 1, trajectory $(p_1, p_2, p_3, p_4, p_5)$ is matched to path $(e_1, e_2, e_9, e_5, e_{12})$.

The major challenge of map-matching is that since a trajectory is composed of a series of "sampling" location points, the information between two sampling points are missing. Hence, it is difficult to precisely predict the actual location of a sampling point. In order to retrieve the missing information, we have to employ human labours driving through the roads and collecting the ground truth. These human annotations may lead to a very high cost.

To reduce the costs of manual annotations, one possible approach is to rank the uncertainties of the location points produced by a specific map-matching algorithm, and then ask the human annotator to fix the "uncertain" points one-by-one. After each uncertain point is fixed, the map-matching algorithm may be applied again and generate new rankings with fewer uncertain points, since more ground truth ("fixed" points) are introduced to the training data. This approach is similar to active learning [3]. However, there are two issues in this approach: 1) since the map-matching algorithm is completely re-applied in each round, the time complexity of the entire approach could be very high; and 2) it is difficult to define the "uncertainty" of matched points.

We propose *iMatching*, an active-learning-based system for map-matching that 1) utilizes the responses of annotators through an interactive map-matching algorithm; and 2) poses effective and efficient queries to the annotator through adaptive query selection strategies. As far as we know, none of the map-matching works could utilize the annotation feedbacks within the process of a map-matching task, and existing query selection strategies from active learning are not specifically designed for the map-matching task.

The main contributions of this paper include:

- We propose and implement a novel interactive map-matching system called *iMatching*, which reduces the costs of manual annotations and achieves a high map-matching precision simultaneously;
- We design a novel interactive map-matching algorithm based on hidden Markov model which could utilize the feedbacks of human annotators;

[☆] An earlier version of this work appeared in the Proceedings of the 22nd International Conference on Database Systems for Advanced Applications [1], March 2017.
[*] Corresponding author.

E-mail addresses: dingye@dgut.edu.cn (Y. Ding), xzhouaa@ust.hk (X. Zhou), liaoqing@hit.edu (Q. Liao), tanhaoyu@ust.hk (H. Tan), luo@ust.hk (Q. Luo), ni@ust.hk (L.M. Ni).

https://doi.org/10.1016/j.neucom.2020.04.155
0925-2312/© 2021 Elsevier B.V. All rights reserved.

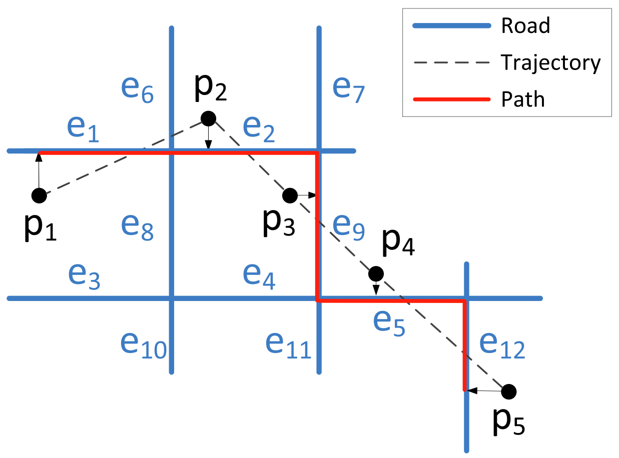


Fig. 1. An example of map-matching.

- We implement various adaptive query selection strategies which reduce the number of queries posed to the human annotators by up to 44%;
- We evaluate the performance of iMatching on both real-world and synthetic trajectory data to show that iMatching has a better performance comparing with the existing methods.

2. Related works

2.1. Map-matching

There are three types of map-matching algorithms: geometric-based, topological-based, and statistical-based. Geometric algorithms [2,4] are based on the geometric information of the spatial road network to identify the actual location of each sampling point. Although these algorithms are able to process map-matching queries efficiently, the performance is sensitive to sampling errors and measurement noises. Topological algorithms [5–7] consider additional information such as the continuity and connectivity of the road network to obtain better performance than geometric algorithms. However, the accuracy of topological algorithms may decrease on the trajectories with low sampling rates. Statistical algorithms, such as voting [8], hidden Markov model [9,10], particle filter [11], Kalman filter [12], and average Fréchet distance [13], perform better than geometric and topological algorithms, and could be accelerated by GPU [14–19]. However, these methods often have high time complexity, and cannot utilize the feedbacks of human annotators to improve the map-matching accuracy.

2.2. Active learning

The goal of active learning is to obtain the desired outputs at new data points through querying the information source interactively [3,20]. There are two types of strategies that define the informativeness of an instance: uncertainty-based strategies [21,22] and utility-based strategies [23–25]. Uncertainty-based strategies select the instances whose labels are the least certain, whereas utility-based strategies select the instances that have the most impact on the learning task. However, most active learning methods cannot be directly applied to map-matching tasks due to the structure difference between regular data and trajectory data. Trajectory has additional features in both spatial and temporal domains, and the features are often intensively co-related and varying over time. Most of the existing active learning approaches cannot handle such data structure.

3. Overview

Before introducing the definition of the interactive map-matching problem, let us first define the following terms which are frequently used in this paper.

3.1. Problem definition

Definition 1 (*Trajectory*). A trajectory T is a sequence of location points (a.k.a. sampling points) $(p_1, p_2, \ldots, p_n)$.

Definition 2 (*Road Segment*). A road segment (a.k.a. edge) $e = \langle v_i, v_j \rangle$ is a polygonal line between two intersections (a.k.a. vertices) v_i and v_j.

In Definition 2, it is called road "segment" because there will be no other interactions within the road segment, and a "road" in real life often consists of a series of road segments.

Two road segments e_i and e_j are called *connected* if there is a vertex v such that $v \in e_i$ and $v \in e_j$.

Definition 3 (*Road Network*). A road network $G = (V, E)$ is a directed graph consists of a set of intersections $V = \{v_1, v_2, \ldots, v_m\}$ and a set of road segments $E = \{e_1, e_2, \ldots, e_n\}$.

In Definition 3, the graph is directed because road segment often has a direction restriction (e.g., one way) in real life. In this paper, we assume that a moving object only travels on the road network, i.e., the vehicle will not drive off-road.

Definition 4 (*Path*). A path P is a sequence of road segments $(e_1, e_2, \ldots, e_n)$ where e_i and e_{i+1} are connected for any $e_i, e_{i+1} \in P$.

Definition 5 (*Match*). A match $m_i = \langle p_i, e_j \rangle$ indicates that the moving object is travelling along $e_j \in P$ when $p_i \in T$ is collected by the positioning device.

Definition 6 (*Map-Matching Query*). A map-matching query $Q(T, G)$ is a query which finds a path P and a set of matches M such that for every $p_i \in T$, there exists exactly one match $m_i \in M$ where $m_i = \langle p_i, e_j \rangle$ and $e_j \in P$.

Based on the above terms, the problem of the interactive map-matching query is defined in Definition 7.

Definition 7 (*Interactive Map-Matching Query*). An interactive map-matching query $Q(T, G, M')$, where M' is a set of matches confirmed by the annotators, is a query that finds a path P and a match set M such that for every $p_i \in T$, there exists exactly one match $m_i \in M$ where $m_i = \langle p_i, e_j \rangle, e_j \in P$, and $M' \subset M$.

3.2. Framework of iMatching system

The *iMatching* system mainly consists of the following parts as shown in Fig. 2:

1. An interactive map-matching algorithm which perform interactive map-matching queries with consideration of human annotations; and
2. A query selection strategy which pick and pose an uncertain match for human annotators to confirm.

The system works as follows. Initially, a trajectory is aligned by the map-matching algorithm. Then, the resulting path will be posed to the annotators to review whether it is 100% accurate. If

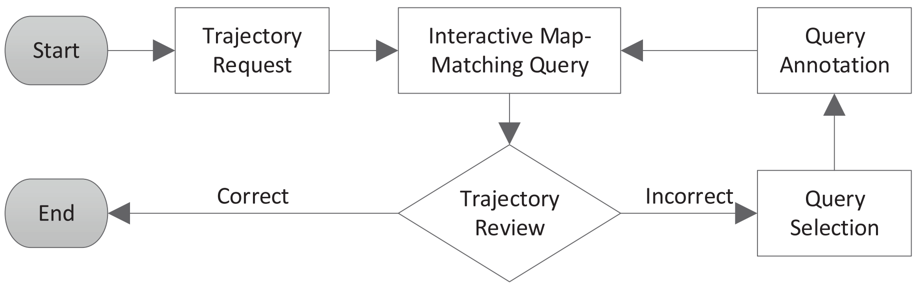


Fig. 2. The framework of *iMatching*.

so, the map-matching task is terminated. Otherwise, the system will pick and pose an uncertain match for the human annotators to confirm. After the match is confirmed, the system will perform the (interactive) map-matching query again and try to fix as many mistakenly matched location points as possible before asking the annotators to review the trajectory again.

The details of the interactive map-matching algorithm is illustrated in Section 4, and the query selection strategies are introduced in Section 5.

4. Interactive map-matching

4.1. Map-matching mmodel

A typical map-matching algorithm works as follows. First, it picks a location point p_i from the trajectory $T = \{p_1, p_2, \ldots, p_n\}$ and then find the "nearest" road segment $e_i \in E$ as the matching of p_i. However, due to that only geometric information may lead to many errors such as detours (the vehicle travels from e_1 to e_6 and then back to e_2 in Fig. 1, where e_6 is clearly a mistake), many works use topological information such like the connectivities between road segments to conquer them. As for the above example, if we know that e_1 and e_2 are straightly connected, it would be irregular if the point p_2 is matched to e_6, and it should be matched to e_2.

The above algorithm could be modelled as a hidden Markov model [9]. The hidden Markov model for map-matching is shown in Fig. 3. The geometric information of the map-matching task is shown as the states of the hidden Markov model, referring to $e_{1,1}, e_{1,2}, \ldots, e_{1,r}$, where $r = |E|$. Meanwhile, the topological information fo the map-matching task is labelled as the transitions of states.

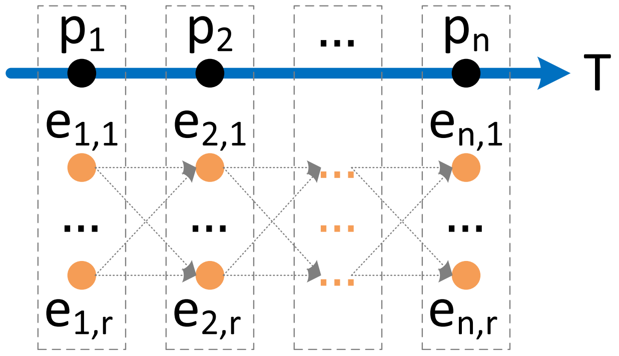


Fig. 3. The interactive map-matching model.

The *emission probability* of the hidden Markov model is represented by the geometric information of the map-matching task, where the "nearest" road segment has the highest probability to be the correct matching. More specifically,

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

where δ is the measurement noise in terms of standard deviation, and we use *minimum perpendicular distance* [26] $\mathrm{pdist}(e_{i,j}, p_i)$ to represent the distance between a road segment $e_{i,j}$ and a location point p_i. The emission probability follows a Gaussian distribution [9].

The transition probability of the hidden Markov model is represented by the topological information of the map-matching task, where each transition is denoted as the probability of p_{i+1} being matched to $e_{i+1,j_{i+1}}$ if p_i is matched to $e_{i,j}$. More specifically,

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

where β is the rate parameter [9]. The topological difference between different transitions are represented by the absolute difference of the driving distance $\mathrm{route}(p_i, p_{i+1})$ and the direct physical distance (in this paper, the great circle distance) $\mathrm{cdist}(p_i, p_{i+1})$ between p_i and p_{i+1}.

4.2. Interactive map-matching algorithm

As defined in Definition 6, the goal of a map-matching query is to identify an "optimal" path P such that all $p_i \in T$ are matched to some road segment $e_i \in E$. In terms of the hidden Markov model, it aims to find a series of hidden states $P = (e_{1,j_1}, e_{2,j_2}, \ldots, e_{n,j_n})$ such that the joint probability $\Pr(P)$ of P is the highest. Formally,

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

The optimal solution P^* is *argmaxPr*(P). There are many algorithms designed to find the optimal solution for the hidden Markov model. For example, the Viterbi algorithm [27]. However, these algorithms cannot take any feedbacks from the annotators in each iteration. Hence, traditional algorithms are not sufficient for the interactive map-matching task.

Hence, we introduce a novel interactive map-matching algorithm in this paper which extends the Viterbi algorithm to support taking feedbacks from annotators.

The *initial formulation* is defined as:

$$C(1,j) = \begin{cases} 1 & \langle p_1, e_{1,k}\rangle \in M', \ k = j \\ 0 & \exists\langle p_i, e_{i,k}\rangle \in M', \ k \neq j \\ \Pr(e_{i,j}|p_i) & \nexists\langle p_i, \ e_{i,k}\rangle \in M' \end{cases} \tag{4}$$

where $1 \leqslant k \leqslant r$.

The *forward formulation* [27] is defined as:

$$C(i,j) = \begin{cases} O(i,j) & \langle p_i, e_{i,k}\rangle \in M', k = j \\ 0 & \exists \langle p_i, e_{i,k}\rangle \in M',\ k \neq j \\ \Pr(e_{i,j}|p_i) \times O(i,j) & \nexists \langle p_i,\ e_{i,k}\rangle \in M' \end{cases} \tag{5}$$

$$O(i,j) = \max_{1 \leqslant k \leqslant r} C(i-1,k)\Pr(e_{i-1,k}, e_{i,j}|p_{i-1}, p_i) \tag{6}$$

where $1 \leqslant k \leqslant r, i > 1$, and $O(i,j)$ is the recursive variable.

In Formula (5), $C(i,j)$ is the highest probability of the sequences of states $P_i = (e_{1,j_1}, e_{2,j_2}, \ldots, e_{i,j_i})$ for the first i observations $T_i = (p_1, p_2, \ldots, p_i)$ with $e_{i,j}$ as the final state.

In order to inject the feedbacks from annotators, in each iteration of the forward formulation, if $\langle p_i, e_{i,j}\rangle \in M'$, indicates that there exists one match $\langle p_i, e_{i,j}\rangle$ provided by the annotators, we will 1) set $Pr(e_{i,j}|p_i)$ to 1 and resulting $C(i,j) = 1 \times O(i,j) = O(i,j)$; and 2) set $Pr(e_{i,k}|p_i)$ to 0 for all $1 \leqslant k \leqslant r$ and $k \neq j$, and resulting $C(i,j) = 0 \times O(i,j) = 0$. Otherwise, the forward formulation works as normal in Formula (5).

The forward procedure terminates when the last observation p_n is handled. Then the optimal sequence of states with respect to $C(i,j)$ is fetched by the *backward formulation* [27] defined as:

$$e_{i,j_i} = \begin{cases} \arg\max_{1 \leqslant j \leqslant r} C(i,j) & i = n \\ \arg\max_{1 \leqslant j \leqslant r} \frac{C(i+1,j)}{\Pr(e_{i,j}, e_{i+1,j_{i+1}}|p_i, p_{i+1})} & 1 \leqslant i < n \end{cases} \tag{7}$$

In this paper, we use dynamic programming to find the optimal solution P^*. The detailed implementation of the algorithm is shown in Algorithm 1.

Algorithm 1: Interactive Map-Matching Algorithm

Input: $T = (p_1, p_2, \ldots, p_n)$, $G = (V, E)$, M'

Output: $P^* = (e_{1,j_1}, e_{2,j_2}, \ldots, e_{n,j_n})$

```
1. for all $e_{1,j} \in E$ do
2.   if $m(p_1, e_{1,j}) \in M'$ then
3.     $C(1,j) \leftarrow 1$
4.   else if $m(p_1, e_{1,k}) \in M', \exists k \neq j, 1 \leqslant k \leqslant r$ then
5.     $C(1,j) \leftarrow 0$
6.   else
7.     $C(1,j) \leftarrow \Pr(e_{1,j}|p_i)$
8.   end if
9. end for
10. for $2 \leqslant i \leqslant n$ do
11.   for all $e_{i,j} \in E$ do
12.     if $m(p_i, e_{i,j}) \in M'$ then
13.       $C(i,j) \leftarrow \max_{1 \leqslant k \leqslant r} C(i-1,k)\Pr(e_{i-1,k}, e_{i,j}|p_{i-1}, p_i)$
14.     else if $m(p_i, e_{i,k}) \in M', \exists k \neq j, 1 \leqslant k \leqslant r$ then
15.       $C(i,j) \leftarrow 0$
16.     else
17.       $C(i,j) \leftarrow \Pr(e_{i,j}|p_i) \times \max_{1 \leqslant k \leqslant r} C(i-1,k)\Pr(e_{i-1,k}, e_{i,j}|p_{i-1}, p_i)$
18.     end if
19.   end for
20. end for
21. $e_{n,j_n} \leftarrow argmax_{1 \leqslant k \leqslant r} C(n,k)$
22. for $n-1 \geqslant i \geqslant 1$ do
23.   $e_{i,j_i} \leftarrow argmax_{1 \leqslant k \leqslant r} \frac{C(i+1,k)}{\Pr(e_{i,k}, e_{i+1,j_{i+1}}|p_i, p_{i+1})}$
24. end for
25. return $P^* = (e_{1,j_1}, e_{2,j_2}, \ldots, e_{n,j_n})$
```

5. Adaptive query selection

As introduced in Section 1, a good query selection strategy could reduce the cost of the interactive map-matching query. Intuitively, when a trajectory is considered as incorrectly map-matched by an annotator, the annotator is likely to check the points along the trajectory to find out mismatched points in a sequential order or a random order. For the sequential checking order, the number of points to be checked is determined by the position of the mismatched points. For example, if the 49th point of a trajectory that contains 50 points is mismatched, the annotator has to check 49 points in order to find the mismatched point. During the process, a large portion of correct matches are checked, which is unnecessary and wasteful. For the random checking order, the number of points to be checked may vary even for the same trajectory due to the randomness. In this case, the number of unnecessary checks can also be very large. Therefore, it is desirable to have a good query selection strategy that provides candidate points that are likely to be mismatched to the annotator. Next, we will introduce several adaptive query selection strategies respectively.

5.1. Distance-based strategy

Uncertainty sampling is one conventional method introduced by active learning for query selection, where the least certain item is picked in each iteration for the annotators to confirm. Specifically in the map-matching task, the "uncertainty" could be defined as the distance ambiguity for a location point p_i towards a set of candidate road segments $E_i = \{e_{i,1}, e_{i,2}, \ldots, e_{i,r}\}$ where $r = |E|$. For example, in Fig. 4, p_1 and p_3 are obviously closer to e_1 and e_7 respectively, but p_2 is ambiguous since it is close to both e_2, e_3, and e_6. Hence, in this example, p_2 should be the most "uncertain" item that should be confirmed by the annotators.

Formally, we use Shannon entropy [3] to model the uncertainty $H(p_i)$ as the distance ambiguity of a location point p_i:

$$H(p_i) = -\sum_{j=1}^{r} \Pr(e_{ij}|p_i)\log(\Pr(e_{ij}|p_i)) \tag{8}$$

Hence, given a trajectory T, the next location point p' that should be confirmed by the annotators is defined as:

$$p' = \underset{p_i \in T}{argmax}\, H(p_i) \tag{9}$$

In each iteration, if a location point p_i is confirmed by the annotator, $H(p_i)$ will be set to $-\infty$ to avoid duplicate confirmations. In conclusion, $O(1)$ is the time complexity of the distance-based strategy.

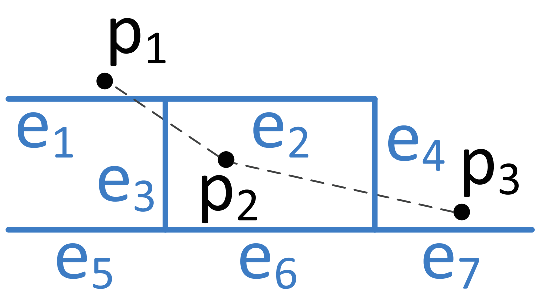


Fig. 4. Example of the scenario considered by distance-based strategy.

5.2. Confidence-based strategy

The distance-based query selection strategy is purely geometric and lack of consideration to the topological information. For example in Fig. 5, p_2 is close to both e_2 and e_4, while p_3 is also close to both e_3 and e_5. However, we can see that p_3 is slightly closer to e_5 than e_3. Hence, according to the distance-based strategy, p_2 will be considered as most uncertain.

If we consider the topological information of the road network, either p_3 is matched to e_3 or e_5 will only affect the sub-path choices of (e_3, e_6) or (e_5, e_9). However, the paths differ a lot if p_2 is matched to e_2 or e_4. If p_2 is matched to e_2, the path could be $(e_1, e_2, e_3, e_6, e_{10})$ or $(e_1, e_2, e_5, e_9, e_{10})$. But if p_2 is matched to e_4, only $(e_1, e_4, e_8, e_5, e_3, e_6, e_{10})$ will be the resulting path and this path is quite irregular in real life. Hence, it is more important to confirm p_2 before p_3 if we consider the topological information of the road network.

Hence in this section, we define a *confidence* probability $\Pr(\langle p_k, e_{k,l}\rangle)$ for each candidate match $\langle p_k, e_{k,l}\rangle$ of a location point p_k. By applying the interactive map-matching query $Q(T, G, \{\langle p_k, e_{k,l}\rangle\})$, we have a match set $M_{k,l}$ and a local optimal path $P_{k,l}$. Formally,

$$Pr(\langle p_k, e_{k,l}\rangle) = Pr(P_{k,l})$$
$$= \prod_{i=1}^{n} \Pr(e_{i,j_i}|p_i) \times \prod_{i=1}^{n-1} \Pr(e_{i,j_i}, e_{i+1,j_{i+1}}|p_i, p_{i+1}) \tag{10}$$

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

Similar to Formula (10), the uncertainty $H(p_i)$ of location point p_i is defined as:

$$H(p_i) = -\sum_{j=1}^{r} \Pr(\langle p_i, e_{i,j}\rangle) \log(\Pr(\langle p_i, e_{i,j}\rangle)) \tag{11}$$

The next location point p' that should be confirmed by the annotators is the same as Formula (9). Similar to the distance-based strategy, $O(1)$ is also the time complexity of the confidence-based strategy.

5.3. Dynamic confidence-based strategy

The above two strategies will scan all the location points $p_i \in T$ with $H(p_i) \neq -\infty$ and check if p_i fits the uncertainty criteria. Hence, they are not efficient when some locations points have influence on each other. For example, in Fig. 6, p_1, p_2 and p_3 are all uncertain with respect to e_1 and e_2. However, it is clear that if one of them is matched to either e_1 or e_2, the rest of them should also be matched to the same road segment, because the vehicle cannot switch between e_1 and e_2 due to they are not even connected. Hence, confirming only one of them is enough for the map-matching algorithm, and there is no need to let the annotators confirm all of them. This further indicates that the confidence-based strategy should recalculate the confidence $\Pr(\langle p_k, e_{k,l}\rangle)$ for all the location points in each iteration before applying the map-matching query.

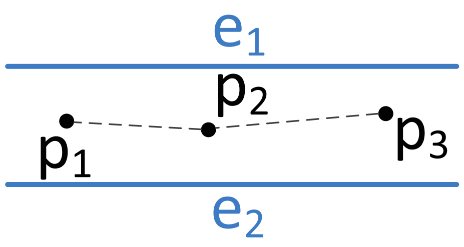


Fig. 6. Example of the scenario considered by dynamic confidence-based strategy.

More specifically, instead of generating the confidence probability through $Q(T, G, \{\langle p_k, e_{k,l}\rangle\})$, the dynamic confidence-based strategy generates the confidence probability through $Q(T, G, M' \cup \{\langle p_k, e_{k,l}\rangle\})$ in each iteration, where M' is the cumulative matches confirmed by the annotators:

$$M' = \bigcup_{\forall p_i \in T, H(p_i) = -\infty} \{\langle p_i, e_{i,j_i}\rangle\} \tag{12}$$

The uncertainty $H(p_i)$ and the next location point p' that should be confirmed by the annotators is same as Formula (11) and (9), respectively. Since M' is recalculated in each iteration via Formula (12), the dynamic confidence-based strategy has a time complexity of $O(n \times r)$, where $n = |T|$ and $r = |E|$. Please note that the efficiency of the dynamic confidence-based strategy could be increased by setting $r <= |E|$ with a ranking of all $e_{i,j} \in E$. The details are shown in Section 6.2.

5.4. Stability-based strategy

As introduced in the dynamic confidence-based strategy, the confidence probability of some location point may be influenced by the matchings of other locations points. For example in Fig. 7, p_1 is more confident to be matched to e_1 if one of p_2, p_3 and p_4 is also matched to e_1. However, in some specific cases such like p_5, it is not only confident to be matched to e_3 if p_7 is also matched to e_3, but also confident to be matched to e_4 if p_6 is also matched to e_4. This phenomenon is called *unstable* for a location point towards the matchings of other location points.

Formally, we define the *influence* of two location points $p_a, p_b \in T$ as $p_a \prec p_b$ if and only if $\langle p_a, e_a\rangle \notin M_b$. M_b is the set of matches by querying $Q(T_b, G)$ where

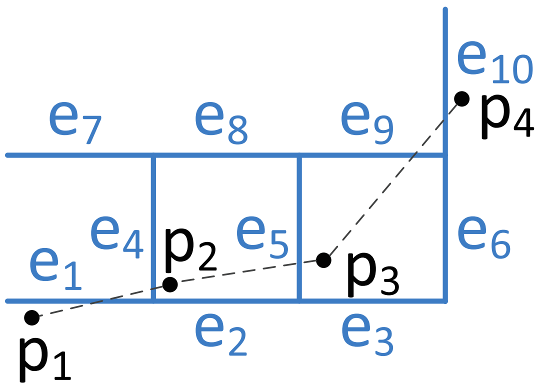


Fig. 5. Example of the scenario considered by confidence-based strategy.

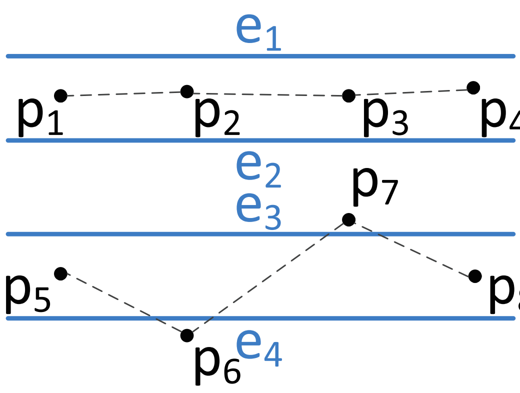


Fig. 7. Example of the scenario considered by stability-based strategy.

$T_b = (p_1, p_2, \ldots, p_{b-1}, p_{b+1}, \ldots, p_n)$ is the trajectory omitting p_b. Then, the uncertainty, i.e., the stability is defined as:

$$H(p_i) = \sum_{\forall p_j \in T, p_i \neg \prec p_j} 1 \tag{13}$$

Hence, given a trajectory T, the next location point p' that should be confirmed by the annotators is defined as:

$$p' = arg\,min_{p_i \in T} H(p_i) \tag{14}$$

Since the stability of a location point is calculated using all the other locations points in each iteration, $O(n^2)$ is the time complexity of the stability-based strategy.

6. Evaluation

In this section, we evaluate the performance of iMatching. We conduct extensive experimental studies to show the effectiveness, efficiency, and scalability of the interactive map-matching algorithm introduced in Section 4, and the query selection strategies introduced in Section 5.

6.1. Experiment data

In this paper, all trajectory data will be mapped to a road network obtained from a transportation department in China. The road network data is composed of 36,451 road segments and 25,613 intersections [28]. In order to clearly show the performance of iMatching under different situations of traffic, we employ both synthetic data generated through a trajectory generator, and real-world trajectory data provided by the government [29–32].

The synthetic trajectory generator consists of the following configurations:

1. n_p, the number of location points of the trajectory;
2. n_e, the number of road segments that the moving object travels along; and
3. δ, the measurement noise represented by the generated standard deviation.

The synthetic trajectory generator works as follows. First, it randomly chooses an intersection v_s and generates a random n_e-hop shortest path $P = (e_1, e_2, \ldots, e_{n_e})$ starting from v_s. Then, it randomly selects a starting point p_1 and an ending point p_{n_p} from e_1 and e_{n_e}, respectively. Next, it computes the driving distance $\text{route}(p_1, p_{n_p})$ along P and calculates the *distance interval* Δ between two consecutive location points:

$$\Delta = \frac{\text{route}(p_1, p_{n_p})}{n_p - 1} \tag{15}$$

Starting from p_1, the synthetic trajectory generator generates the location of next point along path P with $\text{route}(p_1, p_{i+1}) = \text{route}(p_1, p_i) + \Delta$. After all the points are generated, the generator adds a noise to each location point $p_i \in T$ based on Gaussian distribution with the standard deviation δ, and finally finish the desired trajectory $T = (\delta(p_1), \delta(p_2), \ldots, \delta(p_{n_p}))$.

After generating a synthetic trajectory, the generator launches the map-matching query $Q(T, G)$ and compares the result with P to find the *initial accuracy* of the trajectory, which stands for the percentage of correctly matched location points in T.

In the experiments, there are 11 groups of trajectories with at least 50% initial accuracy and consists of 50 location points each that are generated based on the following configurations:

1. *Initial Accuracy*: <60%, 60–70%, 70–80%, 80–90%, and ⩾90% with fixed sampling rate of 1.5 min and measurement noise of 101.04 meters.
2. *Sampling Rate*: 0.5 min, 1.5 min, and 4.5 min with a fixed initial accuracy of 70–80% and measurement noise of 11.23 m. Based on the sampling rate and our data analysis, we set the number of road segments to 10, 30, and 90, respectively.
3. *Measurement Noise*: 11.23 meters, 33.68 meters, and 101.04 meters with a fixed initial accuracy of 60–70% and a sampling rate of 0.5 min, based on our analysis of the experimental data.

The real-world trajectory dataset is collected from 15,231 taxis with 154 million records for 26 days. We have manually map-matched 200 trajectories, and each trajectory contains 50 location points. The map-matching process is through iMatching and the entire annotation history is recorded.

6.2. Evaluation metrics

To evaluate the performance of iMatching on trajectory T, we introduce the following evaluation parameters: 1) $\zeta(T)$, the quantity of mistakenly map-matched location points by the map-matching algorithm; 2) $\eta(T)$, the number of uncertain matches posed to the human annotators to confirm; and 3) $\psi(T)$, the number of "effective" uncertain matches posed to the human annotators to confirm. By "effective", it indicates that the location point $p_i \in \langle p_i, e_j \rangle$ is indeed mistakenly map-matched and the correct match should be $\langle p_i, e'_j \rangle$.

In the experiments, we introduce the following evaluation metrics:

1. *Cost Ratio*: Since the main purpose of the query selection strategy is to reduce the number of queries posed to the annotators, the first metric is defined as the reviewing cost ratio, where a lower cost ratio presents less potion of queries posed to the annotators. Formally,

$$\texttt{CR} = \frac{\eta(T)}{|T|} \tag{16}$$

2. *Selection Accuracy*: Since the workflow of iMatching terminates when all the map-matching errors are corrected by the annotators. The query selection strategy aims to find out all the mismatched points as quickly as possible. Thus, the second metric is defined as the selection accuracy, where a higher selection accuracy indicates a higher rate of selecting mismatched points. Formally,

$$\texttt{SA} = \frac{\psi(T)}{\eta(T)} \tag{17}$$

3. *True Negative Rate*: Since the interactive map-matching algorithm captures the topological correlation between the points on a same trajectory, some of the points could be automatically corrected after the annotator corrects one of them. The annotation cost reduced by such operation is evaluated by the true negative rate, where a lower true negative rate presents more location points are automatically corrected by the interactive map-matching query. Formally,

$$\texttt{TNR} = \frac{\psi(T)}{\zeta(T)} \tag{18}$$

6.3. Evaluation results

The experiments employ all the four adaptive query selection trajectories as introduced in Section 5, including:

- *Stability-based strategy* (STAB);
- *Dynamic confidence-based strategy* (D-CONF);
- *Confidence-based strategy* (CONF); and
- *Distance-based strategy* (DIST).

For comparison, we also adopt two baseline strategies:

- *Sequential strategy* (SEQ): the location points are posed to the annotators in a sequential order; and
- *Random strategy* (RAND): the location points are posed to the annotators in a randomized order.

6.3.1. Performance on synthetic data

Fig. 8 demonstrates the evaluation results on the synthetic trajectory dataset. For CR and SA, the performance results of the proposed query selection strategies are much better than the baseline strategies. Specifically, STAB achieves the best performance among all strategies, and D-CONF outperforms the two global strategies DIST and CONF, since it improves CONF as well as DIST. In terms of TNR, the performance results of all the query selection strategies are almost the same. This indicates that our interactive map-matching algorithm is stable despite of query selection strategies, and can automatically correct mismatched points during each iteration effectively.

According to Fig. 8, by applying the proposed query selection strategies, CR is reduced up to 44%, SA is improved up to 24%, and TNR is automatically corrected up to 59% during human annotations. The results show that the proposed adaptive query selection strategies and the interactive map-matching algorithm significantly reduces the annotation cost.

Next, let us discuss the impact of the configuration parameters on the performance of the query selection strategies. First, both CR and SA drop, and TNR rises upon a higher initial accuracy. This is because if the number of correctly matched points increases, the chance of finding incorrectly matched points will drop, hence the number of points that are automatically corrected will decrease after one of the points being corrected, and the number of iterations that are required to complete the map-matching task will also decrease. As a result, iMatching can reduce $\eta(T)$ despite the initial accuracy of the map-matching task, and performs much better than the baseline methods. Second, both CR and TNR increase, and SA drops when measurement noise and sampling rate are high. However, comparing with the baselines query selection strategies, the proposed query selection strategies are less sensitive to measurement noise, but more sensitive to sampling rate. This is because a higher sampling rate could create a large gap between two consecutive road segments, but a higher measurement noise only affects the geometric distance between the correct road segment and the location point. Hence, the lack of topological information is more serious than the lack of geometric information, which makes the proposed query selection strategies more sensitive to sampling rate rather than measurement noise. Nevertheless, in conclusion, iMatching could generally outperform the baseline methods.

6.3.2. Performance on real-world data

Fig. 9 shows the evaluation results on the real-world trajectory dataset. In Fig. 9, the proposed query selection strategies clearly perform more efficient than the baseline query strategies with respect to CR and SA. More specifically, STAB achieves the best performance by reducing 29% of CR and improving 21% of SA comparing with the baseline query selection strategies. Furthermore, 12% of the mistakenly map-matched location points are fixed by the interactive map-matching algorithm automatically, even though the initial accuracy is already high.

In conclusion, the performance of iMatching on both synthetic and real-world trajectory datasets are similar, and they all outperform baseline methods. Hence, iMatching accomplishes a sufficient performance for all kinds of trajectories, and prominently reduces the annotation cost.

6.4. Scalability

In order to show the scalability of the proposed adaptive query selection strategies, we generate a new trajectory dataset with 8 groups of trajectories consist of 10–80 location points each. The configurations are set to: 60–70% for initial accuracy, 1.5 min for sampling rate, and 101.04 meters for measurement noise.

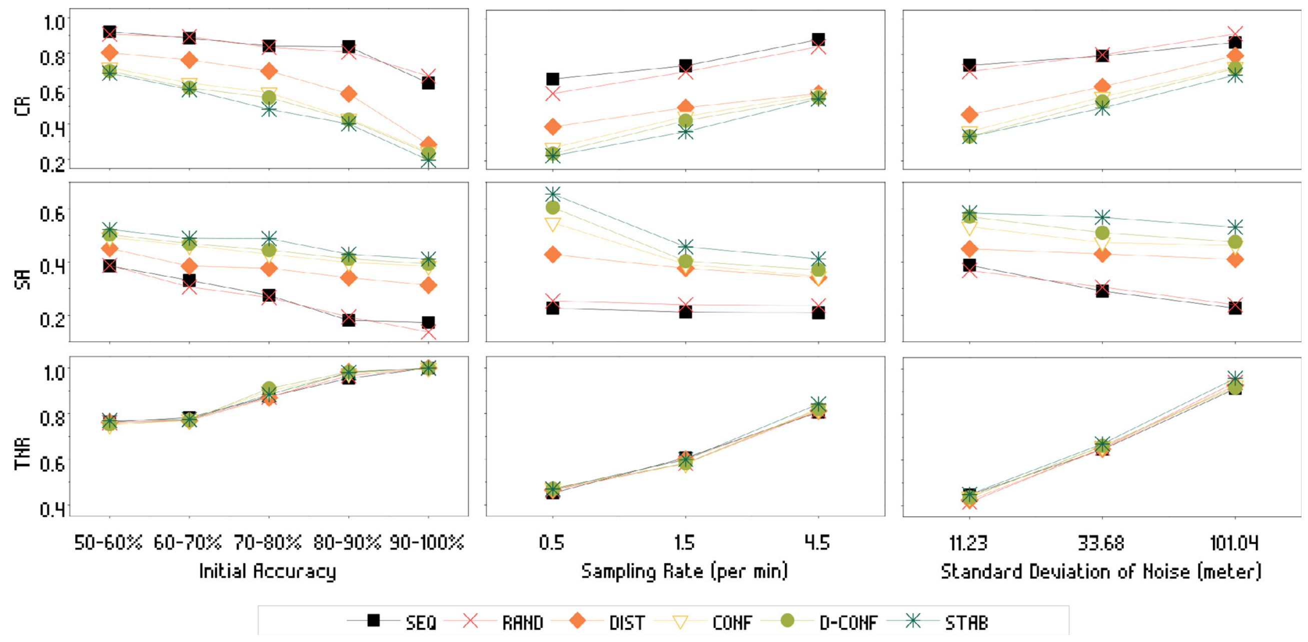


Fig. 8. Performance of iMatching on synthetic trajectory data.

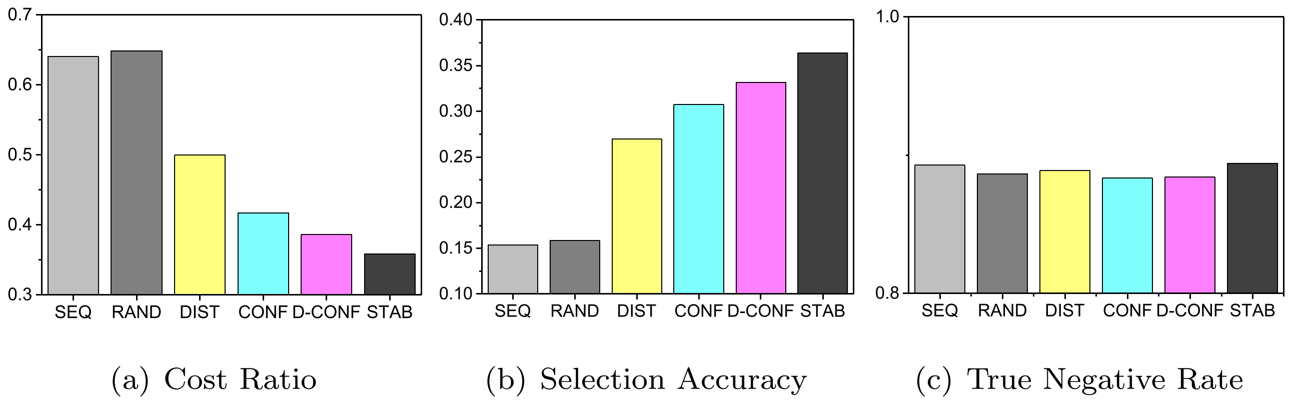
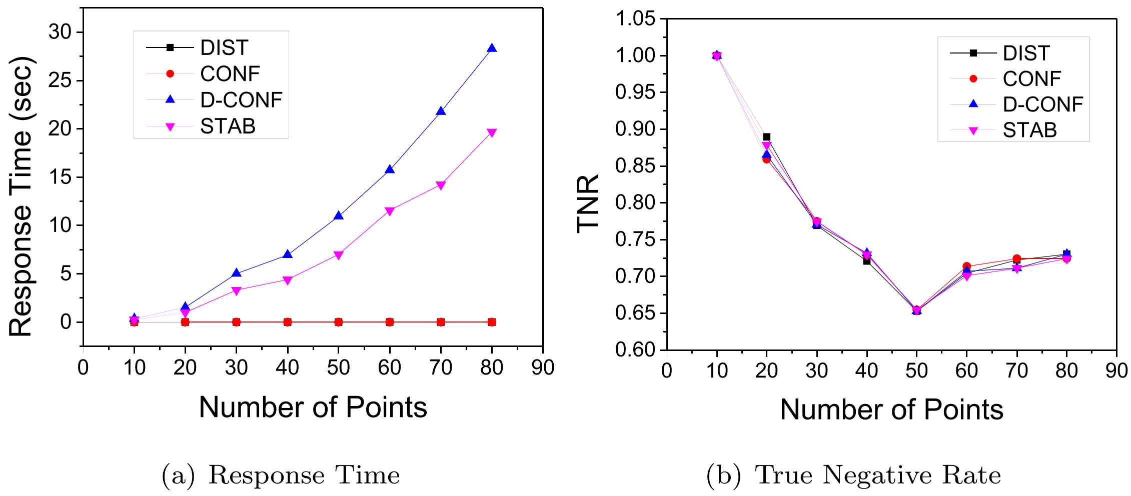


Fig. 9. Performance of iMatching on real-world trajectory data.

Fig. 10a shows the impact in terms of response time by the number of location points, and Fig. 10b shows that impact on TNR. Similar to the theoretical analysis in Section 5.3 and 5.4, the response time increases linearly upon the number of points for D-CONF and STAB. On the contrary, TNR drops before the number of points reaches 50, and rises after. This indicates that TNR performs best if the number of location points is around 50. Based on such observation, we try to break long trajectories into short sub-trajectories consists of 50 location points, and apply iMatching on them separately before joining the results together. The evaluation in terms of the average execution time of the above trajectory-breaking method is shown in Fig. 10c. In Fig. 10c, it is clear that breaking long trajectories into short ones could significantly reduce the task time, especially for D-CONF and STAB.



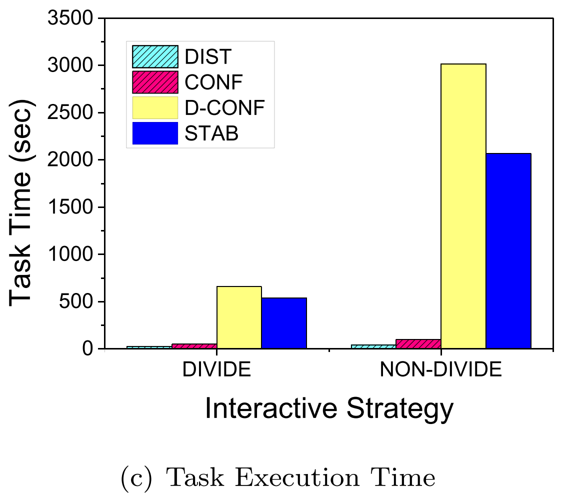


Fig. 10. Scalability of the adaptative query selection strategies.

7. Conclusion

In this paper, we propose an interactive map-matching system, called *iMatching*, which reduces the costs of manual map-matching annotations and maintains the high accuracy of a map-matching task. The system consists of a novel interactive map-matching algorithm and various adaptive query selection strategies. Through the experiments conducted on both synthetic and real-world trajectory datasets, iMatching is proven to be both effective and efficient. Since we only consider independent annotators for the interactive map-matching task, we plan to extend the interactive map-matching algorithm by introducing crowd-sourcing methods, in which multiple annotators are involved in a map-matching task.

CRediT authorship contribution statement

Ye Ding: Conceptualization, Data curation, Writing - review & editing. **Xibo Zhou:** Methodology, Software, Writing - original draft, Visualization. **Qing Liao:** Investigation, Project administration. **Haoyu Tan:** Validation, Formal analysis. **Qiong Luo:** Resources, Funding acquisition. **Lionel M. Ni:** Supervision.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work is supported in part by the National Key-Research and Development Program of China under Grant No. 2020YFB2104003 and the National Natural Science Foundation of China under Grant No. 61976051, U19A2067 and U1811463 and the Basic Research Project of Shenzhen under Grant No. JCYJ20180306174743727 and JCYJ20200109113427092.

Appendix A. Supplementary data

Supplementary data associated with this article can be found, in the online version, athttps://doi.org/10.1016/j.neucom.2020.04.155.

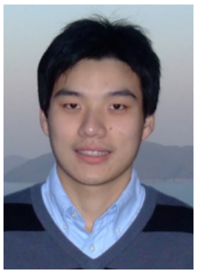
References

[1] X. Zhou, Y. Ding, H. Tan, Q. Luo, L.M. Ni, Himm: An hmm-based interactive map-matching system, in: International Conference on Database Systems for Advanced Applications, Springer, 2017, pp. 3–18.
[2] G. Jagadeesh, T. Srikanthan, X. Zhang, A map matching method for gps based real-time vehicle location, J. Navig. 57 (03) (2004) 429–440.
[3] B. Settles, Active learning literature survey, Univ. Wisconsin, Madison 52 (55–66) (2010) 11.
[4] Q. Liao, Q. Zhang, Local coordinate based graph-regularized nmf for image representation, Signal Process. 124 (2016) 103–114.
[5] M. Yu, W. Chen, Z. Li, Y. Chen, Improvement on integrity and reliability of vehicle positioning by a new map matching method, in: Proceedings of the 17th International Technical Meeting of the Satellite Division of the Institute of Navigation, ION GNSS, 2004, pp. 2086–2094..
[6] W. Xia, H. Jiang, D. Feng, Y. Hua, Similarity and locality based indexing for high performance data deduplication, IEEE Trans. Comput. 64 (4) (2015) 1162–1176.
[7] F.C. Pereira, H. Costa, N.M. Pereira, An off-line map-matching algorithm for incomplete map databases, Eur. Transp. Res. Rev. 1 (3) (2009) 107–124.
[8] J. Yuan, Y. Zheng, C. Zhang, X. Xie, G.-Z. Sun, An interactive-voting based map matching algorithm, in: 2010 Eleventh International Conference on Mobile Data Management, IEEE, 2010, pp. 43–52.
[9] P. Newson, J. Krumm, Hidden markov map matching through noise and sparseness, in: Proceedings of SIGSPATIAL, ACM, 2009, pp. 336–343.
[10] G. Wang, R. Zimmermann, Eddy: an error-bounded delay-bounded real-time map matching algorithm using hmm and online viterbi decoder, in: Proceedings of SIGSPATIAL, ACM, 2014, pp. 33–42.
[11] L. Liao, D.J. Patterson, D. Fox, H. Kautz, Learning and inferring transportation routines, Artif. Intell. 171 (5) (2007) 311–331.
[12] O. Pink, B. Hummel, A statistical approach to map matching using road network geometry, topology and vehicular motion constraints, in: Proceedings of ITSC, IEEE, 2008, pp. 862–867.
[13] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, Y. Huang, Map-matching for low-sampling-rate gps trajectories, in: Proceedings of SIGSPATIAL, ACM, 2009, pp. 352–361..
[14] L. Zhang, K. Li, Y. Xu, J. Mei, F. Zhang, K. Li, Maximizing reliability with energy conservation for parallel task scheduling in a heterogeneous cluster, Inf. Sci. 319 (2015) 113–131.
[15] X. Zhou, K. Li, G. Xiao, Y. Zhou, K. Li, Top k) favorite probabilistic products queries, IEEE Trans. Knowl. Data Eng. 28 (10) (2016) 2808–2821.
[16] C. Chen, K. Li, A. Ouyang, Z. Zeng, K. Li, Gflink: an in-memory computing architecture on heterogeneous cpu-gpu clusters for big data, IEEE Trans. Parallel Distrib. Syst. 29 (6) (2018) 1275–1288.
[17] C. Chen, K. Li, A. Ouyang, K. Li, Flinkcl: an opencl-based in-memory computing architecture on heterogeneous cpu-gpu clusters for big data, IEEE Trans. Comput. 67 (12) (2018) 1765–1779.
[18] C. Chen, K. Li, A. Ouyang, Z. Tang, K. Li, Gpu-accelerated parallel hierarchical extreme learning machine on flink for big data, IEEE Trans. Syst., Man, Cybern. Syst. 47 (10) (2017) 2740–2753.
[19] J. Chen, K. Li, K. Bilal, K. Li, S.Y. Philip, et al., A bi-layered parallel training architecture for large-scale convolutional neural networks, IEEE Trans. Parallel Distrib. Syst. 30 (5) (2018) 965–976.
[20] S. Chakraborty, V. Balasubramanian, A.R. Sankar, S. Panchanathan, J. Ye, Batchrank: a novel batch mode active learning framework for hierarchical classification, in: Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2015, pp. 99–108.
[21] M. Fang, J. Yin, D. Tao, Active learning for crowdsourcing using knowledge transfer, AAAI (2014) 1809–1815.
[22] B. Settles, M. Craven, An analysis of active learning strategies for sequence labeling tasks, in: Proceedings of Empirical Methods in Natural Language Processing, Association for Computational Linguistics, 2008, pp. 1070–1079..
[23] J. Zheng, Z. Jiang, R. Chellappa, J.P. Phillips, Submodular attribute selection for action recognition in video, Adv. Neural Inf. Process. Syst. (2014) 1341–1349.
[24] M. Fang, J. Yin, X. Zhu, Active exploration: simultaneous sampling and labeling for large graphs, in: Proceedings of the 22nd ACM international conference on Conference on information & knowledge management, ACM, 2013, pp. 829–834..
[25] A. Krishnamurthy, A. Agarwal, T.-K. Huang, H. Daumé III, J. Langford, Active learning for cost-sensitive classification, J. Mach. Learn. Res. 20 (65) (2019) 1–50.
[26] Y. Ding, S. Liu, J. Pu, L.M. Ni, Hunts: a trajectory recommendation system for effective and efficient hunting of taxi passengers, in: Proceedings of MDM, vol. 1, IEEE, 2013, pp. 107–116..
[27] L.R. Rabiner, A tutorial on hidden markov models and selected applications in speech recognition, Proc. IEEE 77 (2) (1989) 257–286.
[28] Q. Liao, H. Tan, W. Luo, Y. Ding, Diverse mobile system for location-based mobile data, Wireless Commun. Mobile Comput. (2018).
[29] Y. Ding, J. Zheng, H. Tan, W. Luo, L.M. Ni, Inferring road type in crowdsourced map services, in: Database Systems for Advanced Applications, Springer, 2014, pp. 392–406.
[30] Y. Ding, Y. Li, K. Deng, H. Tan, M. Yuan, L.M. Ni, Dissecting regional weather-traffic sensitivity throughout a city, in: 2015 IEEE International Conference on Data Mining, IEEE, 2015, pp. 739–744.
[31] Y. Ding, Y. Li, K. Deng, H. Tan, M. Yuan, L.M. Ni, Detecting and analyzing urban regions with high impact of weather change on transport, IEEE Trans. Big Data 3 (2) (2017) 126–139.
[32] X. Zhou, Y. Ding, F. Peng, Q. Luo, L.M. Ni, Detecting unmetered taxi rides from trajectory data, in: 2017 IEEE International Conference on Big Data (Big Data), IEEE, 2017, pp. 530–535.

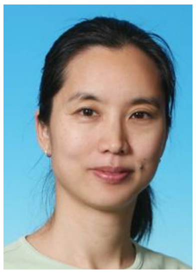
Ye Ding received his Ph.D. degree in 2014 supervised by Prof. Lionel M. Ni from the Department of Computer Science and Engineering of the Hong Kong University of Science and Technology. He is currently an Associate Professor in School of Cyberspace Security at Dongguan University of Technology. His research interests are spatial-temporal data analytics, big data, and machine learning. He is a member of IEEE since 2013.

Xibo Zhou recieved his Ph.D. degree in 2018 supervised by Prof. Lionel M. Ni and Prof. Qiong Luo from the Department of Computer Science and Engineering of the Hong Kong University of Science and Technology. His research interests include spatial-temporal data mining and distributed systems.

Qing Liao received her Ph.D. degree in 2016 supervised by Prof. Qian Zhang from the Department of Computer Science and Engineering of the Hong Kong University of Science and Technology. She is currently an Associate Professor with School of Computer Science and Technology, Harbin Institute of Technology (Shenzhen), China. Her research interests include big data analytics and artificial intelligence.



Haoyu Tan received his Ph.D. degree from the Hong Kong University of Science and Technology in 2013. He has worked as a Research Assistant Professor in Guangzhou HKUST Fok Ying Tung Research Institute since 2013. His research interests include big data management and processing, machine learning, large-scale data mining, human behaviour analysis, urban computing and recommender systems.



Qiong Luo is an Associate Professor at the Department of Computer Science and Engineering, Hong Kong University of Science and Technology (HKUST). Her research interests are in big data systems, parallel and distributed systems, and scientific computing. Current focus is on data management on modern hardware, GPU acceleration for data analytics, and database support for e-science. Qiong received her Ph.D. in Computer Sciences from the University of Wisconsin-Madison in 2002, her M.S. and B.S. in Computer Sciences from Beijing (Peking) University, China in 1997 and 1992 respectively.

Lionel M. Ni is the Provost of the Hong Kong University of Science and Technology and the Chair Professor of the Department of Computer Science and Engineering. As a fellow of IEEE and Hong Kong Academy of Engineering Science, Dr. Ni has chaired over 30 professional conferences and has received eight awards for authoring outstanding papers. He serves on the editorial boards of Communications of the ACM, IEEE Transactions on Big Data and ACM Transactions on Sensor Networks.