CLSTERS: A General System for Reducing Errors of Trajectories Under Challenging Localization Situations

HAO WU, Fudan University & Shanghai Key Laboratory of Data Science
WEIWEI SUN, Fudan University & Shanghai Key Laboratory of Data Science
BAIHUA ZHENG, Singapore Management University
LI YANG, HUAWEI Shanghai Research & Development Center
WEI ZHOU, HUAWEI Shanghai Research & Development Center

Trajectory data generated by outdoor activities have great potential for location based services. However, depending on the localization technique used, certain trajectory data could contain large errors. For example, the error of trajectories generated by cellular-based localization techniques is around 100m which is ten times larger than that of GPS-based trajectories. Hence, enhancing the utility of those large-error trajectories becomes a challenge. In this paper we show how to improve the quality of trajectory data having large errors. Some existing works reduce the error through hardware which requires information such as the time of arrival (TOA), received signal strength indication (RSSI), the position of cell towers, etc. Moreover, different positioning techniques will result in different hardware-based solutions and different data formats, which limit the generalizablity. Other works study a related but different problem, i.e., map matching, with the aid of road network information, to reduce the uncertainty and the noise of trajectory data. However, most of these approaches are designed for the GPS-sampled data, and hence they might not be able to achieve a similar performance when applied directly to trajectories with large errors. Motivated by this, we propose a general error reduction system namely CLSTERS for trajectories with large scale of errors. Our system is hardware independent and only requires the coordinates and the time stamp of each sample point which makes it general and ubiquitous. We present results from experiments using three real-world datasets in three different cities generated by two different localization techniques and the results show that our approach outperforms existing solutions.

CCS Concepts: • **Human-centered computing → Ubiquitous computing**; **Mobile computing**;

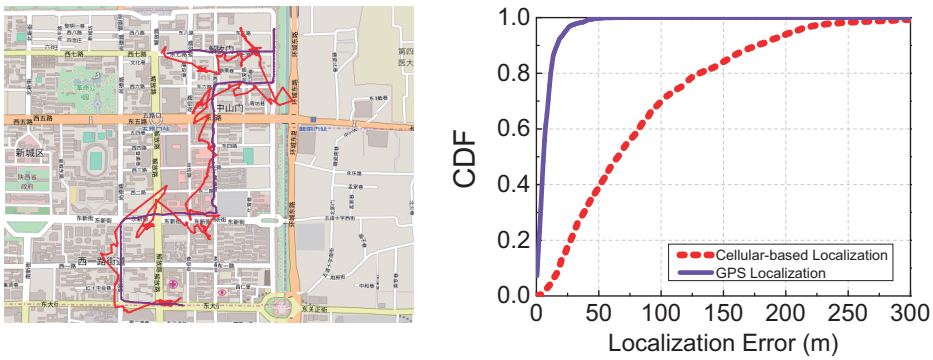
Additional Key Words and Phrases: Localization, error reduction, cellular-based trajectory, map matching

ACM Reference format:
Hao Wu, Weiwei Sun, Baihua Zheng, Li Yang, and Wei Zhou. 2017. CLSTERS: A General System for Reducing Errors of Trajectories Under Challenging Localization Situations. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* 1, 3, Article 115 (September 2017), 28 pages.
DOI: http://doi.org/10.1145/3130981

1 INTRODUCTION

Outdoor activities generate large quantities of trajectories with the development of localization techniques, such as localization based on GPS, WiFi, or cellular network. On top of these trajectory data, many location-based services (LBSs) applications emerge, including travel time estimation [36], traffic flow detection and prediction

ACM acknowledges that this contribution was authored or co-authored by an employee, or contractor of the national government. As such, the Government retains a nonexclusive, royalty-free right to publish or reproduce this article, or to allow others to do so, for Government purposes only. Permission to make digital or hard copies for personal or classroom use is granted. Copies must bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. To copy otherwise, distribute, republish, or post, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2017 Association for Computing Machinery.
2474-9567/2017/9-ART115 $15.00
DOI: http://doi.org/10.1145/3130981



(a) Example of cellular-based trajectory

(b) CDF

Fig. 1. Visualization and statistics of a GPS-based trajectory vs. a cellular-based trajectory.

[11, 23] and taxi hailing and sharing [24, 28]. It is well-known that the trajectory data have inevitable errors due to the limitations of localization techniques. In order to guarantee the service quality of LBSs built on top of the trajectory data, we dedicate this paper on improving the quality of trajectory data.

Given trajectories generated by different types of localization techniques, we focus on those with large error scales (e.g., cellular-based trajectories). GPS-based positioning, although having relatively small errors (about 10 meters), has some restrictions that severely reduce its generality, especially in the context of *ubiquitous computing*. GPS signals are carried through waves at a frequency that does not move easily through solid objects. Consequently, GPS signals are weak or even not available inside a building or inside a tunnel. In addition, GPS is expensive in terms of power consumption and hence many users tend to turn off GPS when GPS-based positioning is not required. On the other hand, some localization techniques that are able to generate trajectories but with relatively large errors are really universal. Take the cellular network as an example. As reported by *statista* (https://www.statista.com), the number of smartphone users worldwide is around 2.32 billions while that of mobile phone users is 4.77 billions. In addition, the cellular-based trajectories can be easily generated without mobile users turning on anything, and its signal strength is better than GPS.

In order to visualize the error ranges of GPS-based trajectory and cellular-based trajectory, Fig. 1(a) plots two trajectories. The one aligning nicely with the underlying road network is generated by GPS while the other one that is much noisier with large scale of errors is generated by cellular network. According to the cumulative distribution of the error for the cellular-based trajectory plotted in Fig. 1(b), we can observe that the error is in the range of tens to hundreds meters that is much larger than that of GPS. Hence, how to enhance the utility of the trajectories with large errors becomes a challenge.

In the literature, some works try to reduce the error for cellular localization through hardware improvements [4, 5, 7, 12, 14–16, 18, 33, 35], i.e., the techniques depend on the hardware environment rather than only depending on the coordinates of original trajectories. However, hardware-based approaches lack the generality and are only applicable in certain hardware environments. In addition, hardware-based approaches require additional information to which the LBS providers might not have direct access, including the time of arrival (TOA) [4, 18], received signal strength indication (RSSI) [14–16, 35], the position of cell towers [35] or Cell-ID [5, 7, 12, 33]. Alternatively, we want to propose a solution that only requires coordinates and time stamps, and hence *independent on* hardware. Even when mobile users are using different types of networks (e.g., GSM, UMTS, LTE), and the raw data of the cellular-based trajectories generated are in very different formats, the coordinates and the time stamps are definitely available. Thus, it is more preferable to adopt a system which only needs to read in the

coordinates and time stamps information of a trajectory. Ideally, such an approach will be transparent from the applications, and ubiquitously applicable to any device.

The problem of *map matching* actually solves a related problem with *the assistance of road network information*. Given the fact that road network data is available (e.g., OpenStreetMap, GoogleMap), and most trajectories are restricted by road network, map matching techniques [22, 25–27, 32, 37, 41] try to map the trajectories to road network which helps to reduce certain errors. However, most of existing map-matching techniques are designed for GPS-based trajectories, and they might not be able to achieve the same performance when applied directly to trajectories generated by other localization techniques with large errors. To the best of our knowledge, map-matching technique presented in [25] is the only piece of work designed for cellular-based trajectories. In addition, given a noisy measurement p that is sampled from a real location q along a road segment r, let ε_y indicate the vertical distance from p to r and ε_x indicate the distance from p to q *along* the segment (i.e., the full error $\varepsilon = dist(p, q) = \sqrt{\varepsilon_x^2 + \varepsilon_y^2}$). Map-matching techniques help to reduce ε_y since they can figure out the right road segment that a given measurement p is on, but they are not able to reduce the along-road error ε_x. An approach that can reduce both ε_x and ε_y (and hence the full error ε) is preferred.

Motivated by the importance of the problem and the lack of solutions, we propose a general error reduction approach for trajectories capturing movements restricted by underlying road networks *with large errors*. Our solution only requires the coordinates and time stamps that are universally available on trajectory data generated by different localization techniques with large noise. To achieve our goal, we theoretically study the error property and make use of all the information that is available, including the road network, the position information (i.e., the coordinates), the temporal information (i.e., the time stamps) as well as the motion property of moving objects. Moreover, we design our model carefully to make it adaptive to different sampling rates. The main contributions of this work can be summarized as follows.

- We propose a general approach for reducing the location error of trajectories restricted by road network. Our approach is based on the coordinates and the time stamps, i.e., information available to lower-level localization techniques. In particular, it focuses on trajectories that are subject to large errors. To the best of our knowledge, this is the first attempt to solve this challenging problem in such a way.
- We propose a novel calibration flow for handling error-prone trajectories given $(x, y, time)$ information. Especially, by making use of road network information, we propose a heuristic candidate selection mechanism and calibrate the trajectory points through interpolation. The idea is novel for tackling such a problem. Moreover, we perform a thorough study to understand the properties of the potential errors of trajectories, which provides a theoretical foundation to our solution.
- We conduct comprehensive experiments through real datasets collected in three different cities generated by different localization techniques. The results demonstrate the effectiveness of our approach. To be more specific, the results show that our approach is always applicable as long as the coordinates and the time stamp are available, regardless of the detailed localization technique used to generate the data.

2 RELATED WORK

2.1 Cellular-based Localization and Error Reduction

In the literature, there are many researches on increasing the localization accuracy on cellular networks. One commonly used localization technique is Cell-ID whose mechanism is very simple. Each base transceiver station keeps broadcasting its Cell-ID messages to the cellphones within its signal range; while the cellphone receives the Cell-ID and uses the position of corresponding base station as its own position [33]. A similar approach is adopted by Google's MyLocation [1]. It is often combined with other information to reduce the error, e.g., work presented in [5] uses Cell-ID and round trip time (RTT) information to further improve the localization accuracy.

Time-of-arrival (TOA) is another way for localization under cellular networks. It estimates the position by the time a signal takes to travel from the cell phone to the base station [18]. Angle-of-arrival based technique uses the antenna arrays to measure the angle between the base stations and the cell phone to estimate the location [21]. This approach requires specialized devices, i.e., antenna arrays. Thus it is not commonly adopted. RSSI is also an information source for localization. Most RSSI-relevant approaches are based on fingerprinting techniques. They store the RSSI signature of cell towers in the database and match the current RSSI to the signature in the database to get the location. Fingerprinting techniques can be clustered into deterministic and probabilistic ones. Deterministic fingerprinting techniques store a vector representing the signature of each cell tower and mostly use k-nearest neighbors classification algorithm to match the current RSSI to the signature [10, 35]. Probabilistic fingerprinting techniques store information about the distribution of RSSI and use Bayesian inference approach to estimate the position [14, 16]. [15] also uses the RSSI information but is not fingerprinting-based, instead it uses hidden Markov-model (HMM) by modeling RSSI as the observation and the grid location as the hidden states to estimate the location of the cell phone. However, these approaches are all hardware relevant and need other information or specific devices which violate the ubiquity. On the other hand, our approach only needs the coordinates and the time stamp that are universally available. The fact that our approach does not require other sensors or information makes our approach general and ubiquitous. We claim that our approach is orthogonal to hardware based approaches, and it can be applied after applying any of these hardware-based approaches.

2.2 Map Matching

On the other hand, there are a series of researches on map matching. They can be generally divided into two categories, namely *geometric-based* and *probabilistic-based*. The geometric approaches only consider the geometry information of the trajectory and the road segments such as the point and the curve. The simplest approach is based on point-to-point matching [3] which maps the trajectory points to the closest nodes or shape points in the road network. Another alternative is the point-to-curve matching. Instead of mapping the trajectory points to the points, it maps each trajectory point to the nearest road segment by defining the distance between the point and the segment [3]. Other geometric-based approaches solve the problem in a curve-to-curve matching way by defining the distance metric between the trajectory and the route such as using the Fréchet Distance [6, 9] or path shape measures [13]. However, these approaches are often not suitable for the trajectories with long sampling intervals and large noise. The probabilistic approaches consider the connectivity and transition information of the road network by the probabilistic model (mostly HMM-based). [20] is the first approach adopting the HMM by combining it with the Kalman filter which did not consider the transition information at that time. [19] first considers the transition information and models it into HMM. However, the most classic one is [26], which proposes an HMM-based map matching model by modeling the road segments as the hidden states and the trajectory points as the observations. By properly defining the transition probability and the emission probability, this approach shows strong robustness to the noise and sparsity on GPS trajectories. Furthermore, its variants [32, 37] achieved the first and the second places in the map matching contest, ACM SIGSPATIAL Cup 2012 [2].

In addition to the above approaches, there are some approaches designed for the sparse-sampled trajectories. [22] proposes an approach called ST-Matching for low sampling rate GPS trajectories. [41] proposes an interactive-voting based approach to solve the data sparsity problem. [27] studies the driving behavior and adopts the maximum entropy inverse reinforcement learning model to perform map matching under sparse-sampled trajectories. [42] proposes an approach that incorporates both temporal and spatial dynamics to recover the path between two distant positions in a sparse-sampled trajectory.

However, these map matching approaches are all designed for the GPS trajectories which have small error scale thus they might not be able to achieve high accuracy when applied to large-error trajectories. Among all map matching approaches, [25] is the only solution proposed for cellular-based trajectories. However, it still can

not eliminate ϵ_x, the error along the road. In contrast, our approach is designed for large-error trajectories and can reduce both the vertical distance and the error along the road (i.e., both ϵ_x and ϵ_y) to provide trajectory data with better quality for LBS applications.

3 PRELIMINARY

In this section, we first present several important definitions, and then formalize the problem studied in this paper.

Definition 3.1 (Road Network). A road network is modeled as a directed graph $G(V, E)$, where V refers to the vertex set representing crossroads and E refers to the edge set representing road segments. Each edge $r \in E$ is from a vertex $v \in V$ to another vertex $v'(\neq v) \in V$ where $r.s = v$ and $r.e = v'$ represent the source and the end of the edge respectively.

Definition 3.2 (Trajectory). A trajectory $\mathcal{T} = \{p_1 \rightarrow p_2 \rightarrow \cdots \rightarrow p_N\}$ is a list of locations ordered by time, where $p_i = \{p_x^{(i)}, p_y^{(i)}, t^{(i)}\}$ records the x-coordinate, the y-coordinate and the time stamp of the i-th location. Note that in some places we also use the notation of the bold alphabets, e.g., $\mathbf{p} = \begin{bmatrix} p_x \\ p_y \end{bmatrix}$, which explicitly declares that it is a vector/matrix.

Definition 3.3 (Route). A route $\mathcal{R} = \{r_1 \rightarrow r_2 \rightarrow \cdots \rightarrow r_M\}$ is a list of adjacent road segments recording the path of an object moving restricted by a road network G, where each two consecutive road segments are connected in G, i.e., $r_i.e = r_{i+1}.s$. The length of a route $\mathcal{R}$, denoted as $len(\mathcal{R})$, refers to the total length of the segments passed by, i.e., $len(\mathcal{R}) = \sum_{r_i \in \mathcal{R}} len(r_i)$ where $len(r_i)$ is the length of road r_i.

Definition 3.4 (Calibration error). Given a calibrated trajectory $\tilde{\mathcal{T}} = \{\tilde{p}_1 \rightarrow \tilde{p}_2 \rightarrow \cdots \rightarrow \tilde{p}_N\}$ that is calibrated by an error reduction technique from the original noisy trajectory $\mathcal{T} = \{p_1 \rightarrow p_2 \rightarrow \cdots \rightarrow p_N\}$, and the ground truth trajectory $\mathcal{T}_{gt} = \{q_1 \rightarrow q_2 \rightarrow \cdots \rightarrow q_N\}$, the calibration error of $\tilde{\mathcal{T}}$, denoted as $\varepsilon_{\tilde{\mathcal{T}}}$, is set to the average distance from a point $\tilde{p}_i \in \tilde{\mathcal{T}}$ to its counterpart point $q_i \in \mathcal{T}_{gt}$, i.e., $\varepsilon_{\tilde{\mathcal{T}}} = \frac{1}{N} \sum_{i=1}^{N} \sqrt{\left(\tilde{p}_x^{(i)} - q_x^{(i)}\right)^2 + \left(\tilde{p}_y^{(i)} - q_y^{(i)}\right)^2}$.

Problem Formalization. Given a noisy trajectory $\mathcal{T}$ generated from a true trajectory $\mathcal{T}_{gt}$ moving restricted by the road network G, and an error reduction technique $\mathcal{A}$, let $\tilde{\mathcal{T}}$ be the output of $\mathcal{A}$ with $\mathcal{T}$ being the input and we have $|\tilde{\mathcal{T}}| = |\mathcal{T}|$. The performance of $\mathcal{A}$ is evaluated by the calibration error $\varepsilon_{\tilde{\mathcal{T}}}$, the smaller the better.

4 CLSTERS

In this section, we propose our Challenging Localization Situation-aimed Trajectory Error Reduction System, in short CLSTERS. CLSTERS consists of four phases, i.e, filtering, route inference, candidate construction and interpolation, as shown in Fig. 2.

Filtering performs rule-based filtering to purify the original noisy trajectory $\mathcal{T}$ by removing points having large errors or abnormal behaviors, with the help of three filters, i.e., speed filter, angle filter and weighted mean filter. The output of this step is a purified trajectory $\hat{\mathcal{T}}$. Second, **route inference** is to incorporate road network information. It maps $\hat{\mathcal{T}}$ to a route $\mathcal{R}$ in road network G through an HMM-based map matching algorithm. Then, **candidate construction** procedure is invoked to re-select the point from $\mathcal{T}$ having small error heuristically with the aid of route $\mathcal{R}$. Bayesian smoothing technique is performed to further reduce the error of the selected candidates and to return the smoothed candidate points $\tilde{C}$. Finally, **interval-based interpolation** is adopted to interpolate the remaining points $(\mathcal{T} - \tilde{C})$ w.r.t. the candidates $\tilde{C}$ to get the calibrated trajectory $\tilde{\mathcal{T}}$ with smaller calibration error. In the rest of this section, we will present the details of these four steps.

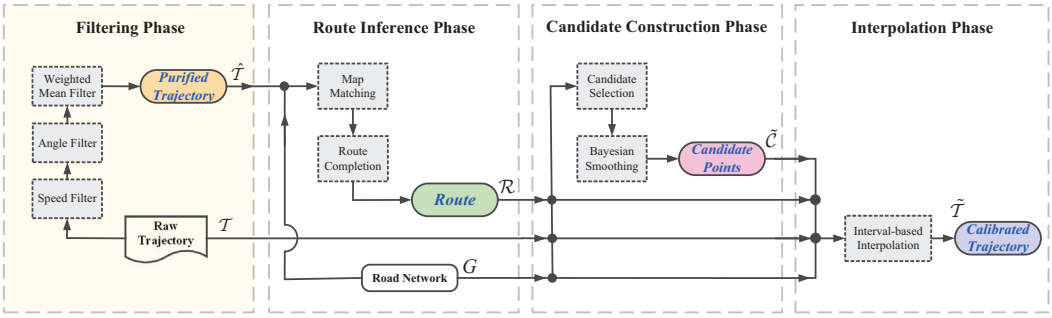
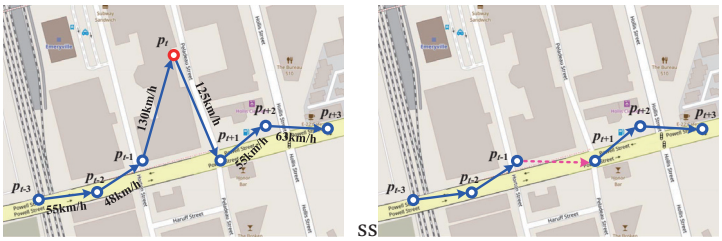


Fig. 2. System overview of CLSTERS. The arrow represents the input and output of each component.

4.1 Filtering Phase

Inspired by the work [25], we adopt rule-based filters on the original trajectory $\mathcal{T}$ to perform the filtering, via three filters, i.e., *speed filter*, *angle filter* and *weighted mean filter*. The first two filters filter out the points which are contradict to our common knowledge and the third one is to filter out the outliers and to smooth the trajectory.

4.1.1 Speed Filter. Work [25] estimates the speed of a point, say p_t, as the mean velocity using a window and filters the point according to the speed limitation of the road. However, there are some issues with this strategy. When we use a window to average the speed of a point with significant errors, the speed may be averaged to a legal speed and hence the point will *not* be filtered out. Take the outlier point p_t shown in Fig. 3(a) as an example. If we use the average speed in the context window around it, say a window covering 7 points from p_{t-3} to p_{t+3}, the speed of p_t will be estimated as 79km/h which is normal and thus point p_t will not be filtered out. Consequently, we adopt a different approach. Our speed filter will filter out the outlier if the immediate speed of a point exceeds the threshold v_{max} as shown in Fig. 3(b).

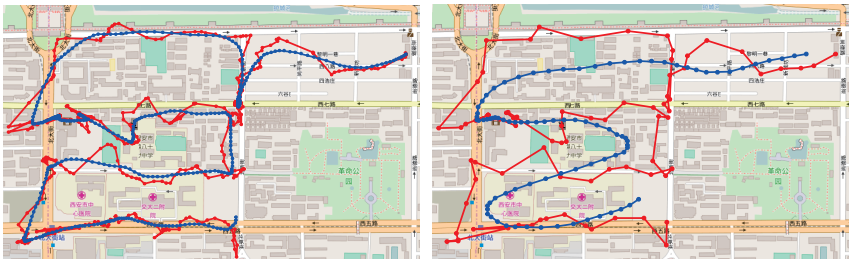


(a) Example of an original trajectory (b) Result of the trajectory adopting the speed filter

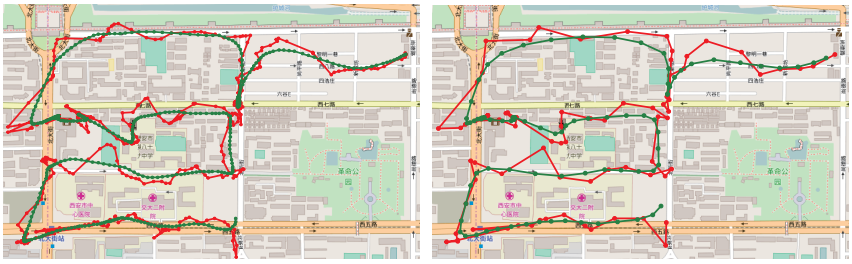
Fig. 3. Example of the poor performance by estimating the mean speed using a window. Point p_t is the objective point which will not be filtered out if the speed of p_t is averaged by points inside the window. It will be filtered out according to our speed filter.

4.1.2 Angle Filter. Besides the irregular immediate speed, another common character shared by the outliers is that outliers tend to form some sharp angles. These sharp angles will result in additional challenges for map matching since the transition probability will be wrongly computed and the output route might have low route accuracy. Thus, the angle filter is performed to filter points causing sharp angles. In detail, for a currently scanned point p_t if the angle $\angle p_{t-1}p_tp_{t+1}$ is smaller than a given threshold θ_{min} and angle $\angle p_tp_{t+1}p_{t+2}$ is also smaller than θ_{min}, then a "ping-pong effect" (i.e., zig-zag shape) is captured and as a result p_t will be regarded as an angle outlier to be filtered out.

4.1.3 Weighted Mean Filter. After filtering the points having abnormal behaviors, we adopt a *mean filter* to i) smooth the trajectory as well as to ii) detect some outliers not captured by the first two filters. A mean filter is a filter with the window size η defined, by averaging coordinates from $p_{i-\lfloor \eta/2 \rfloor}$ to $p_{i+\lfloor \eta/2 \rfloor}$ to adjust the position of p_i, which has been proved to be a useful approach for smoothing the trajectory [43]. However, there are some issues, if we directly adopt the mean filter in our problem. This is because the traditional mean filter averages coordinates according to a window defined by the *number* of points. If the window is too small, the smoothing effect will be insignificant when the sampling interval is low. On the other side, if the window is very big but the sampling interval is high, it will over-filter the data. To visualize the issues of traditional mean filter, two cellular-based trajectories with different sampling rates, together with their respective smoothed versions, are plotted in Fig. 4(a) and Fig. 4(b). The trajectories in red color with sharp angles are the original ones, and the blue ones refer to the smoothed trajectories via mean filter, with window size $\eta = 15$. We can observe that for the low sampling interval trajectory as shown in Fig. 4(a), the mean filter performs relatively well; while if we still use the same window size but increase the sampling interval, the trajectory will be adjusted in a wrong shape as the points in the window become faraway from each other and are eventually located on different roads/directions, as shown in Fig. 4(b). Although the problem can be solved by tuning different η as a parameter to fit data with different sampling intervals, it is not a delicated solution and meanwhile violates the generality.



(a) Sampling interval = 2s, traditional mean filter (b) Sampling interval = 10s, trandition mean filter



(c) Sampling interval = 2s, weighted mean filter (d) Sampling interval = 10s, weighted mean filter

Fig. 4. Sample cellular-based trajectories and their smoothed versions smoothed by the traditional and weight mean filter with $\eta = 15$. The trajectory on the right side (sampling interval = 10s) is sub-sampled from the trajectory on the left side (sampling interval = 2s) and η is set to a fixed value 15 to study the performance of these two mean filters.

To solve this problem, we propose a *weighted mean filter*. The main idea is that, when we smooth a point p_t, those points located far away from p_t shall have smaller influence than those close to p_t. Thus, we use a Gaussian function to model the influence of a point p_j on another point p_i. In detail, for a point p_j in the window with time stamp t_j, the weight function $w(p_i, p_j)$ w.r.t. the scanned point p_i is modeled as

$$w(p_i, p_j) = \frac{1}{\sqrt{2\pi}\sigma_m} \exp\left[-\frac{(t_j - t_i)^2}{2\sigma_m^2}\right] \tag{1}$$

Note that for some points which may be potential outliers and will significantly affect the mean value, we truncate the window by a minimum and a maximum values. Thus, for an η-length window around the objective point p_i, we sort the points fallen within the window according to x-coordinates and y-coordinates, respectively to generate two ordered lists, denoted as $\mathcal{L}_x = \{p_x^{(x_1)}, p_x^{(x_2)}, \cdots, p_x^{(x_\eta)}\}$ and $\mathcal{L}_y = \{p_y^{(y_1)}, p_y^{(y_2)}, \cdots, p_y^{(y_\eta)}\}$, where $p_x^{(x_1)} \leq p_x^{(x_2)} \leq \cdots \leq p_x^{(x_\eta)}$ and $p_y^{(y_1)} \leq p_y^{(y_2)} \leq \cdots \leq p_y^{(y_\eta)}$. In addition, we denote the corresponding order of the points by $p_{x_1}, p_{x_2}, \cdots, p_{x_\eta}$ and $p_{y_1}, p_{y_2}, \cdots, p_{y_\eta}$. We then adjust the coordinates of p_i by the truncated weighted mean filter as following,

$$\hat{p}_x^{(i)} = \frac{1}{\sum_{k=2}^{\eta-1} w\left(p_x^{(x_k)}, p_i\right)} \sum_{j=2}^{\eta-1} w\left(p_i, p_x^{(x_j)}\right) \cdot p_x^{(x_j)}$$

$$\hat{p}_y^{(i)} = \frac{1}{\sum_{k=2}^{\eta-1} w\left(p_y^{(y_k)}, p_i\right)} \sum_{j=2}^{\eta-1} w\left(p_i, p_y^{(y_j)}\right) \cdot p_y^{(y_j)}$$

The smoothed trajectories (in green color) via weighted mean filter of the previous two sample trajectories are plotted in Fig. 4(c) and Fig. 4(d) for comparison purpose. We can observe that weighted mean filter effectively preserves the shape of the trajectory when the sampling interval is high. In addition, it can also perform well in the low sampling interval data without tuning the window size as shown in Fig. 4(c). Consequently, weighted mean filter is general, stable and meanwhile resilient to sampling rates of trajectories.

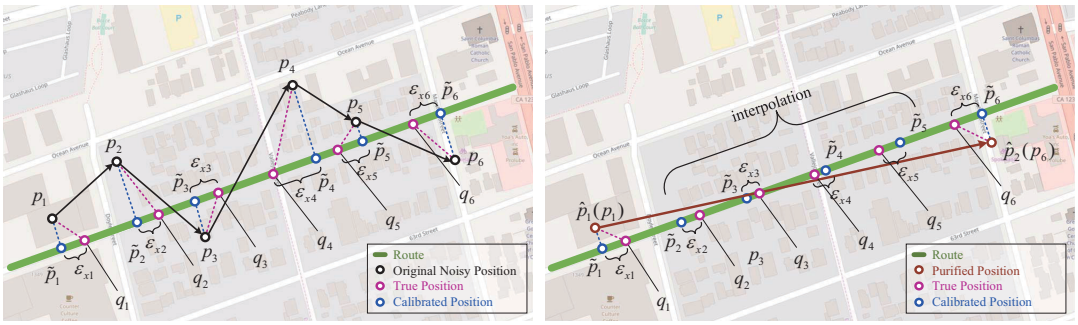
4.2 Route Inference Phase

After purifying the original trajectory $\mathcal{T}$ via three filters, most outliers will be removed from the new trajectory $\hat{\mathcal{T}}$. Next, we try to infer the route $\mathcal{R}$ according to the purified trajectory $\hat{\mathcal{T}} = \{\hat{p}_1 \rightarrow \hat{p}_2 \rightarrow \cdots \rightarrow \hat{p}_{\hat{N}}\}$. This step is to reduce the errors by fully utilizing the knowledge on underlying road network.

We adopt the hidden Markov model (HMM)-based map matching algorithm [26] to map each point in $\hat{\mathcal{T}}$ to a road segment. Since we have removed many outliers, the quality of the trajectory is improved. We adopt the HMM-based map matching algorithm to perform map-matching, which outputs a sequence of road segments $\{r_1, r_2, \cdots, r_{\hat{N}}\}$ such that for each point $\hat{p}_i \in \hat{\mathcal{T}}$, r_i is the corresponding road that $\hat{p}_i$ is most likely to locate on. Dependent on the sampling rate of trajectories, the output road segments r_is from a map-matching algorithm might not be able to form a route defined in Definition 3.3 (e.g., when $r_i.e \neq r_{i+1}.s$). Consequently, we need to complete the route by recovering the travel journey not captured by r_is. Here, we decide to recover the journey from r_i to r_{i+1} by the shortest path. This is because r_i is expected to be not far away from r_{i+1}, and people tend to use shortest path for a short journey, as suggested by [38].

4.3 Candidate Construction Phase

After the initial two phases, we get a purified trajectory $\hat{\mathcal{T}}$ ($|\hat{\mathcal{T}}| \leq |\mathcal{T}|$) and a corresponding route $\mathcal{R}$. Ideally, we could further reduce the error by leveraging the information captured by route $\mathcal{R}$. Fig. 5 plots a simple approach under the case of two different input trajectories. As trajectories capture the movement along the road network, the real points of a trajectory are expected to be located along the segments. Consequently, the projected location $\tilde{p}_i$ of a point p_i (or $\hat{p}_i$) along the corresponding road segment $r_i \in \mathcal{R}$ that p_i is expected to be located on could serve as a calibrated position. Fig. 5(a) shows an example if we take *the raw trajectory* $\mathcal{T}$ as an input, which can reduce the vertical error. Alternatively, we could also take *purified trajectory* $\hat{\mathcal{T}}$ as an input. We want to highlight that the number of points in $\hat{\mathcal{T}}$ could be different from that of $\mathcal{T}$, as some outlier points are removed. As the sample trajectory shown in Fig. 5(b), we have $|\hat{\mathcal{T}}| = 2$, which is smaller than the original count $|\mathcal{T}| = 6$ since $p_2 \sim p_5$ are all filtered out in the filtering phase. After we project points in $\hat{\mathcal{T}}$ to the road segments of $\mathcal{R}$, we can



(a) Based on raw trajectory $\mathcal{T}$ (b) Based on purified trajectory $\hat{\mathcal{T}}$

Fig. 5. Leveraging route information to further reduce the error.

recover the points in $\mathcal{T}$ but not captured by $\hat{\mathcal{T}}$, i.e., $p_2 \sim p_5$, via certain interpolation strategy. As points filtered out in the first filtering phase are considered to be outliers, they are expected to have relatively large errors. Consequently, utilizing the points in $\hat{\mathcal{T}}$ and recovering the positions of filtered outline points via interpolation might be a better approach. As shown in Fig. 5, leveraging route information and trajectory data (either $\hat{\mathcal{T}}$ or $\mathcal{T}$) is able to reduce the vertical distance ε_y but the error along the road (i.e., ε_x) is still there. Thus, in order to utilize the route information in a more effective way and to find ways to reduce along road error ε_x as well, we perform a thorough study to understand the properties of the errors better, based on which a new approach is proposed.

4.3.1 Noise Model. We first define the noise model of localization for one point. To simplify the representation, we define the coordinates by setting the direction along the road segment as the x-axis and the direction vertical to the road segment as the y-axis, as shown in Fig. 6. It is natural to assume that the measurement $\mathbf{p}$ is generated by the true position $\mathbf{q}$ with a zero-mean 2-D Gaussian noise $\mathcal{N}(\mathbf{0}, \Sigma)$, as defined in [26, 39]. Thus, by denoting the true position of the object $\mathbf{q} = \begin{bmatrix} q_x \\ q_y \end{bmatrix}$ which is located on the road segment, the probability of generating a noisy measurement $\mathbf{p}$ given its true position is $P(\mathbf{p}|\mathbf{q}) = \mathcal{N}(\mathbf{q}, \Sigma)$. Note that since we have defined the x-axis as the road segment itself, we can infer that $q_y = 0$.

As the route $\mathcal{R}$ corresponding to a trajectory has been inferred by route inference phase, we are aware of ε_y, the error vertical to the road segment, by computing the projection distance from the observation $\mathbf{p}$ to the road segment. Unlike the situation with only raw trajectory $\mathcal{T}$, now we have more valuable information, i.e., $\mathcal{R}$ and ε_ys. The issue we want to address is how to estimate the full error ε of any point given the knowledge on its ε_y. If we can address this issue, we will be able to *selectively* choose some points with small estimated error to be the candidates working as the references of interpolation, and hence have the control on the final error. Theorem 4.1 presented below provides one solution to this issue.

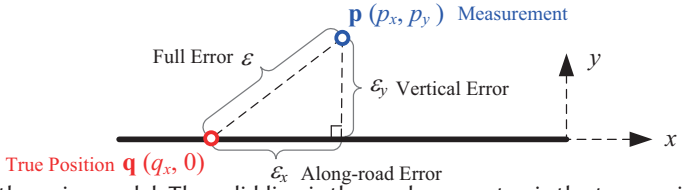


Fig. 6. The illustration of the noise model. The solid line is the road segment. $\mathbf{q}$ is the true position of the object which is restricted on the road and $\mathbf{p}$ is the noisy measurement.

Theorem 4.1. *If the noisy measurement* $\mathbf{p} = \begin{bmatrix} p_x \\ p_y \end{bmatrix}$ *is generated according to a 2-D Gaussian distribution* $\mathcal{N}\left(\begin{bmatrix} q_x \\ 0 \end{bmatrix}, \begin{bmatrix} \sigma_x^2 & \rho\sigma_x\sigma_y \\ \rho\sigma_x\sigma_y & \sigma_y^2 \end{bmatrix}\right)$*, suppose we have known the error* ε_y *in the y-direction of the measurement* $\mathbf{p}$*, the expectation of the squared full error of* $\mathbf{p}$*, i.e,* $\mathbb{E}[\varepsilon^2]$ *is* $\left(\left(\frac{\varepsilon_y^2}{\sigma_y^2} - 1\right)\rho^2 + 1\right)\sigma_x^2 + \varepsilon_y^2$.

Proof. See Appendix. □

4.3.2 Candidate Selection. According to Theorem 4.1, we understand that $\mathbb{E}[\varepsilon^2] = \left(\left(\frac{\varepsilon_y^2}{\sigma_y^2} - 1\right)\rho^2 + 1\right)\sigma_x^2 + \varepsilon_y^2$. That is to say, if ε_y is small, $\mathbb{E}[\varepsilon^2]$ will also be small and moreover, the along-road error ε_x will also be reduced if $\rho \neq 0$. Consequently, a good strategy to reduce the errors of $\mathbf{p}$ is to consider points in *original trajectory* $\mathcal{T}$ with small ε_y values (e.g., $p \in \mathcal{T}$ with corresponding ε_y smaller than a given threshold ζ). This selection strategy can help to limit the along-road error to an upper bound of $\sqrt{\left(\left(\frac{\zeta^2}{\sigma_y^2} - 1\right)\rho^2 + 1\right)\sigma_x^2}$.

Notice that selecting candidates strictly according to the threshold will not have any guarantee on the distance of two consecutive selected candidates, which means we may unfortunately select two candidates far away from each other and have to interpolate the points between these two candidates. Obviously, the longer the distance between them is, the larger the errors corresponding to those interpolated points will be. In order to address this issue, we propose a *soft candidate selection* strategy. Instead of only selecting points having $\varepsilon_y < \zeta$, we further consider the *distance between two consecutive candidates*. In detail, we design a *soft vertical error* ϵ_y instead of ε_y, where ϵ_y is computed as follows,

$$\epsilon_y = \varepsilon_y \times \left[1 - \frac{1}{1 + \exp(-\lambda(d - d_0))}\right]$$

Note, d refers to the distance between the currently scanned point and the previously selected candidate, and λ and d_0 are parameters. In addition, $\frac{1}{1+\exp(-\lambda(d-d_0))}$ is actually a logistic function $\ell(d)$ within the range $(0, 1)$, which can be used as a smoothed threshold function [8, 40] with d_0 being the threshold. $\ell(d)$ will be closer to 0 if $d < d_0$ and it will be closer to 1 if $d > d_0$. The parameter λ controls the steepness. As a summary, given a previously selected candidate c_i, our soft candidate selection scans and evaluates the points in original trajectory $\mathcal{T}$ right behind c_i one by one, until a new candidate is selected. For each point p_i that is evaluated, p_i is selected as a new candidate if the corresponding $\epsilon_y < \zeta$.

With the help of soft candidate selection, we can infer that when scanning the points close to c_i, their distances d to c_i will be small and hence $\ell(d) \approx 0$ and $\epsilon_y \approx \varepsilon_y$. In other words, when the distance between two candidates is not far (i.e., $d < d_0$), soft candidate selection strategy still safely implements the original selection criteria $\varepsilon_y < \zeta$. When the nearby points of c_i fail to become new candidates, the distance from next potential candidate to c_i (i.e., d) increases gradually, which results in the increase of $\ell(d)$ and hence $\epsilon_y < \varepsilon_y$ with their gap increased. Consequently, as d increases, the criterion of $\epsilon_y < \zeta$ becomes looser, and when d exceeds d_0, we have $\ell(d) \approx 1$ and hence $\epsilon_y \approx 0$ so a new candidate will be nearly unconditionally selected. The tolerance distance between two adjacent candidates is bounded by d_0 (200m in our experiments) which is controllable.

4.3.3 Bayesian Smoothing. In the previous section, we have obtained the candidate set $C = \{c_1 \rightarrow c_2 \rightarrow \cdots \rightarrow c_{|C|}\}$. C can be regarded as a sub-trajectory of the original trajectory $\mathcal{T}$ with the help of soft candidate selection strategy that enables us to select points with small expected error and hence help to reduce errors. However, there are still rooms for improvement. In the following, we introduce *Bayesian smoothing technique*, a smoothing technique on dynamical state space model, to further reduce the errors of the selected candidates C. A dynamical state space model captures a sequence of noisy measurements, where each measurement $\mathbf{c}_t$

is generated from a (unobserved) state $\mathbf{z}_t$ according to a distribution $P(\mathbf{c}_t|\mathbf{z}_t)$ that is usually called *emission probability*. The state dynamically evolves according to the previous state, i.e., the distribution $P(\mathbf{z}_t|\mathbf{z}_{t-1})$ which is the *transition probability*. The dynamical state space model is similar to HMM but the former is continuous while the state space of HMM is discrete.

Here, we adopt the Rauch-Tung-Striebel Smoother (RTSS) [29] which is the optimal Bayesian smoother with the closed form solution for the dynamical state space models when it is *linear Gaussian*, i.e., the state transition probability and the emission probability can be represented as

$$P(\mathbf{z}_t|\mathbf{z}_{t-1}) = \mathcal{N}(\mathbf{A}_t\mathbf{z}_{t-1} + \mathbf{B}_t, \mathbf{Q}_t)$$
$$P(\mathbf{c}_t|\mathbf{z}_t) = \mathcal{N}(\mathbf{H}_t\mathbf{z}_t + \mathbf{C}_t, \mathbf{R}_t)$$

Then, it can be proved that the marginal posterior of state $\mathbf{z}_t$ is also a Gaussian [29], i.e.,$(\mathbf{z}_t|\mathbf{p}_{1:T}) = \mathcal{N}(\mathbf{m}_t^s, \mathbf{P}_t^s)$. Thus, the key of RTSS is to compute the mean and covariance matrix of the posterior, i.e., $\mathbf{m}_t^s$ and $\mathbf{P}_t^s$ at each time step. This problem has a recursive closed-form solution which makes it optimal. The detail of RTSS can be found in [29].

Motion Model. The key of a dynamical state space model is to design a state transition model and a measurement model. Here, we include higher order information, i.e., defining the state $\mathbf{z}$ as the *coordinate* and the *velocity* in both x and y directions of the moving object. The measurement $\mathbf{c}$ is defined as the observed noisy position. Then we have,

$$\mathbf{z_t} = \left[q_x^{(t)}, q_y^{(t)}, v_x^{(t)}, v_y^{(t)}\right]^\top, \mathbf{p_t} = \left[c_x^{(t)}, c_y^{(t)}\right]^\top$$

Let $\mathbf{q_t} = \left[q_x^{(t)}, q_y^{(t)}\right]^\top$ and $\mathbf{v_t} = \left[v_x^{(t)}, v_y^{(t)}\right]^\top$. If we assume the object moves according to a constant speed $\mathbf{v}_t$ between two time steps, from the fact that $\mathbf{q}_t = \mathbf{q}_{t-1} + \delta_t\mathbf{v}_t$, where δ_t refers to the time interval between $\mathbf{z}_t$ and $\mathbf{z}_{t-1}$, we can induce the parameters $\mathbf{H}_t$, $\mathbf{C}_t$ and $\mathbf{Q}_t$ in transition model as follows,

$$\mathbf{H_t} = \begin{bmatrix} 1 & 0 & \delta_t & 0 \\ 0 & 1 & 0 & \delta_t \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},\ \mathbf{C}_t = 0,\ \mathbf{Q_t} = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{v_x}^2\delta_t & 0 \\ 0 & 0 & 0 & \sigma_{v_y}^2\delta_t \end{bmatrix} \tag{2}$$

Here, we assume the speed in each direction varies according to a 1-D Gaussian with the variance of $\sigma_{v_*}^2\delta_t$. Note the variance is multiplied with the time interval δ_t since the larger the δ_t is, the larger the speed will change. We also assume the measurement is generated according to the true position $\mathbf{q}_t$ with a zero-mean Gaussian distribution. Thus, we can derive the parameters $\mathbf{A}_t$, $\mathbf{B}_t$ and $\mathbf{R}_t$ of the measurement model as follows,

$$\mathbf{A}_t = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},\ \mathbf{B}_t = 0,\ \mathbf{R_t} = \begin{bmatrix} \sigma_{p_x}^2 & 0 \\ 0 & \sigma_{p_y}^2 \end{bmatrix}$$

For the two initial parameters $\mathbf{m}_0$ and $\mathbf{P}_0$, they are initialized as follows

$$\mathbf{m}_0 = \begin{bmatrix} c_x^{(1)} \\ c_y^{(1)} \\ 0 \\ 0 \end{bmatrix},\ \mathbf{P}_0 = \begin{bmatrix} \sigma_{p_x}^2 & 0 & 0 & 0 \\ 0 & \sigma_{p_y}^2 & 0 & 0 \\ 0 & 0 & \sigma_{v_x}^2 & 0 \\ 0 & 0 & 0 & \sigma_{v_y}^2 \end{bmatrix}$$

Now adopting the RTSS on the selected candidates $C = \{c_1 \to c_2 \to \cdots c_{|C|}\}$, which, according to the motion model defined above, form a sub-trajectory, will further reduce the error of C and generate the smoothed trajectory $\tilde{C} = \{\tilde{c}_1 \to \tilde{c}_2 \to \cdots \tilde{c}_{|C|}\}$.

Incorporating road network information. Note that the RTSS smoother is not aware of any road network information but does the smoothing according to the linear Gaussian motion model. It is not aware of the location of roads. We next propose a simple but effective technique to incorporate road network information. Before adopting RTSS on the candidates C, for each point $c_i \in C$, we are aware of the road segment that c_i is located on. Accordingly, we project each c_i onto its corresponding road segment and use the projection as its new measurement $\left[c_x'^{(t)}, c_y'^{(t)}\right]^\top$. In such a way, we feed the model the knowledge that an object moves on straight lines (since most, if not all, road segments are straight) which meets the linear assumption of the model. Thus, the model will tend to not smooth the point in the direction vertical to the road segment since it has already met the linear transition assumption. As a result, the model focuses more on smoothing the points in the direction *along* the straight line. We will demonstrate the effectiveness of this technique in our experimental study.

Discussion. Notice that in Section 4.1.3 we have proposed a weighted mean filter to smooth the trajectory, but we adopt RTSS in the candidate construction phase here. The reason is that in the candidate construction phase, each selected candidate point affects the accuracy of the final calibrated trajectory and if possible, we prefer each candidate to be as accurate as possible. However, as shown in Fig. 4, the weighted mean filter will tend to form a large curve when the trajectory is turning. This phenomenon will result in the large error of candidates. On the other hand, in the filtering phase, our objective is to generate a smoothed trajectory that can help to *obtain a route* $\mathcal{R}$ in the next route inference phase. In other words, how to guarantee the accuracy of each individual point is *not* the focus. That is the reason why the weighted mean filter is proper for the filtering phase but not the candidate selection phase. In addition, RTSS enables the incorporation of the latent relation between two points, i.e., higher order information such as the speed, making it more reasonable than the mean filter and having a higher chance to reduce the along-road error. As a result, we claim that the mean filter is not proper for this objective.

4.4 Interpolation Phase

After constructing the candidates $\tilde{C}$, in this phase, we calibrate the remaining points, i.e., $\mathcal{T} - \tilde{C}$, leveraging the confident information of the candidates. Thus, we apply an interval-based interpolation to obtain these remaining points. Although the measurements of positions are very noisy, there still remains the information relatively accurate which we can take advantage of, that is the *time stamp* of each trajectory point. It is reasonable to assume that in a short interval, each object moves at a constant speed. Consequently, we can infer that the moving distance between two true positions should be proportional to the time interval between them. To be more specific, let's consider interpolating the points between certain two consecutive candidates, say $\tilde{c_i}$ and $\tilde{c}_{i+1}$. Assume those points between $\tilde{c}_i$ and $\tilde{c}_{i+1}$ in the original trajectory which need to be calibrated are $p_{i,1}, p_{i,2}, \cdots, p_{i,k}$. Their calibrated positions, denoted as $\tilde{p}_{i,1}, \tilde{p}_{i,2}, \cdots, \tilde{p}_{i,k}$ have following property:

$$\Delta d_j = \frac{t_{i,j} - t_{i,j-1}}{t_{i+1,k+1} - t_{i,0}} L_i,\ j = 1, 2, \cdots, k$$

where Δd_j refers to the road network distance between $\tilde{p}_{i,j}$ and $\tilde{p}_{i,j-1}$. Specifically, for simplicity of representation, we rename $\tilde{c_i}$ to $\tilde{p}_{i,0}$ and $\tilde{c_{i+1}}$ to $\tilde{p}_{i,k+1}$. We denote the time stamp of each $p_{i,j}$ as $t_{i,j}$. Moreover, L_i is the road network distance between $\tilde{c}_i$ and $\tilde{c}_{i+1}$. Take Fig. 7 as an example. c_i and c_{i+1} are two consecutive candidate points selected by the candidate selection step, and $\tilde{c}_i$ and $\tilde{c}_{i+1}$ are the corresponding smoothed candidate positions smoothed by Bayesian filter process. $p_{i,1} \sim p_{i,4}$ are the points in the original trajectory between c_i and c_{i+1} which are going to be interpolated for calibration. From the time stamps of these points, we know the intervals between two consecutive points are 5 seconds. Accordingly, the position of the interpolated position of the first point $p_{i,1}$ should go along the corresponding route starting from $\tilde{c}_i$ by a distance of $\Delta d_1 = \frac{t_{i,1} - t_{i,0}}{t_{i,5} - t_{i,0}} L_i = \frac{5}{25} L_i$; the position of the interpolated position of the second point $p_{i,2}$ should go along the route starting from $\tilde{p}_{i,1}$ by a distance of

ALGORITHM 1: CLSTERS

Input: Noisy Trajectory $\mathcal{T} = \{p_1, p_2, \cdots, p_N\}$ and the road network G.
Output: The calibrated trajectory $\tilde{\mathcal{T}}$.

1 $\hat{\mathcal{T}} \leftarrow speed\_filter(\mathcal{T})$
2 $\hat{\mathcal{T}} \leftarrow angle\_filter(\hat{\mathcal{T}})$
3 $\hat{\mathcal{T}} \leftarrow weighted\_mean\_filter(\hat{\mathcal{T}})$
4 $R \leftarrow map\_matching(\hat{\mathcal{T}}, G)$
5 **for** i *from* 1 *to* $|\hat{\mathcal{T}}| - 1$ **do**
6 $\quad \mathcal{R}_i \leftarrow route\_completion(R[i], R[i+1], G)$
7 $\quad \mathcal{R} \leftarrow \mathcal{R} \cup \mathcal{R}_i$
8 **end**
9 $C \leftarrow candidate\_selection(\hat{\mathcal{T}}, \mathcal{R}, G)$
10 $\tilde{C} \leftarrow Bayesian\_smoothing(C, \mathcal{R}, G)$
11 $\tilde{\mathcal{T}} \leftarrow interpolation(\tilde{C}, \mathcal{T} - \tilde{C}, \mathcal{R}, G)$
12 **return** $\tilde{\mathcal{T}}$

$\Delta d_2 = \frac{t_{i,2}-t_{i,1}}{t_{i,5}-t_{i,0}} L_i = \frac{5}{25} L_i$. Finally, it holds that

$$\Delta d_1 : \Delta d_2 : \Delta d_3 : \Delta d_4 : \Delta d_5 = (t_{i,1} - t_{i,0}) : (t_{i,2} - t_{i,1}) : (t_{i,3} - t_{i,2}) : (t_{i,4} - t_{i,3}) : (t_{i,5} - t_{i,4})$$

$$L_i = \Delta d_1 + \Delta d_2 + \Delta d_3 + \Delta d_4 + \Delta d_5$$

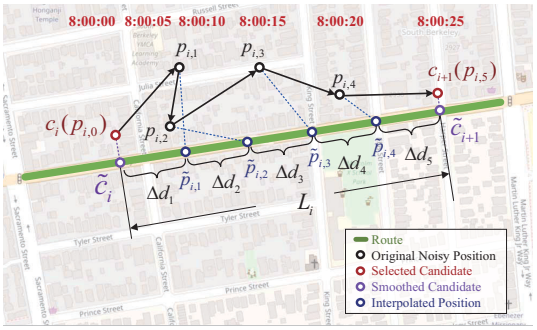


Fig. 7. Example of the interval-based interpolation. The time stamps corresponding to trajectory points are listed on the top of the figure.

4.5 Summary

Till now, we have presented the details of CLSTERS. As a conclusion and to provide a clearer view, we list the complete flow of CLSTERS in Algorithm 1. Note lines 1 to 3 refer to the filtering phase, lines 4 to 8 refer to the map matching phase, lines 9 to 10 refer to the candidate construction phase, and line 11 is the interpolation phase.

5 EXPERIMENT

In our experimental study, we use three noisy cellular-based trajectory datasets collected in Xi'an and Beijing in China as well as Kuwait City in Kuwait. The statistics of those three datasets are listed in Table 1 and their visualization is plotted in Fig. 8(a), 8(b) and 8(c). These three trajectories have different characteristics. E.g., for

Table 1. Dataset Description

	Xi'an	Beijing	Kuwait City
# Points	10018	1338	1188
# Road Segments Passed	276	151	110
Duration	3h13m49s	1h19m	28m28s
Sampling Interval (s)	1.16	3.57	1.44
Length (km)	35.96	36.25	21.19
Average Speed (km/h)	10.81	27.33	45.20
Original Localization Error (m)	82.86	212.95	167.57
Moving Style	covering road network	covering road network	normal driving
Cellular Network	4G (LTE)	3G (UMTS)	4G (LTE)
Raw Signal	RSRP	Cell-ID	RSRP
Localization Technique	RFPM[34]	WCCL[30]	RFPM[34]

the driving style, the trajectories in the Xi'an and Beijing datasets are generated by covering the road network in corresponding area for testing the robustness of error reduction algorithms under different road types as well as driving route complexity while the Kuwait dataset is generated by normal driving style. For the error scales, the Beijing and Kuwait datasets have relative large error scales while the Xi'an dataset has the normal error scale (i.e., around 100m). Moreover, to further justify the generalization capability of CLSTERS, these three datasets are collected via two different localization techniques with different raw signal and cellular networks. In detail, the raw signals of Xi'an and Kuwait are collected in 4G (LTE) network in the form of received signal code power (RSCP) and are localized to point positions by radio frequency pattern matching (RFPM) technique [34]. The raw signals of Beijing dataset are collected via 3G (UMTS) in the form of raw Cell-ID, and the corresponding localization technique used is weighted centroid correction localization (WCCL) [30]. Notice that the data which will be used in the following experiments only contains the latitudes and the longitudes generated by the localization techniques, as well as the timestamps.

To get the trajectory under different sampling rates, we sub-sample the original trajectory according to sub-sampling rate s, i.e., to pick up $p_1, p_{1+s}, p_{1+2s}, \cdots$ from the raw trajectory to form the new trajectory, with s being 1, 2, $\cdots$, 10. When $s = 1$, all the raw points are considered, while $s = 10$, only 10% of the original raw points are considered. Note that for Beijing dataset, when the data is down-sampled to 10%, only about 100 points are available. In other words, the down-sampled trajectory with 100 points loses too much information, which makes it impossible to be recovered by any algorithm, given such a complex trajectory. Consequently, for Beijing dataset, we specifically set the max sub-sampling rate s to 5. For the ground truth, when we get a cellular-based trajectory position, we simultaneously get a sample by AGPS-localization. Since the GPS error is very low, we adopt the map matching algorithm to project each GPS sample onto the corresponding road segment and use the projection as the ground truth position of each sample. We adopt the calibration error defined in Section 3 as the main performance metric.

We implement following approaches as the representatives of the state-of-the-art approaches.

SnapNet. The only map matching approach [25] that is designed for cellular-based trajectories.

HMM-MM. Hidden Markov model-based map matching algorithm, the well-known map matching approach proposed in [26] which is regarded as the representative of map matching techniques.

RTSS. The Rauch-Tung-Striebel Smoother [29] which has been introduced in Section 4.3.3. Since the smoother is designed for smoothing the trajectory and reducing the noise, we can directly adopt it on the original noisy trajectory to study whether this approach is workable.

(a) Xi'an: raw data & grount truth (b) Beijing: raw data & grount truth (c) Kuwait City: raw data & grount truth

(d) Xi'an: the calibrated result of CLSTERS (e) Beijing: the calibrated result of CLSTERS (f) Kuwait City: the calibrated result of CLSTERS

Fig. 8. The visualization of the three datasets. Trajectories in red are generated by cellular-based localizations. Those in purple are generated by GPS and mapped to the road network which can be regarded as the ground truth and those in blue are the calibrated trajectories output by CLSTERS. Note that the measure scales of these maps are different.

RTSS+PJ. RTSS is adopted to smooth the original trajectory and then each smoothed point is projected onto the nearest road as the final calibrated position.

KF. The Kalman Filter [17] is the optimal solution for Bayesian filter [31] and commonly used for locating the outliers and smoothing trajectories [43]. The difference between KF and RTSS is that KF is a Bayesian filter which can not get the observations after currently filtering time step t while RTSS can use the observations after t.

KF+PJ. Similar to RTSS+PJ, it projects the filtered point onto the nearest road as the final position.

MF. The mean filter [43], introduced in Section 4.1.3, is common for smoothing the trajectory.

MF+PJ. Similarly, it projects the point after adopting MF onto the nearest road as the final position.

PJ. This is the naive solution. It just projects each original noisy trajectory point onto the nearest road segment.

5.1 Overall Evaluation

5.1.1 The Main Evaluation. The main objective of CLSTERS is to reduce errors for cellular-based trajectories and we plot the CDF of the calibration error w.r.t. the original localization error as shown in Fig. 9(a), Fig. 9(b) and Fig. 9(c). We can see that our error reduction system does help to reduce the error of trajectories with large error scales. Moreover, the visualizations of the calibrated trajectory shown in Fig. 8(d), Fig. 8(e) and Fig. 8(f) further justify the effectiveness of our approach. Next, we perform two sets of experiments to demonstrate the superior performance of CLSTERS, as compared with its competitors. Note that all the parameters of existing algorithms are optimized to achieve the best performance. Similarly, we also fine-tune the parameters of CLSTERS.

The generality of parameter settings will be evaluated in the next subsection and the impacts of parameters in CLSTERS will be explained in Section 5.2.

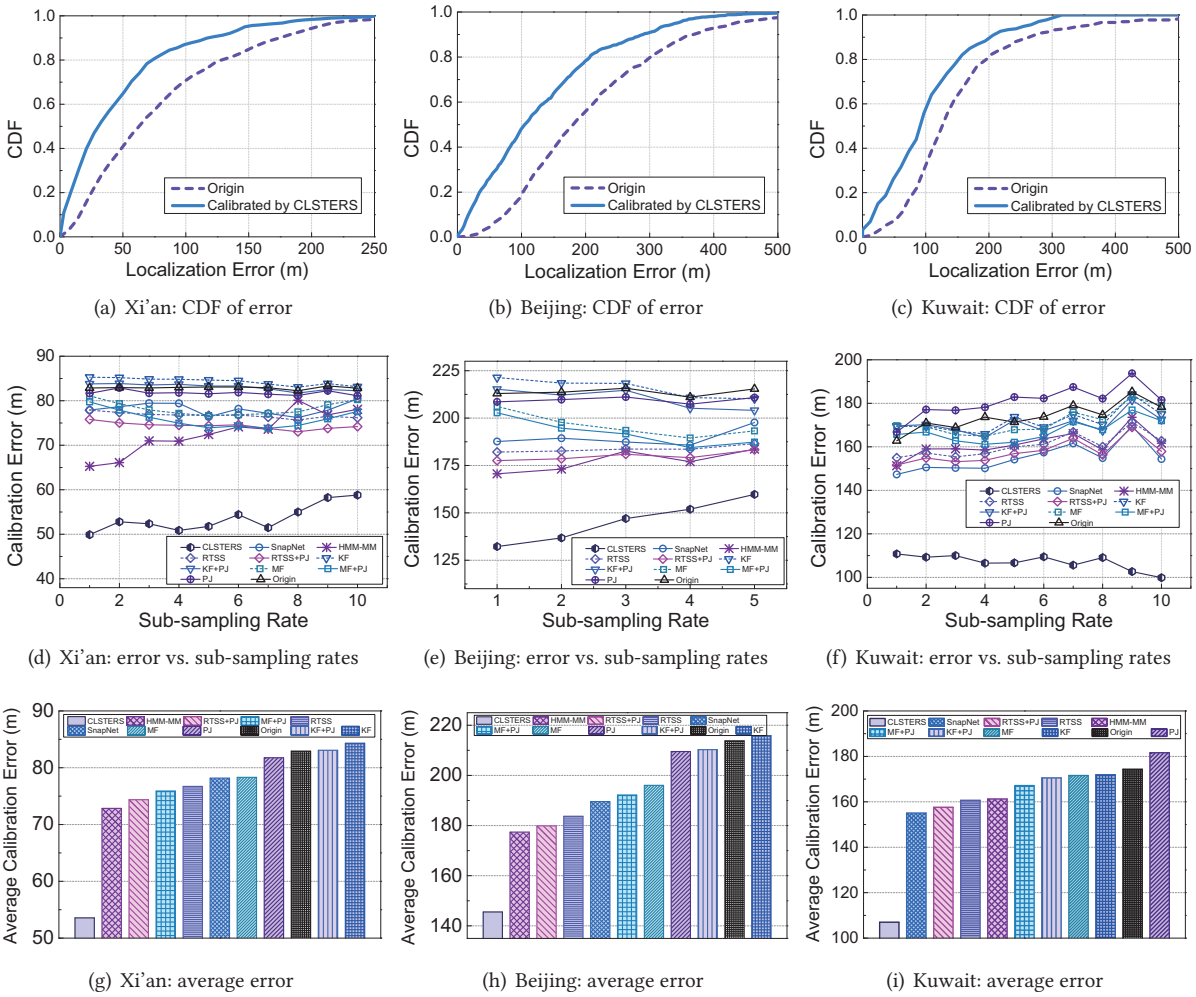
The first set of experiments is evaluating the calibration error of these approaches which is the key objective of this paper. Fig. 9(d), Fig. 9(e) and Fig. 9(f) plot the results w.r.t. different sub-sampling rate corresponding to three datasets. From the results we can see that CLSTERS has successfully reduced about 40% error while other approaches have very subtle effects. This effectively demonstrates the generality and robustness of CLSTERS under different datasets with different driving styles and error scales. Moreover, we can see that CLSTERS achieves the best performance under different trajectory data localized by different localization techniques, which further justifies the transparency between CLSTERS and localization techniques. Although the errors of CLSTERS in the Xi'an and Beijing datasets increase slightly when the sub-sampling rate becomes high, we claim that it is reasonable, because the trajectory in the Xi'an and Beijing datasets is much more complicated than that in Kuwait. As a result, when the trajectory is sub-sampled and becomes sparse (e.g., when s is set to a large value), it loses some information that is important for error reduction, while the Kuwait trajectory does not include many turnings which makes our approach relatively stable under different sub-sampling rates.

To make things clearer, we average the results under different sub-sampling rates and report the average calibration errors in the ascending order in Fig. 9(g), Fig. 9(h) and Fig. 9(i). From the results we can infer some facts. First, among the three filter/smoothers we evaluated (i.e., RTSS, MF, and KF), RTSS performs the best and KF performs the worst, i.e., i.e., $\varepsilon_{RTSS} < \varepsilon_{MF} < \varepsilon_{KF}$. RTSS is the optimal Bayesian smoother and it has both "optimal" and "looking forward" features; while MF only has the "looking forward" property. This explains why RTSS outperforms MF in all three datasets. KF is an optimal Bayesian filter. Unlike Bayesian smoother, it can only look backward thus it has no information after current time stamp to refer to, which makes it worse than MF. Second, for RTSS, KF and MF, their "+PJ" versions all perform slightly better than the original versions which is consistent with our expectation. This is because "+PJ" version actually incorporates more information and some prior knowledge (e.g., moving restricted by the road network), and hence it will have some chances to get rid of vertical errors and to outperform original versions. Moreover, among all the existing approaches we implemented, we can observe that HMM-MM or SnapNet (both are map matching-based approaches) achieves the best performance, which further proves that road network information is important and very useful. Last but not least, we want to highlight that RTSS, HMM-MM, PJ, MF approaches are actually adopted by CLSTERS in certain steps. The fact that CLSTERS significantly outperforms all existing approaches confirms that the problem studied in this paper is not easy, and it cannot be solved by adopting any existing approach solely; however, fully understanding the problem and the power of existing approaches and combining them in the right way is the key to conquer this problem.

Since our approach can also recover the route of a trajectory, we are also interested in figuring out the performance of inferring the route. Hence, we conduct the second set of experiments to evaluate the route accuracy of these approaches. Let $\mathcal{R}_{gt}$ denote the ground truth route, and $\mathcal{R}$ denote the route inferred by an approach, the route accuracy $\alpha_{\mathcal{R}}$ is computed as

$$\alpha_{\mathcal{R}} = \frac{len(\mathcal{R} \cap \mathcal{R}_{gt})}{\max(len(\mathcal{R}), len(\mathcal{R}_{gt}))} \in [0, 1]$$

The maximum operation is adopted to penalize the algorithm if it returns a route by covering as many roads as possible, which is a commonly used criterion [38, 42]. Notice that KF, MF and RTSS are excluded from this evaluation as they do not use road network information. Alternatively, we use their "+PJ" versions to get the route by connecting the road segments that each point is projected to. To get the entire route, we complete the route according to Section 4.2 by inserting the shortest path between two unadjacent road segments. For the case that a point p_i is projected to a bi-directional road segment (i.e., two road segments share the exact shape but opposite directions), we select the one whose direction is closer to $p_{i-1} \rightarrow p_i$.



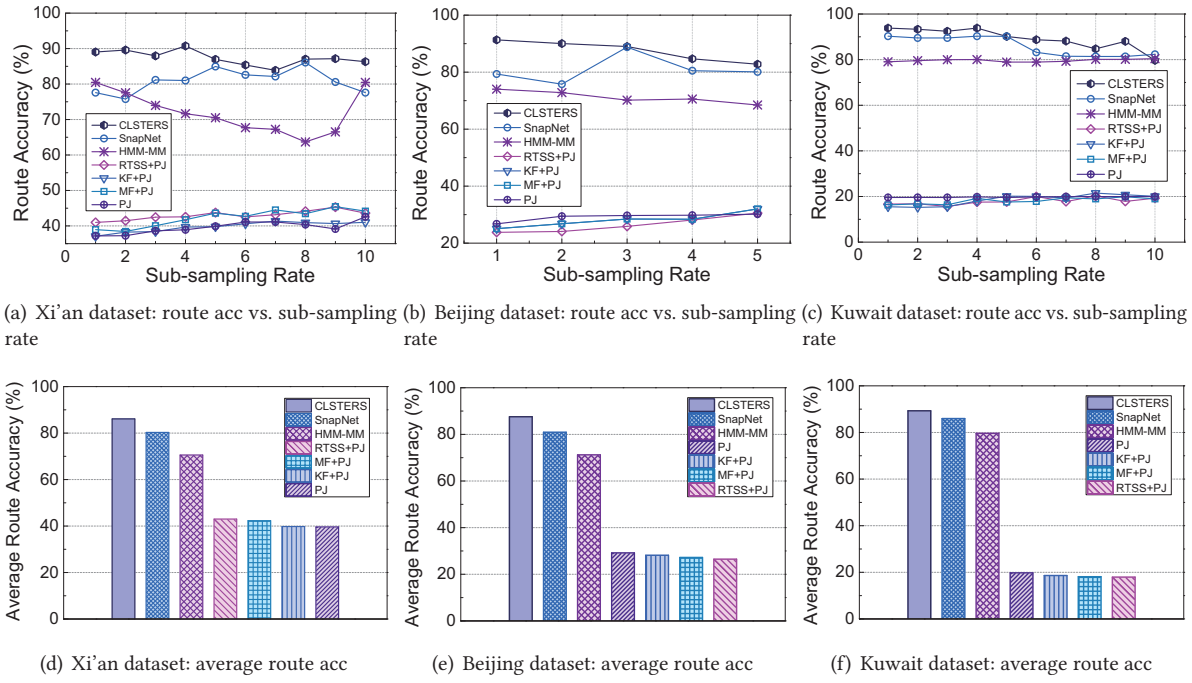
(a) Xi'an: CDF of error (b) Beijing: CDF of error (c) Kuwait: CDF of error

(d) Xi'an: error vs. sub-sampling rates (e) Beijing: error vs. sub-sampling rates (f) Kuwait: error vs. sub-sampling rates

(g) Xi'an: average error (h) Beijing: average error (i) Kuwait: average error

Fig. 9. The performance of calibration error of all approaches. "Origin" refers to the error of the original trajectory. The sub-figures in the last column are the results averaged by sampling rates and the results are sorted in the ascending order.

Fig. 10(a), Fig. 10(b) and Fig. 10(c) plot the results. We can infer that non-map-matching-based approaches, i.e., RTSS+PJ, KF+PJ, MF+PJ and PJ, perform worser than map-matching-based approaches. This is because most of roads are bi-directional [39] and the trajectories have many "ping-pong effects" and outliers which may result in the wrong selection of the road segment if only considering the direction of $p_{i-1} \rightarrow p_i$. Among those map-matching-based approaches, CLSTERS outperforms other approaches in most cases, while SnapNet is the second best approach which is superior to HMM-MM. Since SnapNet is also designed for cellular-based trajectories, its route accuracy is better than that of HMM-MM. However, according to the calibration error reported previously, we observe that SnapNet is not able to reduce/correct the along-road error. On the other hand, HMM-MM based approach cannot handle large error trajectory, so it has the lowest accuracy among CLSTERS, SnapNet and HMM-MM. We also average the performance of different approaches by averaging all sub-sampling rates and report the results in the descending order in Fig. 10(d), Fig. 10(e) and Fig. 10(f). We observe



(a) Xi'an dataset: route acc vs. sub-sampling rate (b) Beijing dataset: route acc vs. sub-sampling rate (c) Kuwait dataset: route acc vs. sub-sampling rate

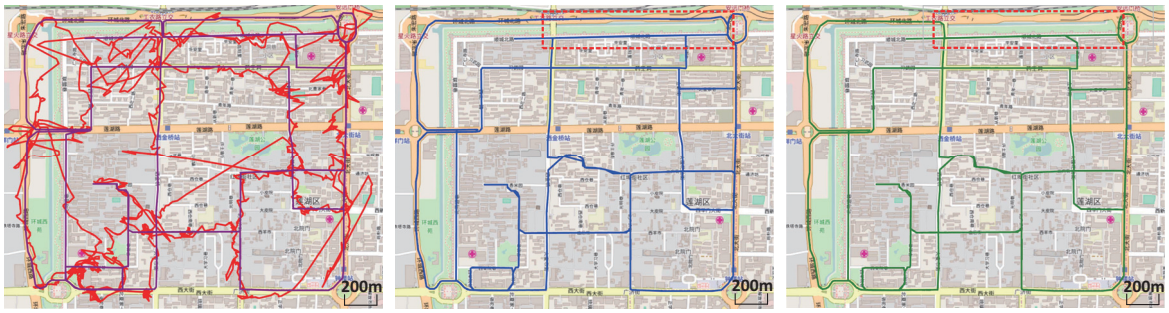
(d) Xi'an dataset: average route acc (e) Beijing dataset: average route acc (f) Kuwait dataset: average route acc

Fig. 10. The performance of route accuracy of all approaches. The sub-figures in the row below are the results averaged by sampling rates and the results are sorted in the descending order.

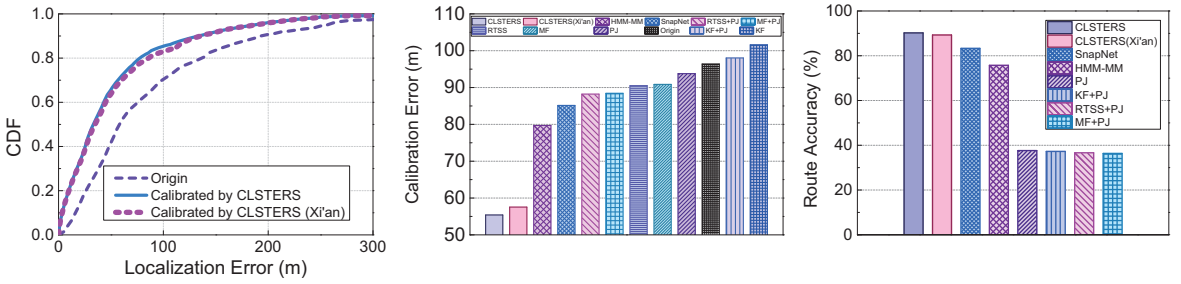
that for filtering/smoothing approaches (those with PJ), their performances are similar and they only achieve around 20% to 40% route accuracy while those map-matching-based approaches are able to achieve a much higher accuracy.

5.1.2 Evaluation on Generality of Parameter Settings. Recall that in the previous set of experiments, the parameters of all approaches (including CLSTERS) are tuned to reach the best performance. We conduct an additional experiment showing the generality of parameter settings of CLSTERS on unknown trajectories. We collected another trajectory in Xi'an, denoted as Xi'an-2. It has a length of 24.5km containing 5,608 points with the original localization error of 96.4m, similar to the origin Xi'an dataset. The visualization of the dataset Xi'an-2 can be found in Fig. 11(a). In terms of CLSTERS, we report its performances on Xi'an-2 under two different parameter settings, denoted as CLSTERS and CLSTERS(Xi'an). CLSTERS refers to the optimal performance where parameters are tuned based on the underlying dataset (i.e., Xi'an-2 dataset); while CLSTERS(Xi'an) refers to performance of CLSTERS on Xi'an-2 dataset with the parameters tuned based on Xi'an dataset. We then report the calibration error and the route accuracy metric in Fig. 11(e) and Fig. 11(f).

From Fig. 11(e) and Fig. 11(f), we can observe that the performances of all approaches remain, as compared with their performance under Xi'an dataset. CLSTERS still achieves the best performance in terms of both calibration error and route accuracy metrics. Although CLSTERS(Xi'an) performs slightly below CLSTERS, the gap is insignificant, as compared with the performance of other approaches. This finding demonstrates that the performance of CLSTERS is not very sensitive to the parameter settings; and the parameters tuned based on one trajectory could be used for another trajectory corresponding to the movement in the same road network. The CDF between the calibrated trajectory via CLSTERS and that via CLSTERS(Xi'an), as reported in Fig. 11(d), double confirms our finding, as their difference is almost negligible.



(a) raw data & grount truth (b) The calibrated result under CLSTERS setting (c) The calibrated result under CLSTERS(Xi'an) setting



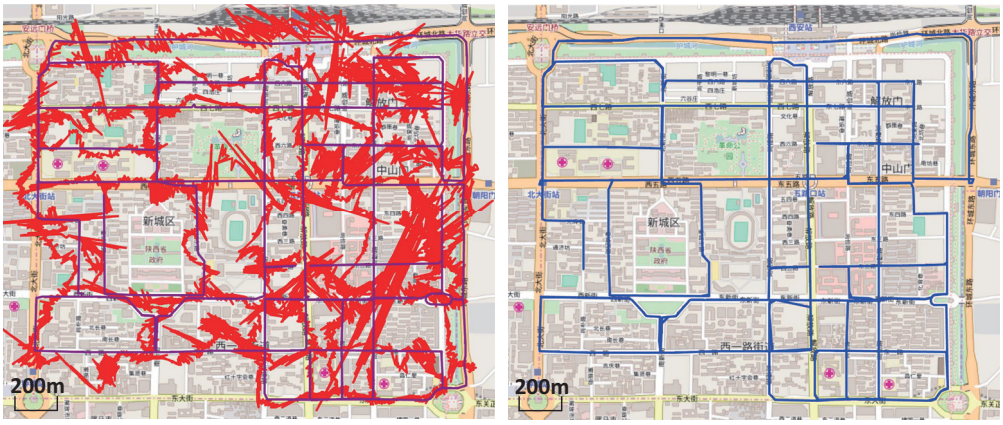
(d) CDF of error (e) Calibration error (f) Route accuracy

Fig. 11. The visualization and performance on dataset Xi'an-2. CLSTERS refers to performace where parameters are tuned based on Xi'an-2 dataset, and CLSTERS(Xi'an) refers to the performance of CLSTERS with parameters tuned based on original Xi'an dataset.

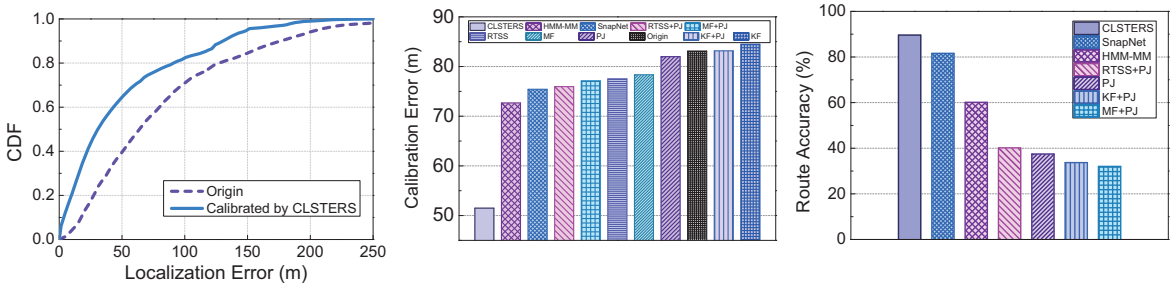
5.1.3 Evaluation on Different Movement Speeds. Intuitively, the error scale, the driving style, the localization technique, the sampling rate and the moving speed may effect the calibration result which have been evaluated previously. Among these factors, we are interested in moving speed which can largely influence the shape of a trajectory and perhaps may affect the performance of calibration algorithms. Although we have reported the performance of all the algorithms under different moving speeds, i.e., Xi'an dataset corresponds to a speed of 10 km/h, Beijing dataset corresponds to a speed of 27 km/h, and Kuwait dataset corresponds to a speed of 45 km/h, these speeds are still in the valid range of vehicle velocity. From Fig. 8, we can infer that the lower the speed is, the more chaotic and messier the trajectory will be, i.e., having more "ping-pong" effects. Consequently, it becomes interesting to evaluate how CLSTERS performs when the speed becomes *extremely low*, e.g., under the pedestrian scenario. Thus, in this set of experiments, we conduct the evaluation on calibrating trajectories moving at a very low speed by creating an artificial dataset to test movement speeds resembling that of pedestrians.

Suppose the driving speed is v_d and the required walking speed is v_w. We first modify the time stamp of each existing point in the vehicle trajectory $\mathcal{T}_d$ to "slow down" the trajectory. In detail, for two consecutive points $p_1, p_2 \in \mathcal{T}_d$ with the time stamps t_1 and t_2, we increase the duration between t_1 and t_2 by the ratio of $\frac{v_d}{v_w}$, i.e., having $t_2' = t_1 + \frac{v_d}{v_w}(t_2 - t_1)$. In such a way, we can ensure that the moving speed is slowed down to the required walking speed. Since we want to get a pedestrian trajectory having the same sampling interval as the original vehicle trajectory, we need to interpolate $\lceil \frac{v_d}{v_w} \rceil - 1$ points between p_1 and p_2 to make sure the sampling interval does meet the requirement. For each interpolated point p_i, its ground truth can be simply obtained via linear interpolation along the route passing from p_1 to p_2. After deciding the ground truth of p_i, its measurement can

be simulated by generating an error vector ε from a 2-D Gaussian distribution and getting the measurement by moving the ground truth position along the generated error vector. Note that the mean and co-variance matrix of Gaussian are obtained by estimating a small window (11 points in the experiment) of original vehicle trajectory to ensure the locality of the noise.



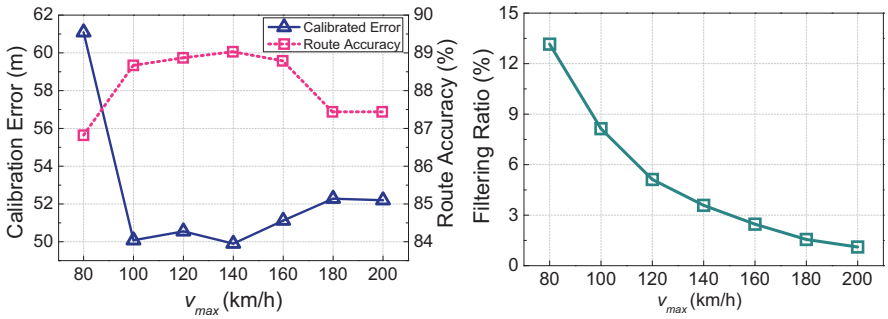
(a) Simulated pedestrian trajectory (b) The calibrated result of CLSTERS



(c) The CDF of error (d) Calibration error (e) Route accuracy

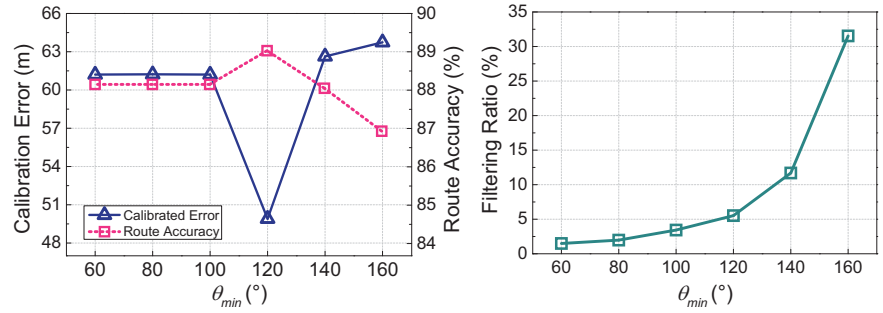
Fig. 12. The visualization of simulated pedestrian trajectory using the Xi'an dataset and the calibrated results as well as the quantified results.

Fig. 12(a) plots the simulated pedestrian trajectory with walking speed set at 5km/h and sampling interval at 1s. We can find that the trajectory becomes extremely messy since the moving speed is slowed down and more points are generated which bring more noises. Fig. 12(b) shows the calibrated results of CLSTERS that has successfully calibrated most points even in such a challenging and messy scenario. Fig. 12(c) shows the CDF of error on calibrated trajectory and we can find that CLSTERS does effectively reduce many errors in the original trajectory. Moreover, we test the performance of other competitors, with the results reported in Fig. 12(d) and Fig. 12(e). From the result, we can observe that CLSTERS again achieves the best performance among all approaches with significant performance advantage. It's worth to mention that, in terms of route accuracy, SnapNet outperforms HMM-MM largely. This is because that SnapNet also adopts a filtering process which can filter many noisy points in the original trajectory and enables map matching to perform better. On the other hand, although SnapNet achieves the highest route accuracy among all competitors, its calibration error is still high due to the fact that it is not able to reduce the along-road errors.



(a) Performance vs. v_{max} (b) Filtering ratio vs. v_{max}

Fig. 13. Experimental results on the speed filter.



(a) Performance vs. θ_{min} (b) Filtering ratio vs. θ_{min}

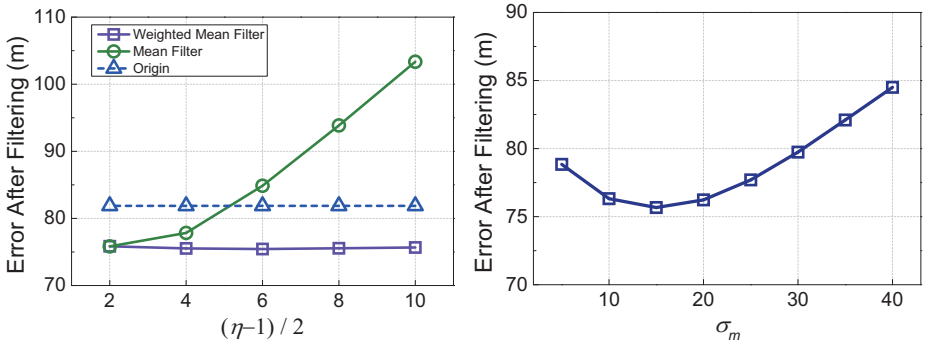
Fig. 14. Experimental results on the angle filter.

5.2 Self Experiments

Recall that although we have tried our best to make CLSTERS as general as possible, parameter tuning is still unavoidable for datasets of different cities since there are multiple factors influencing the scale of parameters such as the error scale, the driving speed, the severity of "ping-pong effect", etc. However, we claim that although the scale of the parameters may change under different datasets, there is no need for our system to change its structure. In the following sets of experiments, we only report the results based on the Xi'an dataset when studying the *impact* of parameters on the performance of CLSTERS, as the Xi'an trajectory is a road-network-covering trajectory, i.e., containing left-turn, right-turn, u-turn as well as covering many types of roads, and the results obtained from Xi'an dataset are representative. Note that by default, all the parameters are optimized to enable CLSTERS to achieve its best performance. When we study the effect of parameters, we only vary one parameter's value in one experiment and fix all the other parameters to their defaults.

5.2.1 Experiments on Speed Filter. First, we study the effect of the threshold parameter v_{max} used by the speed filter introduced in Section 4.1.1, with results shown in Fig. 13(a). We can observe that when v_{max} is set to a small value (e.g., 80km/h), over 13% points are filtered among which some are normal points that should not be filtered out according to Fig. 13(b). The over-filter phenomenon actually explains the relatively large calibration error incurred by CLSTERS when v_{max} =80km/h. On the other hand, when v_{max} is set to a large value, the filtering power weakens, thus the performance drops and will be finally converged to the result without using the filter. Considering Xi'an dataset, the best value of v_{max} is 120km/h which is consistent with common knowledge since the speed is often limited to 120km/h in the real world.

5.2.2 Experiments on Angle Filter. Second, we study the effect of the threshold parameter θ_{min} used by the angle filter introduced in Section 4.1.2. We vary θ_{min} from 60° to 160° and plot the trend of calibration error and route accuracy under different θ_{min} values in Fig. 14(a). We can observe that CLSTERS reaches the best performance when $\theta_{min} = 120°$. This observation indicates that preserving the points forming a near-straight line will be beneficial to the result. However, when θ_{min} increases its value further, the performance drops. The reason is that the further increase of θ_{min} will make the filtering criteria more strict and hence more points will be filtered out as shown in Fig. 14(b). Over-filtering may trigger the issue of filtering out a series of consecutive points and hence some key information will be lost and difficulty of recovering the route will be increased. On the other hand, when θ_{min} decreases its value, the power of the filter will also be decreased and less outliers will be filtered out which also results in the decrease of the performance.



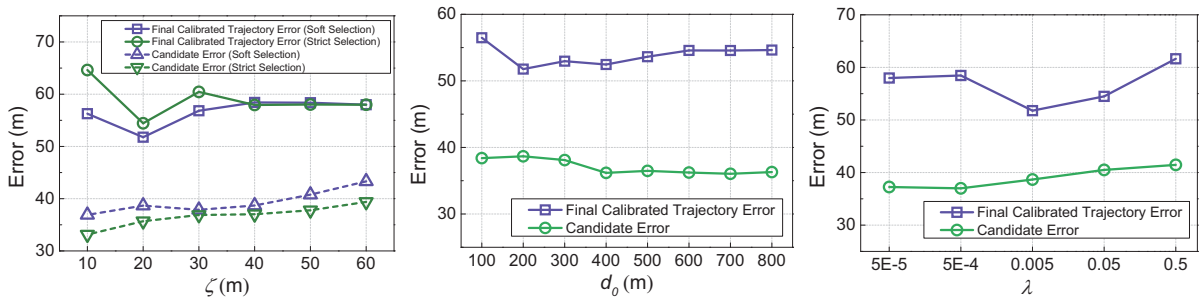
(a) Error vs. half window size (b) Error vs. σ_m

Fig. 15. Experimental results on the weighted mean filter.

5.2.3 Experiments on Weighted Mean Filter. Next, we study the parameters of the weighted mean filter. Recall that it relies on two key parameters, the window size η and the standard deviation σ_m of the Gaussian weight function in Eq. (1). We first conduct the experiments to study the impact of window size η by varying the half window size $(\eta - 1)/2$ from 2 to 10. For a better visualization of the effect of this parameter, we use the Xi'an dataset under the sub-sampling rate 5 (i.e., $s = 5$) to sparse the trajectory. Since this approach is fixing the position and making the trajectory smoother, we use the *error of the filtered trajectory* to directly analyze the performance of the filter. For comparison, we also plot the performance under the traditional mean filter in Fig. 15(a). From the results we can observe that the weighed mean filter shows the strong robustness to the window size. The traditional mean filter has a similar performance when the half window size is 2, however when the window size increases, the error drastically increases. Thus with the help of the weighted mean filter, we can safely set the window size to a large value for better filtering the densely sampled trajectory while maintaining the performance on the sparsely sampled trajectory, which has also been proved in Fig. 4.

For the parameter of σ_m, we vary it from 5 to 40 and we find the proper value of σ_m is in the range of 10 to 20 according to Fig. 15(b). If σ_m is too small, the Gaussian function becomes very sharp which means it only assigns a high weight to the point very close to the current point and hence weakens the effects of the filter with increased error. If σ_m is set to a large value, the Gaussian function becomes flat which means it assigns a high weight to the point sampled far away from the current point. Then if the window is set to a large value too, this will make the filter over-filter the points and force the filter to perform similarly as the traditional mean filter which will increase the error.

5.2.4 Experiments on Candidate Selection. Fourth, we study the candidate selection process. Recall that the parameter ζ serves as the threshold for selecting the candidates whose vertical error is smaller than ζ. Fig. 16(a)



(a) Effects of soft candidate selection strategy (b) Effects vs. d_0 (c) Effects vs. λ

Fig. 16. Experimental results on the candidate selection.

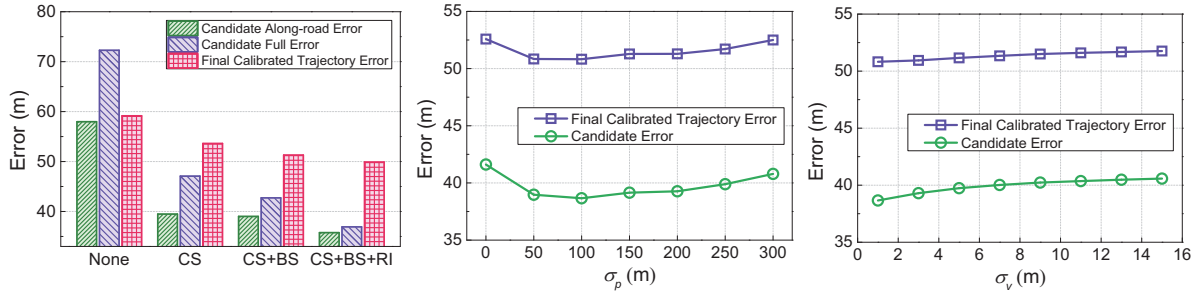
shows the result by varying ζ from 10m to 60m. Looking at the result of the strict selection criteria, we see that the error of the candidate reduces with the decrease of ζ, which is consistent with Theorem 4.1 showing that under this strategy we do have the ability to control the error of the selected candidates. As ζ limits the vertical error ε_y of the candidates, reducing ε_y will result in the decrease of the error ε as well as the along-road error ε_x. Based on the trend of the final calibrated trajectory error, we can figure out that although we prefer the small error of candidates, the performance will reversely be worse if ζ becomes too small. The reason is that when we set ζ too small, for strict selection criteria, there will be fewer points to be selected as the candidates. This means we have to interpolate more points with fewer candidates to be referred to, which will increase the final calibration error. When ζ is set to a larger value, the selection criteria will be loosened and more points will be selected. As a result, fewer points will be interpolated and the result will converge to the result without interpolation. This explains the convergence of the final calibrated error in Fig. 16(a) when $\zeta > 40$m.

Fig. 16(a) also plots the performance of the proposed soft candidate selection strategy. Since it will loosen the criteria by considering the distance between two candidates, the error of the selected candidates will be larger than the error of the candidates selected by the strict criteria, which is reflected in Fig. 16(a). However, thanks to the soft criterion that ensures the distance of two candidates to be bounded, we can set ζ to a small value without worrying about the potential risk of having the number of selected candidates being too small. As shown in the figure, the final calibration error curve of soft selection is much gentler and lower than the strict one which justifies the effectiveness of the soft candidate selection strategy.

Recall that in the soft candidate selection strategy, two parameters are involved, i.e., the candidate distance threshold d_0 and the logistic function's steepness parameter λ. Based on the results plotted in Fig. 16(b), we see that the performance reaches its best values when $d_0 = 200$m. When d_0 further decreases its value, it tends to easily loosen the criteria, which results in accepting points with larger errors. Thus, the candidate error increases as shown in the figure, together with the increase of the final calibration error. On the other hand, when d_0 increases its value, the criterion becomes more strict and it tends to perform similary as the strict selection.

Fig. 16(c) plots the effect of the steepness parameter λ. We can infer that a larger λ will result in a gentler logistic function $\ell(d)$ which tends to easily loosen the criteria. Thus, the performance drops with the increase of λ. On the other hand, when λ is set to a smaller value, $\ell(d)$ will tend to be the hard threshold which means if d is smaller than d_0 it will perform as the strict selection criterion and when d exceeds d_0, $1 - \ell(d)$ will suddenly drop to 0 which losses the smooth property of logistic function and hurts the performance.

5.2.5 Experiments on Bayesian Smoother. Last but not least, we study the Bayesian Smoother. We compare the performance under four strategies. They are (1) Directly using the purified trajectory $\hat{\mathcal{T}}$ after the filtering phase as the candidates and no Bayesian smoothing technique is adopted, which is denoted as None; (2) Adopting candidate



(a) Effects of candidate construction strategies (b) Effects vs. σ_p (c) Effects vs. σ_v

Fig. 17. Experimental results on Bayesian smoothing.

selection strategy introduced in Section 4.3.2 without smoothing, denoted as CS; (3) Adopting both the candidate selection strategy and the Bayesian smoother, i.e., RTSS introduced in Section 4.3.3, to smooth the candidates, without incorporating road network information into the model, denoted as CS+BS; (4) Adopting candidate selection strategy and using RTSS with incorporating road network information, denoted as CS+BS+RI. Fig. 17(a) illustrates the results. For the "None" approach, it does not select the candidate and hence loses the control of the error of candidates, resulting in the worst performance. CS approach effectively reduces the along-road error of the candidates, as compared with "None", showing the effectiveness of the candidate selection strategy and the correctness of our theorem. For CS+BS approach, it further reduces the error w.r.t. CS, demonstrating the necessity of Bayesian smoothing process. Furthermore, with road network information incorporated into RTSS, CS+BS+RI performs the best.

Next, we conduct the experiments on the parameters involved in RTSS. Note that in the motion model introduced in Section 4.3.3, the standard deviation of the measurement model σ_{p_x}, σ_{p_y} and the standard deviation of the transition of speed σ_{v_x}, σ_{v_y} are four parameters in RTSS. To simplify our evaluation, we assume $\sigma_{p_x} = \sigma_{p_y} = \sigma_p$ and $\sigma_{v_x} = \sigma_{v_y} = \sigma_v$ and study the effects of different σ_p, σ_v pairs. Fig. 17(b) shows the result of σ_p. We can find that when σ_p is set to values around $50 \sim 100$m, RTSS will reach the best performance. Moreover, we do the statistics of σ_{p_x} and σ_{p_y} of the candidates in the Xi'an dataset and get $\sigma_{p_x} = 47.6$m and $\sigma_{p_y} = 40.74$m. When we set σ_p to these values, the near-optimal performance inferred from Fig. 17(b) could be achieved.

For the parameter σ_v, from the result presented in Fig. 17(c) we can observe that $\sigma_v = 1$m/s is the optimal value and the larger the variance is, the worse the performance will be. This phenomenon shows that the model tends to assume the speed only varies within a very small range which is consistent with common knowledge since the speed is not likely to change a lot within a short duration.

We also want to mention that for both parameters, the performance w.r.t. these two parameters tends to be relatively stable, which is a useful property in applications since parameter tuning can be avoided. We claim this nice property is contributed by the incorporated road network. As constrained by the road network, candidates are forced to form a trajectory with many straight lines, which helps to eliminate a large portion of noises as well as to meet the assumption of linear Gaussian dynamical state space model.

5.3 Limitations

We are aware of the following limitations of our work. Firstly, we tested the approach on vehicle trajectories as well as on a simulated pedestrian trajectory. Though our results indicate that the approach works well for different movement speeds, it still needs to be investigated how well it generalizes to other modes of transportation. Second, we tested the approach with trajectories created in urban environments with an expected high cell tower

density. It still remains an open question how well the approach works for trajectories created in areas with a lower cell tower density.

6 CONCLUSIONS AND FUTURE WORK

In this paper, we propose a general system CLSTERS for reducing errors of trajectories generated by cellular-based positioning which has been adopted by HUAWEI. Our system requires *zero* hardware information of the trajectory which makes it general and ubiquitous. We theoretically study the along-road error which is the key issue faced by the existing map matching-based approaches. Aiming at this problem, we carefully design the solutions by proposing a candidate selection strategy and adopting the Bayesian smoother. We conduct the evaluation using three real world datasets generated by two different localization techniques in three different cities and the results show that CLSTERS can effectively reduce about 40% of original errors and hence largely outperforms the existing approaches. In the future work, we are going to study the noise relationship between several consecutive points and incorporate the driving behavior to further reduce the error.

A APPENDIX

A.1 Proof of Lemma 4.1

Proof. Notice, for simplicity, we assume the road segment, denoted by r, having the infinity length. We claim it is a reasonable assumption since the probability density of a Gaussian is nearly zero when we consider the part of the road which is far away from $\mathbf{q}$. According to Bayes' theorem,

$$\begin{aligned}\mathbb{E}_{\mathbf{q}\sim P(\mathbf{q}|\mathbf{p})}[\varepsilon^2] &= \int_{\mathbf{q}\in r} \|\mathbf{q}-\mathbf{p}\|_2^2 P(\mathbf{q}|\mathbf{p})\mathrm{d}\mathbf{q} = \int_{\mathbf{q}\in r} \|\mathbf{q}-\mathbf{p}\|^2 \frac{P(\mathbf{p}|\mathbf{q})P(\mathbf{q})}{P(\mathbf{p})}\mathrm{d}\mathbf{q} = \int_{\mathbf{q}\in r} \|\mathbf{q}-\mathbf{p}\|^2 \frac{P(\mathbf{p}|\mathbf{q})P(\mathbf{q})}{\int_{\mathbf{q}\in r} P(\mathbf{p}|\mathbf{q})P(\mathbf{q})\mathrm{d}\mathbf{q}}\mathrm{d}\mathbf{q} \\ &= \frac{\int_{\mathbf{q}\in r} \|\mathbf{q}-\mathbf{p}\|^2 P(\mathbf{p}|\mathbf{q})P(\mathbf{q})\mathrm{d}\mathbf{q}}{\int_{\mathbf{q}\in r} P(\mathbf{p}|\mathbf{q})P(\mathbf{q})\mathrm{d}\mathbf{q}} = \frac{\int_{\mathbf{q}\in r} \|\mathbf{q}-\mathbf{p}\|^2 P(\mathbf{p}|\mathbf{q})\frac{1}{len(r)}\mathrm{d}\mathbf{q}}{\int_{\mathbf{q}\in r} P(\mathbf{p}|\mathbf{q})\frac{1}{len(r)}\mathrm{d}\mathbf{q}} = \frac{\int_{\mathbf{q}\in r} \|\mathbf{q}-\mathbf{p}\|^2 P(\mathbf{p}|\mathbf{q})\mathrm{d}\mathbf{q}}{\int_{\mathbf{q}\in r} P(\mathbf{p}|\mathbf{q})\mathrm{d}\mathbf{q}} \end{aligned} \tag{3}$$

Let $C = \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} \exp\left(-\frac{p_y^2}{2\sigma_y^2}\right)$ and $t = \frac{1}{\sqrt{2(1-\rho^2)}}\left(\frac{p_x-q_x}{\sigma_x} - \frac{\rho p_y}{\sigma_y}\right)$, with the fact of the Gaussian integral, i.e., $\int_{-\infty}^{+\infty} \exp(-t^2)\mathrm{d}t = \sqrt{\pi}$ we have

$$\begin{aligned}\int_{\mathbf{q}\in r} P(\mathbf{p}|\mathbf{q})\mathrm{d}\mathbf{q} &= \int_{-\infty}^{+\infty} \frac{1}{2\pi\sigma_x\sigma_y\sqrt{1-\rho^2}} \exp\left[-\frac{1}{2(1-\rho^2)}\left(\frac{(p_x-q_x)^2}{\sigma_x^2} - \frac{2\rho(p_x-q_x)p_y}{\sigma_x\sigma_y} + \frac{p_y^2}{\sigma_y^2}\right)\right]\mathrm{d}q_x \\ &= -C\sigma_x\sqrt{2(1-\rho^2)}\int_{-\infty}^{+\infty} \exp\left[-\left(\frac{1}{\sqrt{2(1-\rho^2)}}\right)^2\left(\frac{p_x-q_x}{\sigma_x} - \frac{\rho p_y}{\sigma_y}\right)^2\right]\mathrm{d}\left[\frac{1}{\sqrt{2(1-\rho^2)}}\left(\frac{p_x-q_x}{\sigma_x} - \frac{\rho p_y}{\sigma_y}\right)\right] \\ &= -C\sigma_x\sqrt{2(1-\rho^2)}\int_{-\infty}^{+\infty} \exp(-t^2)\mathrm{d}t = -C\sigma_x\sqrt{2\pi(1-\rho^2)} \end{aligned} \tag{4}$$

Since $p_x - q_x = \sigma_x\sqrt{2(1-\rho^2)}\left(t + \frac{\rho p_y}{\sigma_y\sqrt{2(1-\rho^2)}}\right)$, we can get,

$$\int_{\mathbf{q}\in r}\|\mathbf{q}-\mathbf{p}\|_2^2 P(\mathbf{p}|\mathbf{q})\mathrm{d}\mathbf{q} = \int_{-\infty}^{+\infty}\frac{\left((p_x-q_x)^2+p_y^2\right)}{2\pi\sigma_x\sigma_y\sqrt{(1-\rho^2)}}\exp\left[-\frac{1}{2(1-\rho^2)}\left(\frac{(p_x-q_x)^2}{\sigma_x^2}-\frac{2\rho(p_x-q_x)p_y}{\sigma_x\sigma_y}+\frac{p_y^2}{\sigma_y^2}\right)\right]\mathrm{d}q_x$$

$$= -C\sigma_x^3(2(1-\rho^2))^{3/2}\int_{-\infty}^{+\infty}\left[t^2-\frac{\sqrt{2}\rho p_y}{\sqrt{1-\rho^2}\sigma_y}t+\frac{\rho^2p_y^2}{2(1-\rho^2)\sigma_y^2}\right]\exp\left(-t^2\right)\mathrm{d}t - p_y^2C\sigma_x\sqrt{2(1-\rho^2)}\int_{-\infty}^{+\infty}\exp(-t^2)\mathrm{d}t$$

$$= -C\sigma_x^3(2(1-\rho^2))^{3/2}\int_{-\infty}^{+\infty}t^2\exp\left(-t^2\right)\mathrm{d}t + 4C\sigma_x^3(1-\rho^2)\frac{2\rho p_y}{\sigma_y}\int_{-\infty}^{+\infty}t\exp\left(-t^2\right)\mathrm{d}t$$

$$- C\sigma_x\sqrt{2(1-\rho^2)}\left(\frac{\sigma_x^2}{\sigma_y^2}\rho^2+1\right)p_y^2\int_{-\infty}^{+\infty}\exp\left(-t^2\right)\mathrm{d}t$$

Note that $t\exp(-t^2)$ is an odd function thus $\int_{-\infty}^{+\infty}t\exp(-t^2)\mathrm{d}t = 0$ and for $\int_{-\infty}^{+\infty}t^2\exp(-t^2)\mathrm{d}t$, by substituting t^2 by u and with the help of Gamma function $\Gamma(\frac{1}{2}+n) = \frac{(2n)!}{4^n n!}\sqrt{\pi}$, we have

$$\int_{-\infty}^{+\infty}-t^2\exp(-t^2)\mathrm{d}t = 2\int_0^{+\infty}-t^2\exp(-t^2)\mathrm{d}t = 2\times\frac{1}{2}\int_0^{+\infty}u^{(\frac{1}{2}+1)-1}\exp(-u)du = \Gamma\left(\frac{1}{2}+1\right) = \frac{\sqrt{\pi}}{2}$$

Thus,

$$\int_{\mathbf{q}\in r}\|\mathbf{q}-\mathbf{p}\|_2^2 P(\mathbf{p}|\mathbf{q})\mathrm{d}\mathbf{q} = -C\sigma_x\sqrt{2\pi(1-\rho^2)}\left[\sigma_x^2(1-\rho^2)+\left(\frac{\sigma_x^2}{\sigma_y^2}\rho^2+1\right)p_y^2\right] \tag{5}$$

Finally, plugging Eq. (4) and Eq. (5) into Eq. (3), we can derive the final expectation of the squared error. Note that since we have already set the x-axis to the road segment, the vertical error ε_y is actually $|p_y|$. Finally we have,

$$\mathbb{E}_{\mathbf{q}\sim P(\mathbf{q}|\mathbf{p})}[\varepsilon^2] = \frac{\int_{\mathbf{q}\in r}\|\mathbf{q}-\mathbf{p}\|^2P(\mathbf{p}|\mathbf{q})\mathrm{d}\mathbf{q}}{\int_{\mathbf{q}\in r}P(\mathbf{p}|\mathbf{q})\mathrm{d}\mathbf{q}} = \sigma_x^2(1-\rho^2)+\left(\frac{\sigma_x^2}{\sigma_y^2}\rho^2+1\right)p_y^2 = \underbrace{\left(\left(\frac{\varepsilon_y^2}{\sigma_y^2}-1\right)\rho^2+1\right)\sigma_x^2}_{\varepsilon_x^2}+\varepsilon_y^2$$

□

ACKNOWLEDGMENTS

This research is supported in part by the cooperative research project of HUAWEI Wireless Network Research Department, National Natural Science Foundation of China (NSFC) and the National Research Foundation, Prime Ministers Office, Singapore under its International Research Centres in Singapore Funding Initiative.

REFERENCES

[1] Google maps for mobile. http://www.google.com/mobile/maps/.
[2] Mohamed Ali, John Krumm, Travis Rautman, and Ankur Teredesai. ACM SIGSPATIAL GIS cup 2012. In *Proceedings of the 2012 ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL'12)*. 597–600.
[3] David Bernstein and Alain Kornhauser. 1998. An introduction to map matching for personal navigation assistants. (1998).
[4] Jakub Borkowski and Jukka Lempiainen. Geometrical transformations as an efficient mean for reducing impact of multipath propagation on positioning accuracy. In *Proceedings of the 2004 IEE International Conference on 3G Mobile Communication Technologies (3G'04)*. 368–372.
[5] Jakub Borkowski, Jarno Niemelä, and Jukka Lempiäinen. Performance of cell ID+ RTT hybrid positioning method for UMTS radio networks. In *Proceedings of the 2004 European Wireless Conference (EW'04)*. 487–492.

[6] Sotiris Brakatsoulas, Dieter Pfoser, Randall Salas, and Carola Wenk. On map-matching vehicle tracking data. In *Proceedings of the 2005 International Conference on Very Large Data Bases (VLDB'05)*. 853–864.
[7] James J. Caffery and Gordon L. Stuber. 1998. Overview of radiolocation in CDMA cellular systems. *IEEE Communications Magazine* 36, 4 (1998), 38–45.
[8] Chao Chen, Daqing Zhang, Pablo Samuel Castro, Nan Li, Lin Sun, Shijian Li, and Zonghui Wang. 2013. iBOAT: isolation-based online anomalous trajectory detection. *IEEE Trans. Intelligent Transportation Systems* 14, 2 (2013), 806–818.
[9] Daniel Chen, Anne Driemel, Leonidas J. Guibas, Andy Nguyen, and Carola Wenk. Approximate map matching with respect to the Fréchet distance. In *Proceedings of the 2011 Meeting on Algorithm Engineering & Expermiments (ALENEX '11)*. 75–83.
[10] Mike Y. Chen, Timothy Sohn, Dmitri Chmelev, Dirk Haehnel, Jeffrey Hightower, Jeff Hughes, Anthony LaMarca, Fred Potter, Ian Smith, and Alex Varshavsky. Practical metropolitan-scale positioning for GSM phones. In *Proceedings of the 2006 International Conference on Ubiquitous Computing (UbiComp'06)*. 225–242.
[11] Dingxiong Deng, Cyrus Shahabi, Ugur Demiryurek, Linhong Zhu, Rose Yu, and Yan Liu. Latent space model for road networks to predict time-varying traffic. In *Proceedings of the 2016 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD'14)*. 1525–1534.
[12] Nilesh Dubey, Vandana Dubey, and Shivangi Bande. 2011. Cell-ID based vehicle locator and real-time deactivator using GSM network. In *Information Technology and Mobile Communication*. Springer, 82–86.
[13] Stefan Funke and Sabine Storandt. Path shapes: an alternative method for map matching and fully autonomous self-localization. In *Proceedings of the 2011 ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL'11)*. 319–328.
[14] Mohamed Ibrahim and Moustafa Youssef. CellSense: a probabilistic RSSI-based GSM positioning system. In *Proceedings of the 2010 IEEE Global Telecommunications Conference (GLOBECOM'10)*. 1–5.
[15] Mohamed Ibrahim and Moustafa Youssef. A hidden Markov model for localization using low-end GSM cell phones. In *Proceedings of the 2011 IEEE International Conference on Communications (ICC'11)*. 1–5.
[16] Mohamed Ibrahim and Moustafa Youssef. 2012. CellSense: an accurate energy-efficient GSM positioning system. *IEEE Transactions on Vehicular Technology* 61, 1 (2012), 286–296.
[17] Rudolph Emil Kalman and others. 1960. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering* 82, 1 (1960), 35–45.
[18] Regina Kaune. Accuracy studies for TDOA and TOA localization. In *Proceedings of the 2012 IEEE International Conference on Information Fusion (FUSION'12)*. 408–415.
[19] JS Kim. 1996. Node based map matching algorithm for car navigation system. In *Proceedings of the 1996 International Symposium on Automotive Technology & Automation. Global Deployment of Advanced Transportation Telematics/ITS*.
[20] Peter Lamb and Sylvie Thiébaux. Avoiding explicit map-matching in vehicle location. In *Proceedings of the 1999 World Conference on Intelligent Transportation Systems (ITS'99)*. 1–9.
[21] Minghui Li and Yilong Lu. 2008. Angle-of-arrival estimation for localization and communication in wireless networks. In *Proceedings of the 2008 IEEE European Signal Processing Conference (EUSIPCO'08)*. IEEE, 1–5.
[22] Yin Lou, Chengyang Zhang, Yu Zheng, Xing Xie, Wei Wang, and Yan Huang. Map-matching for low-sampling-rate GPS trajectories. In *Proceedings of the 2009 ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL'09)*. 352–361.
[23] Yisheng Lv, Yanjie Duan, Wenwen Kang, Zhengxi Li, and Fei-Yue Wang. 2015. Traffic flow prediction with big data: a deep learning approach. *IEEE Transactions on Intelligent Transportation Systems* 16, 2 (2015), 865–873.
[24] Shuo Ma, Yu Zheng, and Ouri Wolfson. T-share: A large-scale dynamic taxi ridesharing service. In *Proceedings of the 2013 IEEE 29th International Conference on Data Engineering (ICDE'13)*. 410–421.
[25] Reham Mohamed, Heba Aly, and Moustafa Youssef. 2016. Accurate real-time map matching for challenging environments. *IEEE Transactions on Intelligent Transportation Systems* (2016).
[26] Paul Newson and John Krumm. Hidden Markov map matching through noise and sparseness. In *Proceedings of the 2009 ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL'09)*. 336–343.
[27] Takayuki Osogami and Rudy Raymond. Map matching with inverse reinforcement learning.. In *Proceedings of the 2013 International Joint Conference on Artificial Intelligence (IJCAI'13)*. 2547–2553.
[28] Meng Qu, Hengshu Zhu, Junming Liu, Guannan Liu, and Hui Xiong. A cost-effective recommender system for taxi drivers. In *Proceedings of the 2014 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD'14)*. 45–54.
[29] Herbert E. Rauch, F. Tung, and C. T. Striebel. 1965. Maximum likelihood estimates of linear dynamic systems. *AIAA Journal* 3, 8 (1965), 1445–1450.
[30] Xin-Long Luo Rong-Zheng Li and Jia-Ru Lin. 2011. Weighted centroid correction localization in cellular systems. *American Journal of Engineering and Applied Sciences* 4, 1 (2011), 37–41.
[31] Simo Särkkä. 2013. *Bayesian filtering and smoothing*. Vol. 3. Cambridge University Press.

[32] Renchu Song, Wei Lu, Weiwei Sun, Yan Huang, and Chunan Chen. Quick map matching using multi-core CPUs. In *Proceedings of the 2012 ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL'12)*. 605–608.
[33] Emiliano Trevisani and Andrea Vitaletti. Cell-ID location technique, limits and benefits: an experimental study. In *the 2004 IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'04)*. 51–60.
[34] Reza Monir Vaghefi and R. Michael Buehrer. 2014. Cooperative RF pattern matching positioning for LTE cellular systems. (2014), 264–269.
[35] Alex Varshavsky, Mike Y. Chen, Eyal de Lara, Jon Froehlich, Dirk Haehnel, Jeffrey Hightower, Anthony LaMarca, Fred Potter, Timothy Sohn, Karen Tang, and Ian Smith. Are GSM phones THE solution for localization?. In *Proceedings of the 2006 IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'06)*. 20–28.
[36] Yilun Wang, Yu Zheng, and Yexiang Xue. Travel time estimation of a path using sparse trajectories. In *Proceedings of the 2014 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD'14)*. 25–34.
[37] Hong Wei, Yin Wang, George Forman, Yanmin Zhu, and Haibing Guan. Fast Viterbi map matching with tunable weight functions. In *Proceedings of the 2012 ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (SIGSPATIAL'12)*. 613–616.
[38] Hao Wu, Jiangyun Mao, Weiwei Sun, Baihua Zheng, Hanyuan Zhang, Ziyang Chen, and Wei Wang. Probabilistic robust route recovery with spatio-temporal dynamics. In *Proceedings of the 2016 ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD'16)*.
[39] Hao Wu, Weiwei Sun, and Baihua Zheng. Is only one GPS position sufficient to locate you to the road network accurately?. In *Proceedings of the 2016 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp'16)*. 740–751.
[40] Hao Wu, Chuanchuan Tu, Weiwei Sun, Baihua Zheng, Hao Su, and Wei Wang. GLUE: a parameter-tuning-free map updating system. In *Proceedings of the 2015 ACM International on Conference on Information and Knowledge Management (CIKM'15)*. 683–692.
[41] Jing Yuan, Yu Zheng, Chengyang Zhang, Xing Xie, and Guang-Zhong Sun. An interactive-voting based map matching algorithm. In *Proceedings of the 2010 International Conference on Mobile Data Management (MDM'10)*. 43–52.
[42] Kai Zheng, Yu Zheng, Xing Xie, and Xiaofang Zhou. Reducing uncertainty of low-sampling-rate trajectories. In *Proceedings of the 2012 IEEE International Conference on Data Engineering (ICDE'12)*. 1144–1155.
[43] Yu Zheng and Xiaofang Zhou. 2011. *Computing with spatial trajectories*. Springer.

Received May 2017; accepted July 2017