Received July 29, 2018, accepted August 30, 2018, date of publication September 13, 2018, date of current version October 8, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2869852

A Green Self-Adaptive Approach for Online Map Matching

QI AN[1,2], ZHIYONG FENG[1,3], (Member, IEEE), SHIZHAN CHEN[1,2], (Member, IEEE), AND KEMAN HUANG[4], (Member, IEEE)

[1]Tianjin Key Laboratory of Cognitive Computing and Application, Tianjin 300350, China
[2]School of Computer Science and Technology, Tianjin University, Tianjin 300350, China
[3]School of Computer Software, Tianjin University, Tianjin 300350, China
[4]Sloan School of Management, Massachusetts Institute of Technology, Cambridge, MA 02139, USA

Corresponding author: Keman Huang (keman@mit.edu)

This work was supported in part by the National Natural Science Foundation of China under Grants 61572350, 61502333, and 61672377, and in part by the National Key R&D Program of China under Grant 2017YFB1401201.

ABSTRACT GPS trajectory data plays a critical role for the intelligent transportation. Due to many factors like measure error, sampling error and battery power-saving requirement, the directly obtained trajectories will inaccurately align to the digital map. This requires an online map matching algorithm with high precision, low latency, and energy consumption saving. However, the existing approaches need to make some trade off among these criterions. Hence, based on the hidden Markov model, this paper proposes an adaptive online map matching algorithm to improve the performances in these perspectives at the same time: 1) the probabilistic method integrating the geometric information and topological information is developed to improve the accuracy; 2) the adaptive sampling frequency method is proposed to reduce the energy consumption; and 3) the adaptive sliding window method is presented to reduce the output delay. The experiments demonstrate that our approaches can not only improve the matching precision, but also reduce the latency and energy consumption simultaneously.

INDEX TERMS Online map matching algorithm, hidden Markov model, adaptive sampling, adaptive sliding window.

I. INTRODUCTION

In recent years, with the development of GPS-based positioning devices [34], a large number of GPS trajectory data can be real-time obtained from the taxi, smartphones and other GPS-based intelligent devices. Using these trajectory data, online path planning, traffic incident detection, and travel time prediction can be easily implemented to provide location-based services.

However, due to the GPS data error and the energy limitation [2], the coordinate obtained from the GPS-based intelligent devices, especially for the mobile devices, cannot be used directly. The limitation of the GPS technology itself, including the GPS data measured, transmit and accept, will cause the measure error. Meanwhile, the availability of the data storage and transmission bandwidth inevitably constrain the GPS trajectory sampling interval, resulting into the sampling error. In addition, nowadays the location-based services based on the online map matching are typically installed on mobile where the high energy consumption is a critical issue which cannot be ignored.

Many solutions have been proposed to solve the online map matching algorithms over these years. Geometric-based [24] map matching algorithm is simple to implement, but it only considers the geometric information of the road network regardless of the connection information between the road networks. Topological-based [3], [4] map matching algorithm improves the Geometric-based algorithm by making full use of the road network connectivity and continuity. Probabilistic-based [5] map matching algorithm further considers the probability of the GPS point being selected. Recently, some advanced algorithms using Kalman filtering [6], [7], Hidden Markov Models [8], [9], [33] or machine learning [11], [26] are proposed to further improve the performance.

To reduce the battery power consumption, some solutions [16], [17] using low sample rate trajectories are presented for the off-line scenarios. However, for the online scenarios, due to the loss of the valid information be offer, this results into a significant reduction in the matching accuracy. The straightforward solution here is to increase the sliding window size [8], [15], which insteadly cause significant output delay.

Hence, it has become critical and challenged to simultaneously improve these three metrics, including high precision, low energy consumption, and low output delay. To achieve this goal, this paper proposes an Adaptive Online Map Matching algorithm based on HMM, named AOMM:

A. MULTI-INFORMATION INTEGRATION

To improve the accuracy, AOMM integrates the information including the geometric information of the space road network, the connectivity and continuity information of road network, and the basic idea of probability matching.

B. ADAPTIVE SAMPLING FREQUENCY

In the most of existing map matching algorithms [12], [16], the GPS data sequences are sampled with the fixed frequency without considering the road condition. For some road segments, the fixed sampling rate could be too low, so that the effective information in the sliding window is too little, leading to lower matching accuracy. On the contrary, for some other road segments, the fixed sampling rate is too high which does not provide more effective information to increase the accuracy but only lead to power consumption. Hence, in this paper, the adaptive sampling frequency method which enables different sampling frequencies in different roads can not only guarantee the matching accuracy but also reduce the energy consumption.

C. ADAPTIVE SLIDING WINDOW

The sliding window method is widely adopted in the online map matching scenarios. Most existing online map matching solutions employ the fixed sliding window sizes, which makes it difficult to achieve the high accuracy and low latency simultaneously. This is because a too large window size will cause high output delay while the too small window size cannot offer enough information to guarantee the accuracy. Therefore, in this paper, we develop an adaptive sliding window method which can balance the accuracy and the output delay by adjusting the window size adaptively according to the real-time road conditions.

To evaluate the effectiveness of our approach, we use the real dataset provided by CAR Inc[1] in the November 2015 Beijing. The result shows that comparing with the state-of-the-arts, our approach always achieves a higher average accuracy, reaching 96.41%. More importantly, it achieves a 46.51% reduction in map matching time, 19.05% reduction in window size, and 2/3 decreasing of the energy consumption at the same time.

The rest of this paper is organized as follows. Section II discusses the related works. Section III shows some preliminary definitions and baseline model. Section IV details our algorithms. Section V reports the experiments and the results. Section VI concludes this paper.

[1]www.zuche.com.

II. RELATED WORK

Depending on how the trajectory data and road network information are used, the map matching algorithms are divided into four types: geometric-based, topological-based, probabilistic-based and other advanced algorithms. The geometric map-matching algorithm is a basic matching algorithm. A pioneering work on map-matching by considering the geometric information of the space road network was published by Greenfeld and Joshua [24] in 2002. This algorithm is easy to implement. However, since it does not consider the connection information between the road networks, it has poor matching accuracy. From a different perspective, topology map-matching algorithm [4], [13] makes full use of connectivity and continuity of the road network, and significantly improves the map matching accuracy. However, the accuracy is still not efficiency enough that they can only be used in the scenarios which don't require a high accuracy. Furthermore, to deal with the intersections or complex road segments, the probabilistic map matching algorithm [5] is developed and significantly improves the matching accuracy. More recently, some advanced map matching algorithm using Kalman filtering [6], [7], fuzzy logic model [10], Hidden Markov Models (HMM) [8], [9], [33] etc, are developed to deal with complex road conditions. Typically, the HMM [9] is used to identify the optimize matching path when considering the noisy and sparse location data.

Depending on the range of trajectories used, the existing map matching algorithms can be divided into two types: the local/incremental [14], [15] map matching algorithm, and the global [16], [17] map matching algorithm. The local/incremental algorithm is a greedy algorithm, which only determines the matching results of current trajectory point, and the next point starts from the determined matching point. The local/incremental algorithm has good performance in accuracy when the sampling frequency is very high (e.g., 2-5seconds). As the sampling rate decreases, the arc-skipping problem [12] referring to the scenarios that the vehicle jumps directly from one road segment to another while it is still far away from the intersection, becomes serious, and causes a significant decrease in accuracy. Therefore, the local/incremental algorithm will have a relative high energy consumption due to the high sampling frequency. In this situation, the EnAcq algorithm [32] is proposed when the researcher hopes to propose an algorithm that can reduce energy consumption while ensuring accuracy. The idea of adaptive sampling is first proposed in EnAcq, it can be described as dividing the vehicle speed into three fixed thresholds, and assigning different sampling frequencies to different thresholds. However, the sampling process is not fully adaptive, there is room for improvement. A global map-matching algorithm tries to find the curve in the road network that is as close as possible to the vehicle trajectory. Fréchet distance [18] and Weak Fréchet distance [19] are widely adapted to measure the similarity between the vehicle trajectory and road. Intuitively, the global map matching algorithm requires all trajectories as input data and is only suitable

TABLE 1. Performance comparison.

Num	Algorithm	Type	sampling	Ar(%)[2]
1	ST-matching [12]	global (geometric/topological), FSF	0.5min	90.73
			1.5min	88.27
			3.5min	84.98
2	HMM [15]	local/incremental (HMM), FSF + FSW	0.5min	91.92
			1.5min	89.97
			3.5min	82.42
3	IF-matching [13]	global (geometric/topological), FSF	0.5min	94.21
			1.5min	93.27
			3.5min	91.67
4	HMM+RCM [1]	local/incremental (HMM+RCM), FSF + ASW	0.5min	94.14
			1.5min	93.11
			3.5min	90.12
5	AntMapper [26]	global (geometric/topological), FSF	0.5min	95.29
			1.5min	94.16
			3.5min	91.91
6	EnAcq [32]	local/incremental (HMM), ASF + ASW	-	94.43

for the off-line matching scenarios. Additionally, if only the local trajectories are provided, the matching accuracy will be significantly reduced as there is not enough valid information. To solve this problem, a synthesis algorithm named AntMapper-matching [26] is proposed in 2017. It not only considers local geometric/topological information, but also use a global Ant colony algorithm to find the shortest path. Experiments show that the AntMapper-matching algorithm can provide accurate matching results within a relatively short running time.

As summarized in Table 1, most of these algorithms use the fixed sampling frequency method (or simply 'FSF'). We can see that when the sampling interval increases, the accuracy of the above algorithms decreases obviously, as there is not enough effective information to ensure the accuracy. If the sampling interval is too small, a lot of similar data will lead to a waste of resources, including sampling and transmission. In contrast, the accuracy of the EnAcq algorithm [32] using the adaptive sampling frequency strategy (or simply 'ASF') is stable. In addition, ST-matching, IF-matching, and AntMapper are all global algorithms which are used in the off-line scenarios, and they do not use a sliding window strategy. To achieve the online process, HMM uses a fixed sliding window strategy(or simply 'FSW'). The size of the window has an effect on the accuracy and the output delay. If the window size is too small, it may not find the best match sequence. And if the size is too large, the output delay will be raised. Based on this, the adaptive sliding window strategy

[2]Please refer to Section V to check the detail about the experiment to calculate the accuracy for each algorithm. The dataset used in the comparison algorithms is the same as that in our algorithm. And our paper completely re-implement the comparison algorithms.

TABLE 2. Metadata of GPS LOG.

	$TP_i.lng$	$TP_i.lat$	$TP_i.v$	$TP_i.t$
TP_1	116.325693	39.891172	47.52	2015-10-16 17:59:32
TP_2	116.329658	39.891346	48.39	2015-10-16 18:03:20
		...		
TP_n	116.347566	39.862854	44.67	2015-10-16 18:16:32

(or simply 'ASW') [15] is proposed to solve the above problem, and it has been applied to some map matching algorithms [1]. However, there is almost no algorithm can make the map matching satisfy the high accuracy, low sampling rate and low output delay simultaneously well.

III. PRELIMINARIES

In this section, the basic concepts and models involved in the AOMM-Matching algorithms will be discussed.

A. PROBLEM FORMULATION

Definition 1 (Trajectory Point, TP): Each *TP* consists of a series attributes, including id, longitude, latitude, speed, and timestamps. It can be abbreviated as a tuple represented by $< id, lng, lat, v, t >$, as illustrated in Table 2. For example, the longitude of trajectory point TP_1 is 116.325693, the latitude is 39.891172, and the speed is 47.52km/h. TP_1 was sampled at 2015-10-16 17:59:32.

Definition 2 (GPS Trajectory, T): A GPS Trajectory T is a sequence of *TP*. The time interval between any consecutive *TP* not exceeding a certain threshold ΔT, i.e. $T : TP_1 \rightarrow TP_2 \rightarrow \ldots \rightarrow TP_n$, where $0 < TP_{i+1}.t - TP_i.t < \Delta T$ $(1 \leq i < n)$.

Definition 3 (Road Segment, RS): A road segment *RS* is a directed edge that is associated with an id $RS.id$, a typical travel speed $RS.v$, a starting point $RS.start$, and an ending point $RS.end$.

Definition 4 (Road Network, RN): The Road Network is a directed graph $RN < V, E >$. $RN.V$ is a set of vertices, each of which represents starting point, endpoint or intersection point of *RS*. $RN.E$ is a set of edges, each of which represents a road segment.

Definition 5 (Candidate Road Segment, CRS): *CRS*s of each *TP* can be obtained by traversing the road network. The *CRS* should be satisfied that the distance between *TP* and *CRS* is less than error radius r. These *CRS*s are called Candidate Road Segments corresponding to *TP*.

Definition 6 (Candidate Point, CP): The projection point of the *TP* on its corresponding candidate road segment is called the Candidate Point.

B. BASELINE MODEL

Hidden Markov Model (HMM) [35] is a statistical model used to describe a Markov process with hidden parameters.

Figure 1 displays an ideal hidden markov process in the map matching scenario. The sequence $\{CRS_{i-1}, CRS_i, \ldots, CRS_n\}$ represents the hidden states, corresponding to the

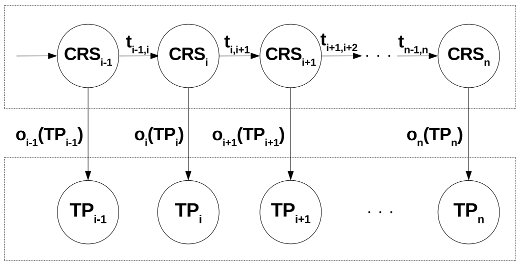

FIGURE 1. An ideal Hidden Markov process.

Candidate Road Segments in the map matching. The sequence $\{TP_{i-1}, TP_i, \ldots, TP_n\}$ represents the observable states, corresponding to the Trajectory Points in the map matching. The $t_{i-1,i}$ represents the transition probability between CRS_{i-1} and CRS_i, corresponding to the transition probability between the Candidate Road Segments. Obviously, the higher the possibility of moving the TP from CRS_{i-1} to CRS_i is, the closer the transition probability between them is to 1. If the two CRSs are parallel, the transition probability between them is 0. $o_i\,(TP_i)$ represents the emission probability between CRS_i and TP_i, corresponding to the probability of the observed trajectory point is on the candidate road segment.

The HMM has been proved effective for the map matching problem. Therefore, in this paper, we also choose the HMM as the basic model of our algorithm.

IV. AOMM: ADAPTIVE MAP MATCHING ALGORITHM

A. ALGORITHM FRAMEWORK

The framework of the AOMM algorithm is illustrated in Figure 2, which consists of three parts: 1) candidate preparation, 2) probability analysis and 3) online map matching.

The implementation of the AOMM can be summarized as follows: First, the trajectory-based application sends a request to the AOMM System, including the location, speed and timestamp of the current vehicle. The criteria for this request should satisfy the sampling specification of the adaptive sampling frequency method. Then, **Candidate Preparation Module** uses GPS trajectory, the geometric information of road networks, and the topological information of road networks to obtain candidate sets. The candidate sets include candidate points and candidate road segments. Next, **Probability Analysis Module** uses candidate sets information to calculate transition probability and emission probability. After that, **Online Map Matching Module** uses the result of probabilistic analysis as input to the online Viterbi algorithm. It should be noted that in the online Viterbi algorithm, we will use the adaptive sliding window algorithm to minimize the output delay and improve the matching accuracy of the algorithm. Finally, the matching result will be returned to the trajectory-based application.

The Viterbi algorithm [27], [28] is based on the known observation sequence and the state transition probability, and obtains the most probable hidden state sequence

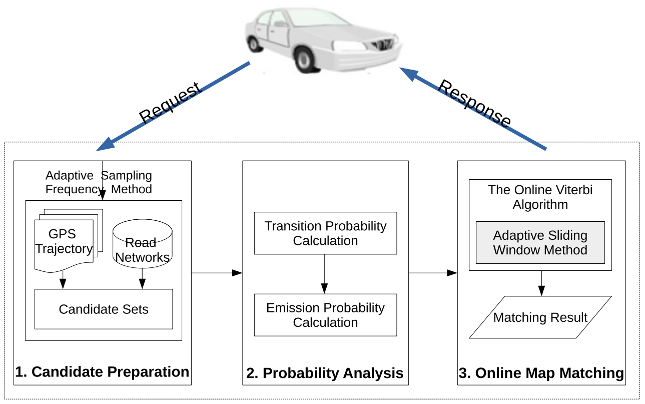
Request
Response
Adaptive Sampling Frequency Method
GPS Trajectory
Road Networks
Candidate Sets
1. Candidate Preparation
Transition Probability Calculation
Emission Probability Calculation
2. Probability Analysis
The Online Viterbi Algorithm
Adaptive Sliding Window Method
Matching Result
3. Online Map Matching

FIGURE 2. The framework of the AOMM algorithm.

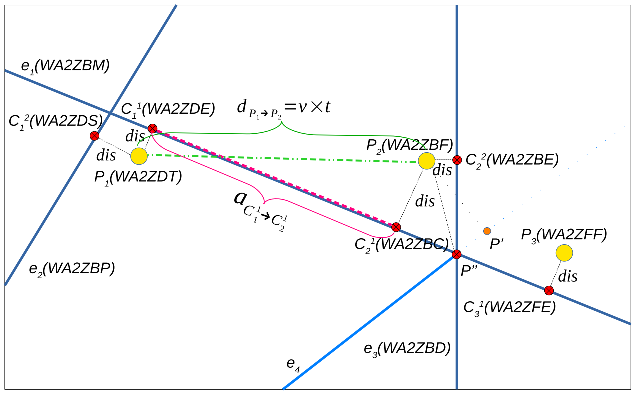

FIGURE 3. An illustration for getting matching result.

by backstepping. It is essentially an HMM decoding problem. It adopts the idea of dynamic programming, and recursively calculates the most probable (local optimal) path in the current candidate pathes by using the backward pointer. Through the online Viterbi algorithm, each GPS trajectory can be matched to its proper road segment.

B. CANDIDATE PREPARATION

The goal of Candidate Preparation is to get the candidate sets for each trajectory point. The candidate sets include candidate segments and candidate points for each TP. To achieve this goal, this paper uses the GeoHash [20], [21] algorithm to convert the coordinates into a string by using the dichotomous infinite approximation in the latitude and longitude interval. The more the number of the same prefixes in the string, the nearer the distance between two GPS points will be. As shown in Figure 3, using the GeoHash algorithm, we get the GeoHash string for the trajectory point P_1, P_2, and P_3 as WA2ZDT, WA2ZBF, WA2ZFF, respectively. And the GeoHash strings of road segments e_1, e_2, and e_3 are WA2ZBM, WA2ZBP, WA2ZBD, respectively.

Secondly, for each trajectory point, this paper will traverse all the road networks to find the corresponding candidate roads. These candidate roads should satisfy that their corresponding GeoHash strings have the same prefix as the GeoHash string of the trajectory point. In this paper, the number of the same prefixes in the string is defined as r, which is also called error range. Take the trajectory point P_2 in Figure 3 as an example, when r is set as 5, e_1 and e_3 can be considered as the candidate road segment for P_2. This is because there are 5 consecutive same characters WA2ZB between them. Similarly, the candidate road segments of the P_1 are e_1 and e_2, and the candidate road segment of the P_3 is e_1. Note that, we may get too many candidate road segments, which will lead to the increase of computational complexity. To avoid this, we will predefine the maximum number of candidate road segments as k for each trajectory point. The selection criteria of k is the shortest distance [12] between trajectory point and candidate road segments. In Figure 3, k is set as 2.

Finally, we calculate the candidate point on each candidate segment corresponding to each *TP*. In this paper, the candidate point is defined as the projection point of the *TP* on the candidate segment. If the projection point does not exist, the endpoint of the candidate segment closest to the trajectory point is defined as the candidate point. Taking trajectory point P_2 in Figure 3 as an example, the projection point of P_2 on the candidate road segment e_1 is C_2^1, so C_2^1 is a candidate point of P_2 on the road segment e_1. C_2^2 is a candidate point of P_2 on the road segment e_3. Since the projection point P' of P_2 on the line e_4 is not on the road segment e_4, the endpoint P'' of the candidate segment e_4 closest to the trajectory point P_2 is defined as the candidate point. The candidate points of P_1 are C_1^1 and C_1^2, and the candidate point of P_3 is C_3^1.

C. PROBABILITY ANALYSIS

In this subsection, we will detail the probability analysis for candidate sets. As shown in Figure 2, the probabilistic analysis includes the transition probability calculation and the emission probability calculation.

1) TRANSITION PROBABILITY

This component will calculate the transition probability by using the topological and speed information of the road network. Specifically, the topological information is used to describe distance similarity between GPS trajectory and road segments, the speed information is used to calculate speed similarity between GPS trajectory and road segments.

a: DISTANCE SIMILARITY FUNCTION (DSF)

The *DSF* function calculates the distance similarity probability between the real distance and the matching distance, as shown in formula 1. $d_{i\to i+1}$ is the real driving distance between two adjacent trajectory points, and $a_{i\to i+1}$ is the shortest distance between the corresponding candidate point of these two adjacent trajectory points. Take the trajectory point P_1 and P_2 in Figure 3 as an example. $d_{P_1\to P_2}$ represents the real driving distance from P_1 to P_2. $a_{C_1^1\to C_2^1}$ represents the shortest path length from any candidate point of $P_1(C_1^1, C_1^2)$ to any candidate of $P_2(C_2^1, C_2^2)$. There are four combinations of collocation, $C_1^1 \to C_2^1$, $C_1^1 \to C_2^2$, $C_1^2 \to C_2^1$ and $C_1^2 \to C_2^2$. The more similar the two distances is, the greater the probability is that the vehicle will drive between these two candidate points.

$$DSF\ (TP_{i\to i+1}) = 1 - \frac{|d_{i\to i+1} - a_{i\to i+1}|}{a_{i\to i+1}} \tag{1}$$

In the above formula, $d_{i\to i+1}$ can be calculated by Equation (2). ΔT is represented as a time interval and $\bar{v}$ is the average speed of a vehicle. $a_{i\to i+1}$ can be calculated by A-Star [29] algorithm, which finds the shortest path in a static road network. A-Star algorithm makes the routing results achieve the best compromise between speed and accuracy, compared with the traditional *DFS* algorithm, *BFS* algorithm [30], and *Dijkstra* algorithm [31]. When $d_{i\to i+1}$ and $a_{i\to i+1}$ is more similar, $|d_{i\to i+1} - a_{i\to i+1}|$ value is more close to 0, the value of *DFS* is more close to 1. On the contrary, when the difference between $d_{i\to i+1}$ and $a_{i\to i+1}$ is large, the two candidate points cannot reflect the real situation, and the matching results may not be correct.

$$d_{i\to i+1} = \Delta T \cdot \bar{v} \tag{2}$$

b: SPEED SIMILARITY FUNCTION (SSF)

The *SSF* function calculates the speed similarity probability between the road segment speed limit and the average speed of the adjacent trajectory points. In this paper, the cosine value [22] of the above two speeds is used to describe the speed similarity probability, as shown in formula (3). $r_j.v$ is the speed limit of the road segment. $\bar{v}_{(i\to i+1)}$ is the average speed from TP_i to TP_{i+1}. The range of j is from 1 to q, indicating the number of road segments passing from TP_i to TP_{i+1}. When the above two speed are more similar, the cosine value is more close to 1, the value of *SSF* is more close to 1. On the contrary, if the similarity is smaller, the value of *SSF* is more close to 0. However, the above simple cosine formula values will tends to favor local streets with low free flow speeds instead of major roads with high free flow speeds, when traffic is congested with low average speed. To solve this problem, the congestion factor α is proposed. Because the vehicle speed can reflect the vehicle driving condition[3] to a great extent, we can give different coefficients α to different vehicle speed ranges. When the road condition is smooth, the *SSF* has a great impact on the result, and α is set as 1. Conversely, when traffic congestion occurs, the *SSF* almost no longer affects the result, α is set as 0.25. as shown in table 3.

$$SSF(TP_{i\to i+1}) = \alpha \times \frac{\sum_{j=1}^{q}\left(r_j.v \times \bar{v}_{(i\to i+1)}\right)}{\sqrt{\sum_{j=1}^{q}(r_j.v)^2 \times \sum_{j=1}^{q}\bar{v}_{(i\to i+1)}^2}} \tag{3}$$

[3]Since this paper uses trajectory data of Beijing, the setting standards of α will refer to Beijing local standard "The urban road traffic congestion evaluation index system". For details, please refer to the link: https://max.book118.com/html/2017/0209/89663836.shtm.

TABLE 3. Congestion factor.

Velocity Distribution(km/h)	Traffic condition	α
$<= 10$	Serious congestion	0.25
$(10, 20]$	Moderate congestion	0.5
$(20, 30]$	Mild congestion	0.75
> 30	Unblocked	1

c: TRANSITION PROBABILITY (T)

The transition probability will take into account both **DSF** and **SSF**, as shown in equation (4).

$$TP_{i\to i+1} = DSF\,(TP_{i\to i+1}) \times SSF\,(TP_{i\to i+1}) \tag{4}$$

2) EMISSION PROBABILITY

The Emission probability is used to compute the distance similarity between the trajectory point and candidate road segment. The distance similarity is assumed to satisfy the Gaussian normal distribution, which has been proved effective in many papers [12], [23]. It is shown in equation (5). When the distance is nearer, the probability is higher. On the contrary, the probability will be smaller.

$$E\,(TP_i) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\left(\frac{dis}{\sqrt{2}\sigma}\right)^2} \tag{5}$$

The *dis* represents the projection distance from the trajectory point to the candidate road segment. The σ represent standard deviation. In most of the existing papers, σ is fixed, making it impossible to satisfy the different road conditions in reality. To solve this problem, this paper uses Median Absolute Deviation(MAD)[4] train dynamically [25]. The training process is shown in formula (6).

$$\sigma = \begin{cases} 1 & (n = 1) \\ 1.4826 \times median\,(|dis_i - median\,(dis)|) & (n > 1) \end{cases} \tag{6}$$

Where n represents the size of the sliding window. When the sliding window is initialized, set $\sigma = 1$. Once the size of the sliding window changes, the value of the σ changes dynamically. Among them, dis_i is the vertical distance between each TP_i in the sliding window to the candidate road segment, and all dis_i form a data set *dis*. The function *median*() is used to calculate the median of the *dis*.

Through the above process, we have obtained the trajectory points, candidate road segments, transition probability and emission probability, which correspond to observable states, hidden states, transition probability and emission probability in the HMM, as shown in figure 4. It describes the matching process of the best matching sequence, which is implemented by the Viterbi algorithm. The Viterbi algorithm is used to deal with the decoding problem of this type of HMM, and its implementation principle has been described in the subsection A of this section. After the online Viterbi algorithm, the optimal matching sequence will be output as a result.

[4]When the sample obeys the normal distribution, the MAD is usually used to estimate the standard deviation of the sample. The specific derivation process can be seen in the link http://en.wikipedia.org/wiki/Median_absolute_deviation. And the derivation conclusion is $\sigma = 1.4826 \times MAD$.

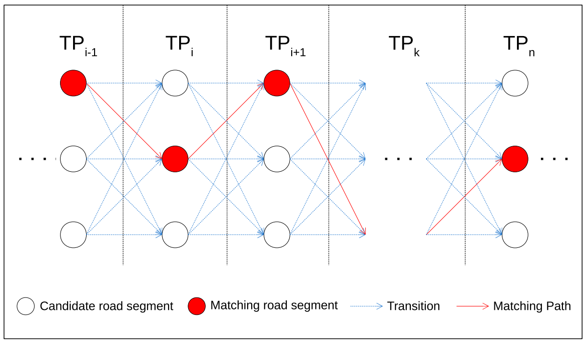

FIGURE 4. The matching process.

D. ADAPTIVE SLIDING WINDOW (ASW) METHOD

Until now, we can see that the approach using the global map matching, which means that it can not be directly applied in the online map matching scenario. This is because that for the online scenario, only a sequence of points before the current one, instead of all the trajectory points, will be available. Hence, the local sliding window strategy is further adopted by the existing global map matching algorithms [12] to solve the online version. In these online algorithms, the sliding window size is set as a fixed value so that we name it as fixed sliding window method(*FSW*).

However, it is difficult to achieve the high accuracy and low latency simultaneously for the *FSW* method. This is because for some road segments, the fixed window size could be too large, which will cause a significant increase in the output delay. On the contrary, for some other road segments, the fixed window size will be too small, so that not enough valid information be offer to identify the optimize path and guarantee the accuracy.

Inspired by this, this paper further proposes an adaptive sliding window method, referred to as *ASW*. The core idea for *ASW* is to automatically change the window size according to the different traffic conditions. So, the dynamic balance can be achieved between the accuracy and the output delay.

Initially, the window size is set as two, as shown in Figure 5(a). Then, based on whether the local path in the window can converge to the leftmost candidate point. we consider the following situations: 1) If it can converge, which indicates that the final matching results will contain this candidate point, and the candidate point is output as a matching result. The above convergent process can be seen in Figure 5(a), the candidate point C_2^1 in the three candidate points of P_2 has the maximum probability. The candidate point C_1^2 in the three candidate points of P_1 has the maximum probability. In addition, the C_1^2 and C_2^1 are reachable. Hence,

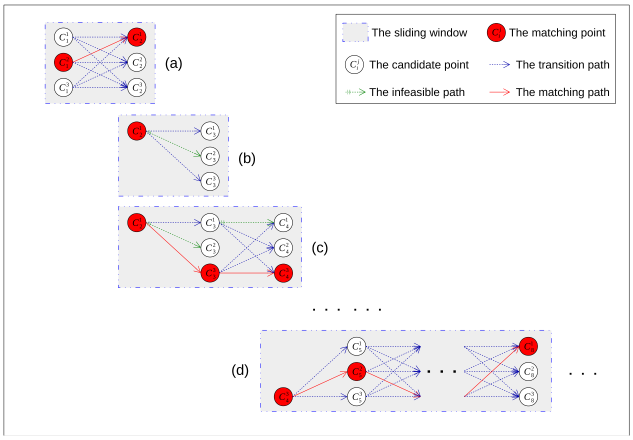

FIGURE 5. The illustration of adaptive sliding window.

C_2^1 can converge to C_1^2, that is, C_1^2 is a matching result. 2) if it cannot converge, the algorithm will continue to increase the window size to the right until the convergence point is found. The above process is presented in Figure 5(b) and 5(c). The candidate point C_3^2 in the three candidate points of P_3 has the maximum probability. However, C_2^1 and C_3^2 are unreachable. Therefore, we need to expand the window to the right. We can find that the candidate point C_4^3 in the three candidate points of P_4 has the maximum probability, and C_2^1 and C_4^3 are reachable. In other words, C_4^3 can converge to C_2^1, that is, C_2^1, C_3^3, and C_4^3 are all matching results. 3) If the convergence point is always not found, it will make the window size increase infinitely, resulting in an increase in the output delay. To avoid this problem, the maximum window size is set as n in this paper. When the window size reaches n while no convergent results are identified, it will stop and output the candidate point with the maximum joint probability corresponding to the leftmost trajectory point. The above process is shown in Figure 5(d).

$$\begin{aligned} O_{(1,k)} &= P\left(TP_{(1,k)}\right) \\ &\vdots \\ O_{(t-1,k)} &= P\left(TP_{(t-1,k)}\right) O_{(t-2,m)} \\ O_{(t,k)} &= P\left(TP_{(t,k)}\right) O_{(t-1,n)} \end{aligned} \tag{7}$$

The above whole adaptive process can be expressed as a formula (7). $TP_{(t,k)}$ denotes the k^{th} candidate point for TP_t. $P\left(TP_{(t,k)}\right)$ represents the probability sum of the trajectory point TP_t, as shown in equation (8). $O_{(t,k)}$ represents the maximum joint probability based on the trajectory sequence $\{TP_1, TP_2, \ldots, TP_t\}$. The index m and n maximize the joint probability are stored as the forward pointer. It points to the previous convergence point.

$$P\left(TP_{(t,k)}\right) = DSF \times SSF + E \tag{8}$$

Algorithm 1 details the sliding window adaptive process. The input parameter is the trajectory sequence in the window, and the output result is the matching result. Line 1 is used to calculate convergence points. If the convergence point is found, line 12 is executed directly to output the matching result. If it is not found, the 2-11 lines are executed to extend the window size. The 5-9 lines are used to determine whether the window has reached the maximum size. If yes, lines 6-7 are executed to output the matching result. Otherwise, line 10 is executed to increase the window size.

Algorithm 1 Adaptive Sliding Window Algorithm(ASW)

```
Input: sequences: store the trajectories in the window;
Output: result: the matching result;
 1: result = getConvergencePoint(sequences);
 2: while result is null do
 3:     size = sequences.getSize();
 4:     if size == n then
 5:         result = getMaxProbabilityPoint(sequences);
 6:         return result;
 7:     end if
 8:     sequences.add(getNextPoint());
 9: end while
10: result = getMaxProbabilityPoint(sequences);
11: return result;
```

E. ADAPTIVE SAMPLING FREQUENCY METHOD

A suitable sampling frequency is crucial for online map matching. The existing map matching algorithms use fixed sampling frequency method, named as *FSF* method. This strategy makes it challenged to achieve the high accuracy and low energy consumption at the same time.

Note that different roads are always corresponding to different road speed limit, road width, and real-time traffic conditions. For some roads, the fixed sampling rate is too high, which means the increasing of available data can hardly improve the matching accuracy but only require more energy to proceed them. For some other roads, the fixed sampling rate could be too low, which means the sampling interval will be large and only too few effective information between the adjacent two trajectory points are available, and the best matching results can not be identified. Hence, it is not suitable to use a fixed sampling frequency for all road conditions.

Based on this, the Adaptive Sampling Frequency Method for urban roads is proposed in this paper, referred to as *ASF* method. The core idea of adaptive sampling rate is that the vehicle can sample at different frequencies on different urban roads. As a result, the matching algorithm can reduce the battery power consumption to the maximum extent while guaranteeing the accuracy. Intuitively, when the vehicle is driving on the urban road, the vehicle speed can reveal the current road condition. When v is small, it indicates that the vehicle may have encountered traffic jams. So even the sampling interval is large, the key information will not be ignored during the driving process. At the same time, it can guarantee the accuracy effect to the maximum extent. On the contrary, when v is high, it indicates that the algorithm can reduce the sampling interval appropriately to ensure sufficient information between adjacent two trajectory points.

As a result, under the prerequisite of sampling interval distance guaranteed, we can dynamically change the sampling frequency according to the current vehicle speed.

The above adaptive sampling frequency determination procedure can be expressed as a formula (9). In this formula, v denotes the vehicle speed, and h represents the adaptive sampling interval distance.

$$SI\,(TP_i) = \frac{h}{v} \tag{9}$$

Algorithm 2 details the adaptive sampling frequency procedure. The input parameters are the velocity v, the timestamp t of the current trajectory point, and the adaptive interval distance h. The output is the next trajectory point. Line 1 calculates the adaptive sampling frequency si. Lines 3-7 are used to calculate the next trajectory point satisfying the sampling interval condition.

Algorithm 2 Adaptive Sampling Frequency Algorithm(ASF)

```
Input: v : the speed of current trajectory point; time : the
    sampling times; h : the adaptive sampling distance;
Output: point : the next point under the adaptive sampling
    interval;
1: si = h / v;
2: while getNowTime() − time >= si do
3:     point = getNextPoint();
4:     return point;
5: end while
```

F. ONLINE MAP MATCHING

Until now, we already detail our approach to integrate different information to increase the accuracy, employ the adaptive sliding window size to reduce the output delay and use the adaptive sampling frequency to reduce the energy consumption. As shown in Algorithm 3, these components work together to achieve the three important measures simultaneously. In algorithm 3, the algorithm 1 is applied to the framework of the entire algorithm. It can be seen that Lines 1-5 construct the initial sliding window. Line 6 uses the *ASW* method to find convergence points. If the convergence point is not found, the lines 8-19 are used to continue to find the convergence point. Among them, Lines 10-14 are used to determine whether the maximum window size is reached. Lines 15-17 enlarge window size by using the *ASF* method. Line 20 generates the matching point sequence on the map for the given GPS trajectory.

V. EXPERIMENT

A. DATA PREPARATION

1) THE TRAJECTORY DATA

The trajectory data is derived from real data provided by CAR Inc during November 2015 Beijing. The sampling interval is 0 $\sim$ 45s, and each record details the longitude, latitude, timestamp and speed of vehicle.

Algorithm 3 Adaptive Online Map Matching Algorithm

```
Input: RN : the road network; T : GPS Trajectory,
    {TP_1, TP_2, ..., TP_n}; h : the adaptive distance;
Output: result : the matched point;
1: initial gpsqueue as an empty list;//trajectory points in the
   adaptive sliding window.
2: p_point = getNextPoint(T);
3: n_point = ASF(p_point.getSpeed(),p_point.getTime(),h);
   //get the next adaptive trajectory by using ASF algorithm
4: gpsqueue.add(p_point);
5: gpsqueue.add(n_point);
6: convergencepoint = getConvergencePoint(gpsqueue);
7: while convergencepoint is null do
8:     if gpsqueue.size() == n then
9:         result = getMaxProbabilityPoint(gpsqueue.poll());
10:        return result;
11:    end if
12:    p_point = n_point;
13:    n_point = ASF(p_point.getSpeed(),p_point.
       getTime(),h);
14:    gpsqueue.add(n_point);
15: end while
16: result = gpsqueue.poll();
17: return result;
```

2) ROAD NETWORK DATA

For the road network data, we use the whole road network in Beijing from the open source site OpenStreetMap,[5] which includes 332112 road segments composed by 182981 intersection points. Then the preprocessing process is used to map the trajectory data to the road network, and only the road network in the mapping area is extracted to complete the later computation. The extracted road network includes 112 road segments composed by 74 intersection points. The above process is completely reasonable because the experiment proves that for the two cases of the whole road network and the mapping road network, 96.05% of the trajectory points get exactly the same candidate road segments, and the algorithm accuracy of the two cases is both 96.41%.

3) VELOCITY DISTRIBUTION

We further calculate the velocity distribution for the trajectory data, as shown in Table 4. It can be seen that more than 48% of the trajectory point velocity is more than 30km/h, more than 41% of the trajectory point speed is between 10 $\sim$ 30km/h. Only about 10% with speeds less than 10km/h.

B. EXPERIMENTAL SETUP

1) PARAMETER SETTINGS

In the comparative experiment, the standard deviation of the measurement error is calculated by using the Median Absolute Deviation formula for our dataset. As a result, the value

[5]www.openstreetmap.org.

TABLE 4. Data of velocity distribution.

Velocity Distribution(km/h)	Number	Percent(%)
$<= 10$	201	9.79%
$(10, 20]$	384	18.70%
$(20, 30]$	476	23.17%
> 30	992	48.34%

of the standard deviation parameter in emission probability models of the HFF and HMM+RCM algorithms both are set as 201 meters. In our experiments, we set the maximum number of candidate points for each trajectory point as $k = 5$, the number of the same prefixes in the string as $r = 5$. In the adaptive sliding window method, the maximum sliding window size is $n = 5$. The adaptive sampling distance is $h = 1500$. We will discuss their impacts on the performance in section V.D.

2) EVALUATION METRICS

As we discussed before, the matching accuracy, the output delay and the energy consumption are critical for the online map matching scenarios. Hence, for each methodology, we will use these three measures to evaluate its performance.

3) THE MATCHING ACCURACY

Ar is calculated the matching accuracy by the formula (10). Ar is the ratio of the number of correctly matched road segments in all road segments. A larger Ar means a higher accuracy.

$$A_r = \frac{\#the\ number\ of\ correctly\ matched\ road\ segments}{\#the\ number\ of\ all\ road\ segments} \tag{10}$$

4) THE OUTPUT DELAY

The output delay for the map matching, $od\,(TP_i)$, refers to the time interval when the trajectory point enters the sliding window until the matching result is calculated, which can be separated into two parts: 1) the time waiting for the sampling, defined as $sp\,(TP_i)$, and 2) the time identifying the matching point, defined as $mp\,(TP_i) = od\,(TP_i) - sp\,(TP_i)$. Given a fixed sampling frequency, $sp\,(TP_i)$ is proportional to the window size. Hence, in the following experiments, we will focus on the window size for the considered algorithms. For the time to identifying the matching trajectory path, we calculate the average time as:

$$OD = \frac{\sum_{i=1}^{p} (od\,(TP_i) - sp\,(TP_i))}{p} \tag{11}$$

5) THE ENERGY CONSUMPTION

Intuitively, the battery power consumption of mobile phones is positively correlated to the number of sampling trajectory points. To illustrate this assumption, the GT-Battery[6] simulator is used to simulate the change of the battery power consumption when sampling different number(500,1000,1500) of trajectory points in five minutes. To make the experimental results more reliable, the experiment was repeated five times, and named as EX1-EX5, respectively. As shown in Figure 6, it can be seen that when 500 points are sampled, the average battery power consumption is 4.85%. When 1000 points are sampled, the average battery power consumption is 10.83%. When 1500 points are sampled, the average battery power consumption is 16.03%. Based on this, we can conclude that the more the number of points is sampled, the more battery power consumption will be. Therefore, in this following experiments, we will use the sampling point number as the measure to evaluate the energy consumption.

[6]http://gt.tencent.com/index.html.

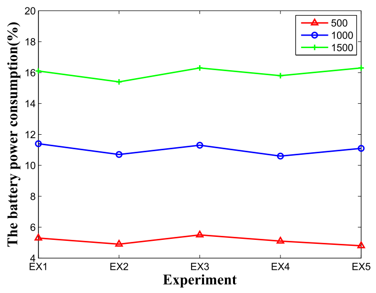

FIGURE 6. The sampling number w.r.t. The battery consumption.

6) COMPARING METHODOLOGY

To demonstrate the effectiveness of the presented approach, AOMM, which employs HMM, ASF and ASW simultaneously, we compare its performance with the following methods:

a: HMM+FSW+FSF, NAMED HFF ALGORITHM

HFF is a basic model used by most of the online map matching algorithms [15]. HMM is used to implement the map matching process, while the fixed window size strategy(FSW) and the fixed sampling strategy(FSF) are employed.

b: HMM+RCM ALGORITHM

HMM+RCM [1] is an online map matching algorithm using HMM and the routing choice model (RCM) to evaluate the probability for each selected path. We named it as HMM+RCM for short. It employs the ASW strategy[15] and FSF strategy. However, the window sliding mode is different from ours, specific performances: HMM+RCM still needs to calculate all the candidate points of the next point after determining a certain convergence point, but AOMM directly determines the other endpoint of the convergence path as the matching point, and continues matching with the endpoint as the starting point.

c: ENACQ ALGORITHM

EnAcq [32] is an online map matching algorithm, this approach employs the same ASW strategy[15] as HMM+RCM and ASF strategy, but the adaptive sampling

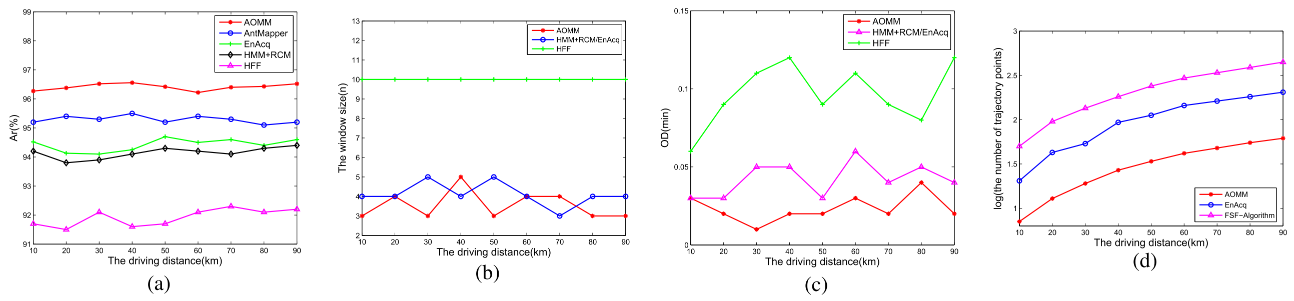

FIGURE 7. **The performance comparison, when FSF = 0.5min, and FSW = 10. (a) The accuracy comparison. (b) The window size comparison. (c) The output delay comparison. (d) The battery comparison.**

mode is different from ours. EnAcq actually uses a semi-adaptive sampling strategy, which gives fixed three speed thresholds, each corresponding to a specific sampling frequency. In contrast, our *ASF* strategy is completely adaptive.

d: ANTMAPPER ALGORITHM

AntMapper [26] uses a global ant colony algorithm to find the shortest path and the local geometric/topological information is employed to match the trajectory points. It is not an online map matching algorithm but an off-line approach. Hence, the sliding window method is not applicable.

In addition, to dig deeper into the performance of the adaptive slide window and the adaptive sampling frequency, we further consider the following two methodologies:

e: HMM+ASW+FSF, NAMED HAF ALGORITHM

For this method, we only use the adaptive slide window strategy but keep the fixed sampling frequency.

f: HMM+FSW+ASF, NAMED HFA ALGORITHM

Unlike 5), we keep the slide window fixed while use the adaptive sampling frequency strategy.

C. EXPERIMENTAL RESULTS AND DESCRIPTION

For the three comparing algorithms, AntMapper, HMM+RCM and HFF, the *FSF* strategy is used so that we named them as FSF-Algorithm for convenient. In addition, we set the fixed window size of HFF as 10. We will discuss the impact of the window size in the following subsection V.D.

Firstly, the fixed sampling frequency for all the three FSF-Algorithms are set as 0.5min. As shown in Figure 7(a), we can observe that the AOMM algorithm always has the highest accuracy, reaching a 96.41% average accuracy. AntMapper algorithm is the following one with a 95.29% accuracy. EnAcq gains a 94.43% accuracy. And HMM+RCM gains a 94.14% accuracy while HFF is the worse, with only 91.92% accuracy. In addition, unlike HFF with the fixed window size 10, we can see from Figure 7(b) that AOMM has a much dynamic and smaller window size, which is fluctuating between 3 and 4, with an average 3.4 window size. Although HMM+RCM and EnAcq use another

TABLE 5. **Accuracy of various algorithms.**

FSF(min)	AOMM	EnAcq	AntMapper	HMM+RCM	HFF
0.1	96.41%	94.43%	96.28%	95.61%	93.86%
0.2	96.41%	94.43%	96.28%	95.55%	93.81%
0.5	96.41%	94.43%	95.29%	94.14%	91.92%
1.5	96.41%	94.43%	94.16%	93.11%	89.97%
3.5	96.41%	94.43%	91.91%	90.12%	82.42%
4.5	96.41%	94.43%	90.19%	88.22%	79.43%
Average	96.41%	94.43%	94.02%	92.79%	88.57%

sliding window algorithm, the required window size is larger than AOMM, which is approximately 4.2. As a result, this will significantly reduce the output latency for the AOMM algorithm as the waiting time to gain the sampling data in the window size will be significantly reduce. In addition, as reported in Figure 7(c), the average matching time of AOMM is about 0.023min, both HMM+RCM and EnAcq are about 0.043min, and HFF is about 0.087min, indicating AOMM gains a 46.51% reduction in matching time comparing with HMM+RCM/EnAcq while it reaches 73.56% reduction when compared with HFF. Finally, Figure 7(d) shows that AOMM requires much less trajectory points. The longer the driving distance is, the larger the reduction of the trajectory points between AOMM, EnAcq and FSW-Algorithms are. Actually, the AOMM only requires about 1/7 trajectories points comparing with the other three FSF-Algorithms(0.5), and only requires about 1/3 trajectories points comparing with the EnAcq. The above all mean that AOMM will significantly reduce the energy consumption. Therefore, comparing with the considered methods, AOMM can achieve a 2.465% average accuracy improvement while at the same time, reduce 64.62% map matching time and need only 1/5 energy consumption.

In addition, to dig into the effectiveness of AOMM in balancing the accuracy and the energy consumption, we vary the fixed sampling frequency of three FSF-Algorithms (AntMapper, HMM+RCM and HFF) among 0.1min, 0.2min, 0.5min, 1.5min, 3.5min and 4.5min. For each algorithm and the sampling frequency, given 90km driving distance, we can calculate its average accuracy. As shown in Table 5, it can be seen that for the three FSF-Algroithms, its accuracy is the closest to ours when the sampling frequency is set as 0.2min and 0.1min. This proves our previous argument that too many sampling data will include some duplicate information and is not necessary to improve the map matching accuracy. In addition, we can see that the larger the fixed sampling frequency is for the FSF-Algorithm, the more the accuracy improvement the AOMM gains. For example, AOMM has a 2.63% average accuracy improvement comparing with the other FSF-Algorithm when the sampling frequency is set as 0.5min. For the scenario in which the sampling frequency is set as 4.5min, this average accuracy improvement will reach 10.46%. This reveals that AOMM can achieve a significant accuracy improvement in the highly sparse location data scenario. For the high sample rate trajectories, AOMM still have a better performance in map matching. Furthermore, as shown in Figure 8(a), we evaluate the number of trajectory points needed for EnAcq and each FSF-Algorithm with different sampling frequency. It can be seen that AOMM has the similar sampling number with FSF(4.5). This means that the AOMM requires the similar energy consumption with the other FSF-Algorithm when the sampling frequency is about 4.5min. Given a 90km driving distance, FSF(0.2) will require 1073 trajectory points, EnAcq will require 184 trajectory points while AOMM only needs 62 trajectory points. It proves that the adaptive sampling algorithm of AOMM can reduce more energy consumption than EnAcq.

Finally, we evaluate the output delay of the HFF at different window size. As discussed above, AntMapper is a off-line map matching algorithm that it considers the global data that the output-delay is not applicable. For the other two ASW algorithms, HMM+RCM and EnAcq, we consider the map matching time as they both have a larger window size comparing with AOMM. As shown in Fig 8(b), the average matching time of AOMM is about 0.023min, while it is about 0.043min for HMM+RCM/EnAcq. When the window size is 10, the average matching time of HFF tends to be stable, about 0.087min. For average, AOMM can achieve a 64.62% reduction in map matching time comparing with other algorithms.

Therefore, we can see that AOMM achieves a better accuracy performance comparing with the state-of-the-art, AntMapper, when the fixed sampling frequency for AntMapper is set as 0.2min. However, AOMM only needs 5.78% of trajectory points in this case. On the other hand, the sampling trajectory points for AOMM is similar to the FSF(4.5min) while comparing with the AntMapper, which has the best accuracy performance in the three FSF-Algorithms, AOMM achieves 10.46% improvement in accuracy. Furthermore, comparing with the FSW-Algorithms, AOMM not only always achieves a higher accuracy and lower energy consumption, but also has a much smaller window size as well as significant reduction (66%) in matching. Besides, compared with the ASW strategy proposed in HMM+RCM, AOMM has a better performance. Our average window size is reduced by 0.8, our average output delay is reduced by 46.51%, and our precision is increased by 3.62%. Similarly, compared with the ASF strategy and the ASW strategy proposed in

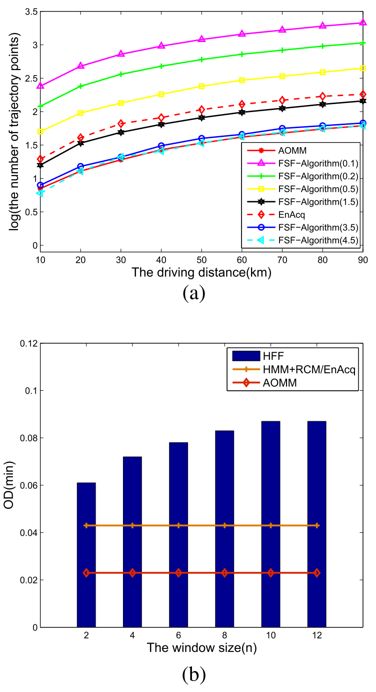

FIGURE 8. The performance comparison in energy consumption and matching time. (a) Eneragy comparison. (b) Matching time comparison.

EnAcq, AOMM also has a better performance. The average sampling number is reduced by 66.30%, the average window size is reduced by 0.8, the average output delay is reduced by 46.51%, and the precision is increased by1.98%. These prove that our approach AOMM can achieve a high accuracy, a low output latency and a low energy consumption simultaneously.

D. PARAMETER INFLUENCE

As discussed in Section IV, in the following sections, we will discuss the impact of the three parameters in AOMM: the maximum candidate points k, the maximum sliding window size n and the sampling distance h.

1) THE INFLUENCE OF THE MAXIMUM CANDIDATE POINTS NUMBER k

Straightforwardly, the number of candidate points k will impact the map matching accuracy. As shown in Figure 9, given $n = 5$, $h = 1500$, it can be seen that with the increase of the candidate points number, the accuracy is constantly improved. This is reasonable because more candidates will enable the algorithm to identify the optimized points. However, it can be seen that the marginal improvement is decreasing. What is worse, the more candidates will increase the time needed to calculate the matching result. The green line in Figure 9 reveals that for our experiment, the running time needed dramatically increases when the candidate number is larger than 5. This is because the increase of the number of candidate points leads to the increase of time complexity. Hence to make the trade-off between the accuracy and the

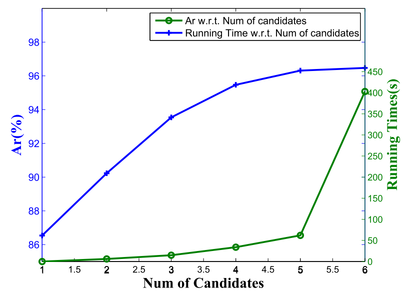

FIGURE 9. Accuracy/Running time w.r.t Number of candidates.

running time, for the previous experiments, we set the candidate points number as 5.

2) THE INFLUENCE OF THE MAXIMUM SLIDING WINDOW SIZE *n*

To dig deeper into the impact of the sliding window size to the performance, in this part, we consider HFA as the comparison. It uses the same adaptive sampling strategy as AOMM. However, unlike AOMM, HFA employs a fixed window size. As shown in Figure 10(a), it can be seen that the AOMM always gains a higher accuracy comparing with HFA, thought when the window size is larger than 18, AOMM and HFA have the very similar performance. Actually, for AOMM, when the maximum window size is set as 5, its accuracy reaches 96.28% and then it stays relatively stable even we continue to increase the maximum window size. As shown in Figure 10(b), We can see that the matching time of the AOMM will tend to be stable more quickly. In average, the output delay of the AOMM is 1.21s while it is 4.17s for HFA. This reveals a 70.98% reduction in matching time for AOMM comparing with HFA. To make the trade-off between the accuracy and the output delay, for the previous experiments, we set the maximum sliding window size *n* as 5.

3) THE INFLUENCE OF ADAPTIVE DISTANCE *h*

Straightforwardly, the sampling frequency distance h will impact the number of the sampling, which then impacts the accuracy. To understand this impact, for AOMM, given $k = 5, n = 5$, we vary h between 500, 1000, 1500, 2000 and 2500. As shown in Figure 11(a), when h is set as 500m, 1000m, and 1500m, the accuracy of above AOMM have the similar average accuracy, which is 96.30%, 96.29% and 96.28%. For AOMM(2000), the average accuracy slightly decreases to 95.16%. For AOMM(2500), the average accuracy is greatly reduced to 90.57%. This means that the accuracy of AOMM is robust to the frequency sampling distance h in a given range, which is 500m to 2500m in our dataset, and then larger h will result into reduction in accuracy because not enough effective sampling is collected. As shown in Figure 11(b), we can see that the larger the distance h is, the fewer the sampling data for map matching are collected. In our dataset, when h is set as 1500m, AOMM can reach the best performance.

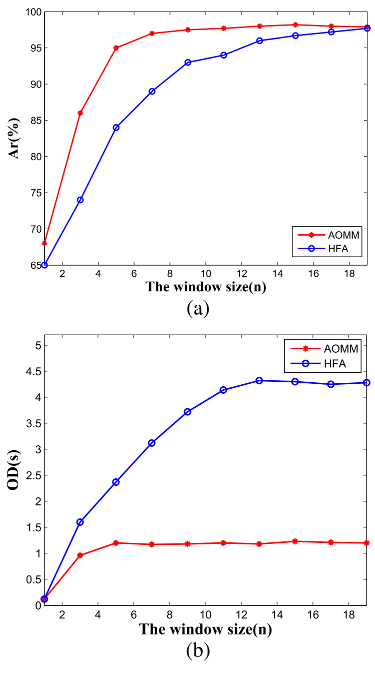

FIGURE 10. The influence of the maximum sliding window size. (a) The influence on accuracy. (b) The influence on latency.

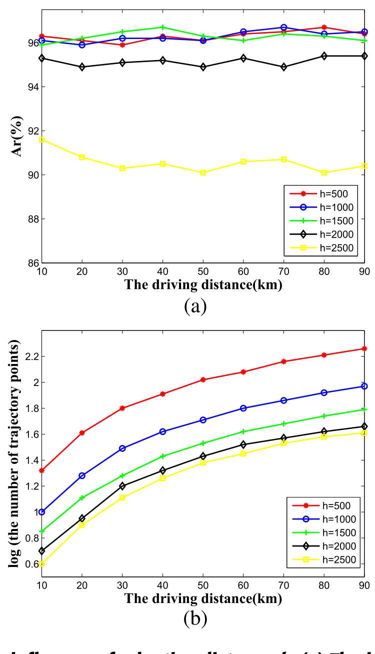

FIGURE 11. The influence of adaptive distance *h*. (a) The influence on accuracy. (b) The influence on battery.

4) THE INFLUENCE OF THE NUMBER OF THE SAME PREFIXES IN THE STRING *r*

Straightforwardly, r will impact the map matching accuracy and the number of calculated candidate road segments. The smaller the r value is, the more number of candidate road segments will be obtained. However, many of these road segments are far away from the trajectory point, and the computation of these road segments is meaningless. Therefore, determining the number of GeoHash string is very

TABLE 6. Data of velocity distribution.

r	The number of candidate road segments	Accuracy(%)
1	332112	96.28
2	115578	96.28
3	21833	96.28
4	6916	96.28
5	926	96.28
6	203	91.21
7	61	83.03

importance, it enables us to find the best results with the fastest time.

In this situation, we did a series of experiments on r, the results is shown in the table 6. It can be seen that when r is set as 5, the candidate road segments number is the minimum for the best accuracy. Reducing r to 1-4 will not improve the accuracy but only increase the candidate road segments while increasing r to higher than 5, will reduce the accuracy.

VI. CONCLUSION

Because of the existence of measurement error, sampling error, and battery power-saving requirements in map matching process, how to match the trajectory points to the digital map in realtime with high accuracy, low latency and low energy consumption simultaneously is an enormous challenge in practice. To deal with this challenge, this paper proposes an adaptive online map matching algorithm called AOMM algorithm. Combing with the geometric structure of the road network (geometric probability), the topology structure of the road network (topology probability) and the time/speed information of the vehicle operation, based on the HMM(a statistical model), the matching probability for each trajectory point is calculated to identify the matching trajectory path. Furthermore, the adaptive window size strategy and the adaptive sampling frequency strategy are developed to reduce the output latency and the sampling data for map matching. Comparing with the state-of-the-art algorithms such as HMM+RCM, EnAcq and AntMapper, our AOMM can achieve a significant performance improvement: 1) Our approach achieves a higher average accuracy, reaching 96.41%. Especially in the high sparse scenario, AOMM can gain a 1.98% accuracy improvement comparing with the state-of-the-art. 2) Employing the adaptive window size strategy benefits us a 19.05% reduction in needed window size and a 46.51% reduction in matching time. 3) The adaptive frequency sampling strategy achieves a 3/4 decreasing of the sampling numbers comparing to the methods with EnAcq and the fixed sampling frequency(0.1min, 0.2min,0.5min,1.5min,3.5min and 4.5min) in our experiment. This demonstrates that our approach can achieve a high accuracy, low output delay and energy consumption reduction at the same time.

In this paper, we introduced the parameter h to achieve adaptive sampling. In the future, we plan to improve the h by bringing more indicators, such as road grade, vehicle angular speed, etc, thus our adaptive sampling algorithm is more adaptable. In addition, in this paper, we only focus on the map matching in city. In the future, we will further extend it to take the freeway into consideration.

REFERENCES

[1] G. R. Jagadeesh and T. Srikanthan, "Online map-matching of noisy and sparse location data with hidden Markov and route choice models," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 9, pp. 2423–2434, Sep. 2017.

[2] J. Yuan, Y. Zheng, X. Xie, G.-Z. Sun, and C. Zhang, "An interactive-voting based map matching algorithm," in *Proc. IEEE 11th Int. Conf. Mobile Data Manage.*, May 2010, pp. 43–52.

[3] A. Gheibi, A. Maheshwari, and J.-R. Sack, "Minimizing walking length in map matching," in *Proc. Int. Conf. Topics Theor. Comput. Sci.* Cham, Switzerland: Springer, 2015, pp. 105–120.

[4] J. Schweizer, S. Bernardi, and F. Rupi, "Map-matching algorithm applied to bicycle global positioning system traces in Bologna," *IET Intell. Transp. Syst.*, vol. 10, no. 4, pp. 244–250, 2016.

[5] Y. Yin, R. R. Shah, and R. Zimmermann, "A general feature-based map matching framework with trajectory simplification," in *Proc. ACM SIGSPATIAL Int. Workshop Geostreaming*, 2016, p. 7.

[6] Y. Cho and H. Choi, "Accuracy enhancement of position estimation using adaptive Kalman filter and map matching," *Int. J. Control Automat.*, vol. 7, no. 7, pp. 167–178, 2014.

[7] E. J. Krakiwsky, C. B. Harris, and R. V. C. Wong, "A Kalman filter for integrating dead reckoning, map matching and GPS positioning," in *Proc. IEEE 21st Century Position Location Navigat. Symp., Rec. Navigat. (PLANS)*, Nov./Dec. 1988, pp. 39–46.

[8] R. Mohamed, H. Aly, and M. Youssef, "Accurate real-time map matching for challenging environments," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 4, pp. 847–857, Apr. 2017.

[9] S. Shahidi and S. Valaee, "Hidden Markov model based graph matching for calibration of localization maps," in *Proc. IEEE Int. Conf. Commun.*, Jun. 2015, pp. 4606–4611.

[10] M. A. Quddus, R. B. Noland, and W. Y. Ochieng, "A high accuracy fuzzy logic based map matching algorithm for road transport," *J. Intell. Transp. Syst.*, vol. 10, no. 3, pp. 103–115, 2006.

[11] C. Smaili, M. E. B. El Najjar, and F. Charpillet, "A hybrid Bayesian framework for map matching: Formulation using switching Kalman filter," *J. Intell. Robot. Syst.*, vol. 74, nos. 3–4, pp. 725–743, 2014.

[12] Y. Lou, C. Zhang, X. Xie, W. Wang, Y. Huang, and Y. Zheng, "Map-matching for low-sampling-rate GPS trajectories," *ACM SIGSPATIAL Int. Symp. Adv. Geograph. Inf. Syst.*, 2009, pp. 352–361.

[13] G. Hu, J. Shao, Y. Wang, H. T. Shen, and F. Liu, "IF-matching: Towards accurate map-matching with information fusion," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 1, pp. 114–127, Jan. 2016.

[14] B. Liang, T. Wang, W. Chen, H. Li, K. Lei, and S. Li, "Online learning for accurate real-time map matching," in *Proc. 20th Pacific–Asia Conf. Adv. Knowl. Discovery Data Mining*, vol. 9652. New York, NY, USA: Springer-Verlag, 2016, pp. 67–78.

[15] C. Y. Goh, J. Dauwels, N. Mitrovic, M. T. Asif, A. Oran, and P. Jaillet, "Online map-matching based on Hidden Markov model for real-time traffic sensing applications," in *Proc. Int. IEEE Conf. Intell. Transp. Syst.*, Sep. 2012, pp. 776–781.

[16] X. Liu, K. Liu, M. Li, and F. Lu, "A ST-CRF map-matching method for low-frequency floating car data," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 5, pp. 1241–1254, May 2017.

[17] X. Liu and F. Lu, "ST-CRF: A novel map matching approach for low-frequency floating car data," *ACM SIGSPATIAL Int. Workshop Geostreaming*, 2015, pp. 9–18.

[18] D. Chen, A. Driemel, A. Nguyen, C. Wenk, and L. J. Guibas, "Approximate map matching with respect to the Fréchet distance," in *Proc. Meeting Algorithm Eng. Exp. Soc. Ind. Appl. Math.*, 2011, pp. 75–83.

[19] J. Shigezumi, T. Asai, H. Morikawa, and H. Inakoshi, "A fast algorithm for matching planar maps with minimum Fréchet distances," in *Proc. Int. ACM SIGSPATIAL Workshop Anal. Big Geospatial Data*, 2015, pp. 25–34.

[20] R. Moussalli, M. Srivatsa, and S. Asaad, "Fast and flexible conversion of geohash codes to and from latitude/longitude coordinates," in *Proc. IEEE 23rd Annu. Int. Symp. Field-Program. Custom Comput. Mach.*, May 2015, pp. 179–186.

[21] Z. Balkić, D. Šoštarić, and G. Horvat, "GeoHash and UUID identifier for multi-agent systems," in *Agent and Multi-Agent Systems. Technologies and Applications*. Berlin, Germany: Springer, 2012, pp. 290–298.

[22] M. Quddus and S. Washington, "Shortest path and vehicle trajectory aided map-matching for low frequency GPS data," *Transp. Res. C, Emerg. Technol.*, vol. 55, pp. 328–339, Jun. 2015.

[23] X. Zhou, Y. Ding, Q. Luo, L. M. Ni, and H. Tan, "HIMM: An HMM-based interactive map-matching system," in *Proc. Int. Conf. Database Syst. Adv. Appl.*, 2017, pp. 3–18.

[24] J. S. Greenfeld, "Matching GPS observations to locations on a digital map," in *Proc. 81th Annu. Meeting Transp. Res. Board*, vol. 1, no. 3, p. 13, 2002.

[25] Y. Wang, Y. Zhu, Z. He, Y. Yue, and Q. Li, "Challenges and opportunities in exploiting large-scale GPS probe data," HP Lab., Palo Alto, CA, USA, Tech. Rep. HPL-2011-109, Jul. 2011.

[26] Y.-J. Gong, E. Chen, X. Zhang, L. M. Ni, and J. Zhang, "AntMapper: An ant colony-based map matching approach for trajectory-based applications," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 2, pp. 390–401, Feb. 2017.

[27] A. Kavcic and J. M. F. Moura, "The Viterbi algorithm and Markov noise memory," *IEEE Trans. Inf. Theory*, vol. 46, no. 1, pp. 291–301, Jan. 2000.

[28] G. Wang and R. Zimmermann, "Eddy: An error-bounded delay-bounded real-time map matching algorithm using HMM and online Viterbi decoder," in *Proc. ACM SIGSPATIAL Int. Conf. Adv. Geograph. Inf. Syst.*, 2014, pp. 33–42.

[29] H. Wang, J. Zhou, Y. Liang, and G. Zheng, "HAS: Hierarchical A-Star algorithm for big map navigation in special areas," in *Proc. Int. Conf. Digit. Home IEEE Comput. Soc.*, Nov. 2014, pp. 222–225.

[30] N. Banerjee, S. Chakraborty, and V. Raman, "Improved space efficient algorithms for BFS, DFS and applications," in *Proc. Int. Comput. Combinat. Conf.* Cham, Switzerland: Springer, 2016, pp. 119–130.

[31] F. Kuipers and F. Dijkstra, "Path selection in multi-layer networks," *Comput. Commun.*, vol. 32, no. 1, pp. 78–85, 2009.

[32] S. Fang and R. Zimmermann, "EnAcq: Energy-efficient GPS trajectory data acquisition based on improved map matching," *Proc. 19th ACM SIGSPATIAL Int. Conf. Adv. Geograph. Inf. Syst.*, 2011, pp. 221–230.

[33] C. Yang and G. Gidófalvi, "Fast map matching, an algorithm integrating hidden Markov model with precomputation," *Int. J. Geograph. Inf. Sci.*, vol. 32, no. 3, pp. 547–570, 2018.

[34] T.-H. Wu and J.-G. Horng, "Establishing an intelligent transportation system with a network security mechanism in an Internet of vehicle environment," *IEEE Access*, vol. 5, pp. 19239–19247, 2017.

[35] S. Adams, A. P. Beling, and R. Cogill, "Feature selection for hidden Markov models and hidden semi-Markov models," *IEEE Access*, vol. 4, pp. 1642–1657, 2016.

QI AN received the B.S. degree in software engineering from Tianjin Normal University, Tianjin, China, in 2016. She is currently pursuing the master's degree in computer science and technology at Tianjin University, Tianjin. Her research interest includes service computing and intelligent transportation.

ZHIYONG FENG (M'10) received the Ph.D. degree from Tianjin University, China. He is currently a Full Professor with the School of Computer Software, Tianjin University. He has authored one book, over 130 articles, and 39 patents. His research interests include knowledge engineering, service computing, and security software engineering. He is a member of the IEEE Computer Society and ACM.

SHIZHAN CHEN (M'10) received the Ph.D. degree from Tianjin University, China. He is currently an Associate Professor with the School of Computer Science and Technology, Tianjin University. He is also leading a research project supported by the National Natural Science Foundation of China. His research interests include service computing and mobile Internet. He is a reviewer of some international conferences and journals.

KEMAN HUANG (M'14) received the B.S. degree from the Department of Automation, School of Economics and Management, Tsinghua University, China, in 2009, and the Ph.D. degree from the Department of Automation, Tsinghua University, in 2014. He was an Assistant Professor with the School of Computer Science and Technology, Tianjin University, from 2014 to 2016. He is currently a Research Scientist with the Sloan School of Management, Massachusetts Institute of Technology, USA. He has authored over 40 journal and conference proceedings papers. His research interests include service ecosystem, cyber security behavior, big data analysis, and semantic Web. He is a member of the ACM. He received the Best Student Paper Award from the IEEE ICWS 2014 and the ICSS 2013. He was in the program committees of many conferences and the Publicly Chair of IEEE ICWS/SCC/MS/BIGDATA Congress 2016.