Accurate Real-time Map Matching for Challenging Environments

Reham Mohamed, Heba Aly, *Student Member, IEEE*, and Moustafa Youssef, *Senior Member, IEEE*

***Abstract*—We present the *SnapNet* system, which provides accurate real-time map matching for cellular-based trajectory traces. Such traces are characterized by input locations that are far from the actual road segment, errors on the order of kilometers, back-and-forth transitions, and highly sparse input data. *SnapNet* applies a series of filters to handle the noisy locations and an interpolation stage to address the data sparseness. At the core of *SnapNet* is a novel incremental HMM algorithm that combines digital map hints in the estimation process and a number of heuristics to reduce the noise and provide real-time estimations. Evaluation of *SnapNet* using actual traces from different cities covering more than 400 km shows that it can achieve a precision and recall of more than 90% under noisy coarse-grained input location estimates. This maps to over 97% and 34% enhancement in precision and recall, respectively, when compared to the traditional HMM map-matching algorithms. Moreover, *SnapNet* has a latency of 0.58 ms per location estimate.**

***Index Terms*—Map matching, Hidden Markov Model, cellular-based trajectories , crowdsourcing.**

I. Introduction

MAP matching, the problem of mapping a set of coordinates with errors to the corresponding points on the road network, has been used widely in location-based applications including car navigation, directions finding, car heading estimation, automatic scheduling of the public transportation systems, traffic analysis, among others. A large number of map matching techniques [2], [14], [20], [23], [27], [29], [34], [42] have been proposed in literature, all based on GPS as the ubiquitous localization technology for outdoors. However, since GPS is an energy hungry device and may not be available everywhere; e.g. due to tall buildings surrounding the road, inside tunnels, and/or in bad weather conditions [15]; a number of map matching techniques have been proposed that can handle a lower sampling rate and lower localization accuracy. These techniques usually depend on using other phone sensors such as WiFi [38] or inertial sensors [6], [21], [39], [40]; or do extensive war driving effort assuming the availability of neighbouring cell tower information [39][1].

In this paper, we argue that there is a growing number of map matching applications that cannot assume the existence of GPS or other phone sensors. In particular, traffic estimation from the cellular provider-side (where only the associated cell tower and signal strength information is available for each cell phone), low-energy GPS-less localization [7], [8], [31], applications that involve low-end phones which do not have GPS or other sensors (e.g. in developing countries), and crowd-sensing applications in which users are not actively involved [17], [18] are examples of such applications. In such scenarios, the typical assumptions of traditional map matching algorithms do not hold, where previous map matching systems assume the presence of inertial sensors, low-sampled GPS or WiFi which are not available in our case. Specifically, as shown in Fig. 1, the phone location is no longer near the actual road segment, the location error is in the order of kilometers, there are back-and-forth transitions (where the input location jumps around due to changing the cell tower association), as well as the input location samples are highly sparse. These strict/harsh constraints lead to a much harder map matching problem in terms of quality of input points and number of candidate road segments, affecting both the accuracy and computational complexity.

We therefore present *SnapNet*, a system for accurate and efficient map matching for challenging environments. *SnapNet* is based on a number of heuristics rooted on a set of observations from real-life usage scenarios. Specifically, we note that the traffic distribution on different roads is non-uniform and that users are more likely to take major roads than non-major ones. In addition, users prefer to take the shortest path between any two points and they would not make several turns in a short time. Finally, the majority of users always follow the traffic rules; therefore their paths follow the topological relations of the road network. Based on this, we present a *novel* incremental Hidden Markov Model (HMM)-based map matching algorithm that takes the noise of the input data into account as well as digital map hints to enhance the accuracy of the estimated road segments and efficiently handles the increased number

Manuscript received November 28, 2015; revised April 11, 2016 and June 30, 2016; accepted July 9, 2016. Date of publication August 10, 2016; date of current version March 27, 2017. A poster has been presented about this work at the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL 2014) [30]. The Associate Editor for this paper was J. E. Naranjo.

R. Mohamed is with the Wireless Research Center, Egypt-Japan University for Science and Technology, Alexandria 21934, Egypt (e-mail: reham.mohamed@ejust.edu.eg).

H. Aly was with the Wireless Research Center, Egypt-Japan University for Science and Technology, Alexandria 21934, Egypt. She is now with the University of Maryland, College Park, MD 20742 USA (e-mail: heba@cs.umd.edu).

M. Youssef is with Egypt-Japan University of Science and Technology, Alexandria 21934, Egypt. He is also on sabbatical from Alexandria University, Alexandria 21526, Egypt (e-mail: moustafa.youssef@ejust.edu.eg).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TITS.2016.2591958

[1]The majority of cell phones in the market only give access to the associated cell tower information, with no access to the neighbouring cell tower information. This reduces the accuracy of the cellular-based localization techniques significantly [24].

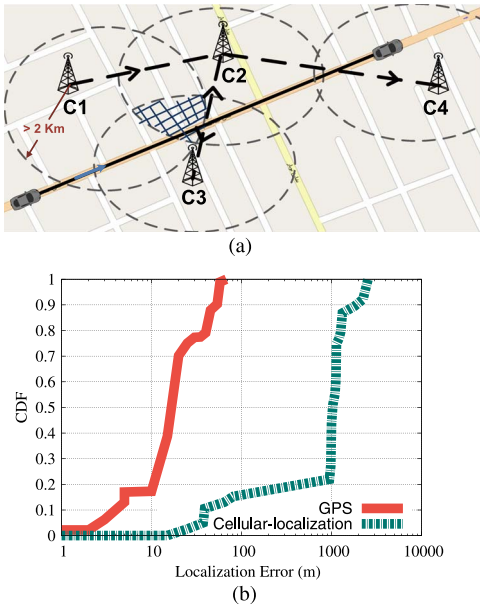


Fig. 1. Challenges of using cellular-based locations in map matching. (a) Cellular-based traces. Reported location is far from the correct road, location update rate is only at handover, and there are back-and-forth jumps (The user is moving straight from left to right on the orange road but her location estimates sequence is C1–C2–C3–C2–C4 due to the cell towers' overlapping coverage in Area A). (b) Localization error for the different localization techniques (**log scale for the x-axis**).

of candidate road segments. This HMM is combined with a number of preprocessing modules that reduce the noise and sparseness in the input data.

We have implemented *SnapNet* and evaluated it on driving traces collected in multiple cities. Our results show that *SnapNet* can achieve 90% accuracy in identifying the correct road segment. This comes with a low latency in computation of 0.58 ms per location estimate.

In summary, our contributions are:

- We present the architecture of *SnapNet*: a real-time map-matcher for challenging environments.
- We present effective preprocessing modules for challenging cellular-based trajectories. The modules handles high noise, back-and-forth transition and sparseness in the input locations.
- We present the details of our novel incremental HMM algorithm that combines digital map hints and a number of heuristics to provide accurate map-matching estimates in real-time.
- We implement the *SnapNet* system and evaluate its performance using real traces with different accuracies as compared to the defacto standard HMM-based map-matching algorithms at different environments.

The rest of the paper is organized as follows: we discuss related work in Section II. We then present the *SnapNet* system architecture and the different pre-processing modules in Section III. Section IV gives the details of the novel map matching algorithm and how it handles the noise of the input data while maintaining the efficiency of computation. In Section V, we present our experimental evaluation. Finally, Section VI concludes the paper.

II. Related Work

The map matching problem has attracted lots of attention from researchers due to its importance for several location-based services, e.g. [1], [3]–[5], [26]. A large number of techniques have been proposed over the years for map matching GPS location data [2], [14], [20], [23], [27], [29], [34], [42]. These techniques can be classified as geometric or topological techniques.

Geometric techniques utilize the geometry of the input trace and the road network. They consider only the shape of the links regardless of their connectivity. Examples of geometric techniques include point-to-point matching [11], point-to-curve matching [11], [41] and curve-to-curve matching [11], [33], [41]. While these techniques could provide good map-matching accuracy for GPS trajectories, they are unsuitable for cellular-based trajectories. Since cellular-based trajectories are a series of noisy cell-tower locations, their shape is unsuitable for matching using geometric techniques.

On the other hand, topological techniques, e.g. [12], [20], [23], [27], [34], leverage the relation, e.g. connectivity, between map elements for map matching. Among those, HMMs provide a probabilistic framework to address the noise in the input data with remarkable accuracy when the sampling rate is high. A subset of these techniques, e.g. [23], [27], investigated the use of HMM for map matching with a low density of GPS samples. However, even with a low-density GPS samples, the accuracy of the reported location is high (input location is close to the correct road segment) and there are no back-and-forth transitions, leading to a much easier problem compared to using the coarse-grained cellular-based location. Hence, their accuracy degrades significantly when map-matching cellular-trajectories (see Section V).

Recently, a few systems were proposed to solve the map matching problem for network-based location data [6], [21], [38]–[40]. VTrack [38] builds an HMM-based map matching scheme leveraging WiFi information to handle the inaccuracy of cellular location information. CTrack [39] alternatively uses *war-driving training* data in addition to inertial sensors to reduce the inaccuracy of cellular-based locations. AutoWitness [21] and WheelLoc [40] leverage inertial sensors and dead-reckoning to reduce the inaccuracy of cellular-based localization. SemMatch [6] leverages inertial sensors to identify road semantics and use them as hints to improve the map-matching accuracy.

Unlike these systems, ***SnapNet is unique*** in map-matching the more challenging raw coarse-grained cellular-based locations without using any training and/or additional sensors. Hence, it is more ubiquitous in terms of covered cellphones and applications domains. Table I highlights the main differences between *SnapNet* and the closest map matching techniques. A poster has been presented about this work [30] covering the basic idea. This paper presents the full system design, architecture, details of the different components, as well as extensive evaluation of the proposed system.

TABLE I
COMPARISON BETWEEN *SnapNet* AND RELATED SYSTEMS

	Req. WiFi	Uses cell. loc.	Req. war-drv.	Req. iner. sen.	Uses 1 cell tow.
VTrack [38]	Yes	No	Yes	No	N/A
CTrack [39]	No	Yes	Yes	Yes	No
AutoWitness [21]	No	Yes	No	Yes	Yes
WheelLoc [40]	No	Yes	No	Yes	Yes
SemMatch [6]	No	Yes	No	Yes	Yes
SnapNet	**No**	**Yes**	**No**	**No**	**Yes**

III. THE *SnapNet* SYSTEM

In this section, we provide an overview of the *SnapNet* system. We start by the system assumptions followed by the details of the different pre-processing modules for handling the noise and sparsity of the input data.

A. System Assumptions

SnapNet is based on the following assumptions that allow it to achieve its goals: First, we assume that users are more probably to take major roads rather than secondary roads. This is confirmed by transportation statistics collected in Britain, where traffic statistics covering 317.1 billion miles, show that 65.7% of the vehicle traffic is on major roads, while 34.3% is on non-major roads.[2] This number is consistent in different countries, such as Scotland.[3] This property is also used by route planning techniques [36] and navigation systems, which exploit the hierarchical structure of the road network and classify roads according to their importance/weight. Second, we further assume that users are more likely to stay on the same road than make frequent turns/road changes. Third, we assume that the distance between the observed point and the candidate roads follows a Gaussian distribution and that the difference in distances between the observed points and their projections on the road segments follows an exponential distribution. These assumptions have been validated in literature [32]. Finally, we assume that the users are more likely to take the shortest path between any two points.

B. Overview

Fig. 2 shows an overview of the *SnapNet* architecture. The input to *SnapNet* is time-stamped location data. Each location is represented as a (latitude, longitude, error) triplet that captures the user estimated location as well as an estimate of the localization error. *SnapNet* targets scenarios with challenging localization environments, as shown in Fig. 1. For example, in a cell-ID based localization system, the location of the device is often estimated as the location of the associated cell tower. This leads to a localization error in the order of kilometers [see Fig. 1(b)] and locations that are far from the actual road segment the user is on. In addition, since the user location will only change when the associated cell tower changes, this leads to a highly sparse location information. Moreover, cellular networks suffer from a phenomenon called the "Ping-Pong" effect [16], [28], where a user within the coverage area of two or more cell towers continuously changes her association due to the dynamic load on the cell towers, leading to fluctuation in the reported user location even with no device movement.

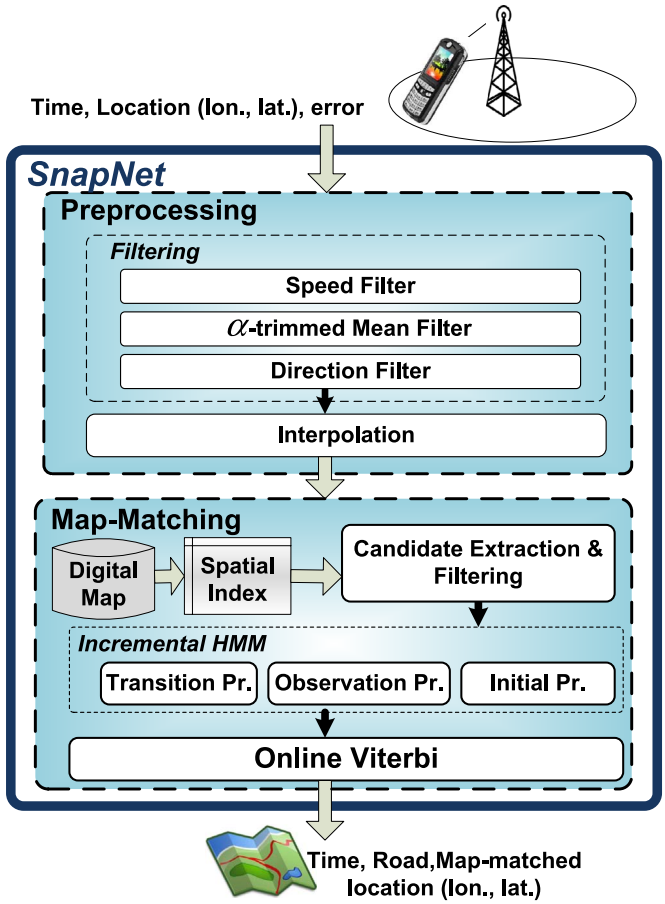


Fig. 2. *SnapNet* system architecture.

SnapNet map-matches the user trajectory incrementally in real-time as the user sends her cellular-based location with its error estimate point by point. It starts by preprocessing these location points in two steps: filtering and interpolation. A series of consecutive filters are applied to the raw data to eliminate the noise. However, this makes the traces much sparser. Therefore, we apply interpolation on the filtered data to reduce this sparseness.

The filtered and interpolated points are then passed to the map matching module that employs a novel HMM algorithm to handle the noisy location data in an efficient manner. Specifically, our map matching algorithm contains three sub-modules: Candidate Extraction and Filtering, Incremental Map Matching, and an Online Viterbi Algorithm. The *Candidate Extraction and Filtering* module determines the candidate road segments from the OpenStreetMaps[4] digital maps taking into account the error in the input location and the previous estimated user location. The *Incremental Map Matching* algorithm integrates

[2]https://www.gov.uk/government/uploads/system/uploads/attachment_data/file/499046/prov-road-traffic-estimates-jan-to-dec-2015.pdf

[3]http://www.transport.gov.scot/statistics/j285663-08.htm

[4]http://www.openstreetmap.org/

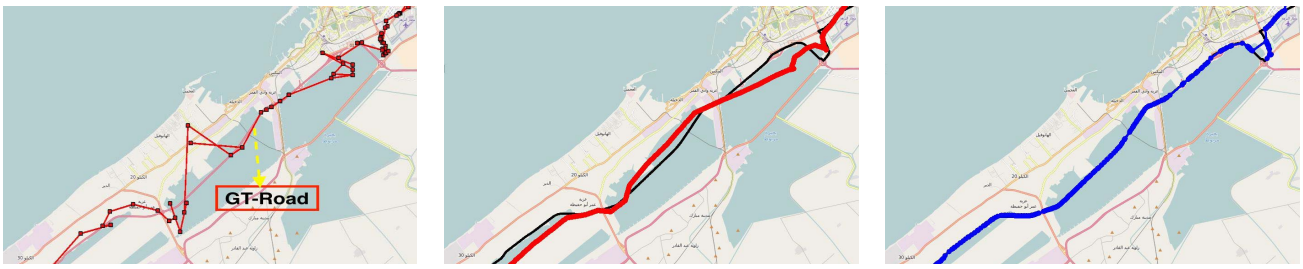


Fig. 3. Operation stages of *SnapNet*. (Black trace) Ground truth. (Red with squares) Raw input trace. The label GT-road shows the actual ground truth road crossing the lake.(a) Raw trace. (b) *SnapNet* preprocessed trace. (c) *SnapNet* map-matched trace.

a number of modifications to the standard HMM algorithm to take the quality of the input data into account as well as digital map hints to enhance the accuracy of the estimated road segments. Finally, the *Online Viterbi Algorithm* efficiently calculates the probabilities of the different road segments and determines the most probable road segment. The Map Matcher outputs the matched road segment along with the estimated location on it. Fig. 3 shows an example of *SnapNet* algorithm stages in action.

In the balance of this section, we provide the details of the different pre-processing modules. We leave the details of the Map Matching modules to the next section.

C. Filtering Modules

Through a series of consecutive filters we aim to detect transitions that are unlikely to be on the actual path and filter them out. In particular, we apply three filters in succession: the *Speed filter*, the α-*trimmed filter*, and the *Direction filter*. We give the details of the different filters in the following subsections.

1) Speed Filter: The user's speed is limited by many factors including the vehicle maximum speed and the road speed limit. Hence, we assume that the user does not exceed a certain speed threshold $(\nu_{\max})$ obtained from the digital map and physical speed limits. If the user's estimated current speed exceeds this threshold, her current location estimate is detected as an outlier.

The speed between any two points can be calculated by dividing the geodesic distance between the two points by the difference in their time-stamps. However, due to the high error in the input locations, this speed is noisy to use (see Fig. 4). Instead, we estimate the user's current speed (ν_p) by averaging the speed between this location and a window of the preceding unfiltered locations as in Equation (1).

$$\nu_p = \frac{1}{w_s} \sum_{i=p-w_s}^{p-1} \frac{d_{i,p}}{(t_i - t_p)} \tag{1}$$

where p is the index of the current input location, w_s is the window size, $d_{i,p}$ is the geodesic distance between locations i and p, and t_i, t_p are the time-stamps of locations i and p respectively.

If ν_p exceeds the speed limit $(\nu_{\max})$, we filter out its corresponding location point.

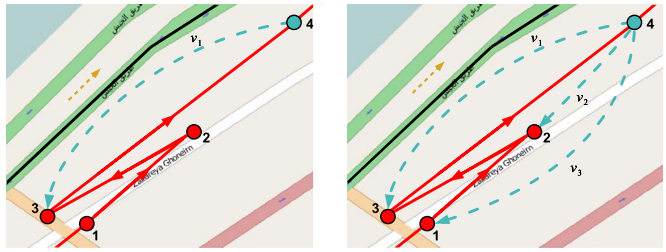


Fig. 4. Advantage of using a window for the speed filter. Estimating speed based on the previous point only will lead to filtering Point 4 due to the noisy Point 3. Using the average speed between Point 4 and a window of points, this effect is reduced. (a) $v = v_1 > v_{\max}$. (b) $v = ((v_1 + v_2 + v_3)/3) < v_{\max}$.

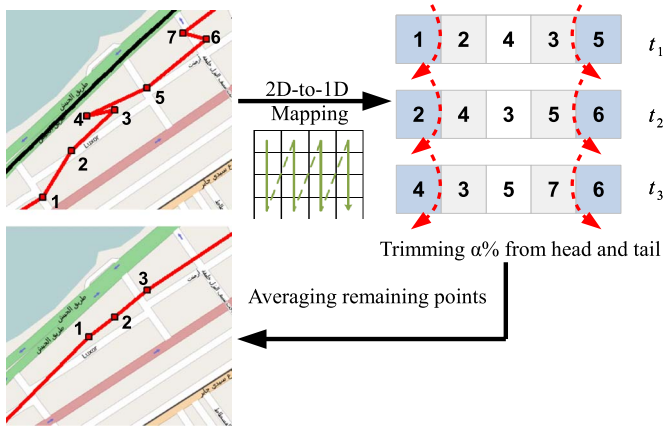
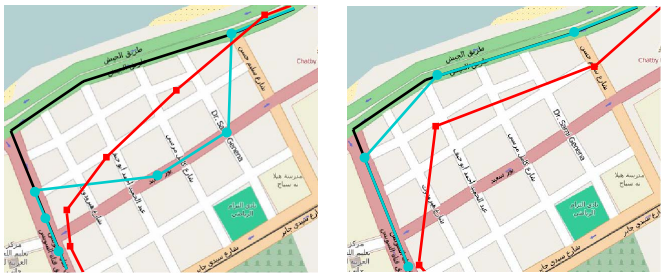


Fig. 5. Operation of the α-trimmed filter $w_a = 5$. Point numbers represent the time index. (a) Two-dimensional points are reordered and mapped into a 1-D vector using the linear space-filling curve. (b) $\alpha\%$ of the points are removed from the head and tail of the vector, and the mean of the remaining points is calculated.

2) α-Trimmed Mean Filter: To reduce the back-and-forth transitions, we apply an α-trimmed mean filter [10] on the location points. An α-trimmed filter has the advantage of handling both impulse and Gaussian noise, as compared to mean and median filters that can handle only one of them. In addition, it is simple to implement.

The basic idea (see Fig. 5) is to look at the neighbors of each point, remove 2α of the extreme neighbors, i.e. outliers, then replace the point by calculating the mean of the unfiltered neighbors. Therefore, at $\alpha = 0$, the filter works as a standard average filter while at $\alpha = 0.5$, the filter works as a median filter.



(a) (b)

Fig. 6. Effect of the α-trimmed filter for the different values of α. (Black trace) Ground truth. (Red with squares) Preprocessed trace. (Blue with circles) Final map-matched trace. (a) $\alpha = 0$ (Average filter). (b) $\alpha = 0.2$.

In particular, we apply the α-trimmed filter on the input locations with a sliding window of size w_a: First we order the location points such that $\text{loc}_1 \leq \text{loc}_2 \leq \cdots \leq \text{loc}_{w_a}$. Then we remove α of the points from the head and tail. Finally, we return the mean of the remaining points. The output of the filter will be:

$$S(\alpha, w_a) = \frac{1}{w_a - 2\lceil \alpha w_a \rceil} \sum_{i=\lceil \alpha w_a \rceil + 1}^{w_a - \lceil \alpha w_a \rceil} \text{loc}_i \qquad (2)$$

where $\lceil . \rceil$ denotes the greatest integer part and $0 < \alpha < 0.5$.

Note that to apply the filter, we need to sort the locations (loc_i). We experimented with different space-filling curves and found that applying the linear space filling curve [9] provides good accuracy while maintaining low computational time. This is intuitive as most of the time the user moves in straight lines.

Fig. 6 shows an example of applying the α-trimmed filter with $\alpha = 0.2$ to a trace with a noisy part. The figure shows that the filtered trace can capture the actual shape of the trajectory better than a standard average filter.

3) Direction Filter: To further reduce the back-and-forth transitions, we apply the direction filter. Note that back-and-forth transitions lead to a change in the user direction, i.e., as if the user made a u-turn. Therefore, the direction filter ensures that the change in the user's direction is only allowed when we are sure that it is originating from an actual change in direction, not due to noise in the location data. To do that, if a location point indicates a direction change, we cache the point temporarily till we get a new location point. If the new location confirms the direction change, we add the two points to the pre-processed trace. Otherwise, we drop the cached point. We found that using only one new location to confirm the direction change was enough as most of the noisy points were filtered in the preceding filtering stages.

D. Interpolation

After applying the filters on the raw input location points, the sparseness of the data points increases. To overcome this, we apply linear interpolation [25] on the unfiltered location points when needed. Specifically, for each unfiltered point (p), we calculate the distance between it and its preceding point $(p-1)$, if the distance between them is above a threshold, we add linearly interpolated points between them at equally-spaced distance as in Equation (3).

$$\text{loc}_{\text{inter}_i} = \text{loc}_{p-1} + \frac{\text{step} \times i \times (\text{loc}_{p-1} - \text{loc}_p)}{d_{p-1,p}} \qquad (3)$$

where $\text{loc}_{\text{inter}_i}$ is the ith interpolated point, $d_{p-1,p}$ is the geodesic distance between locations $p-1$ and p, and *step* is the equal distance between the interpolated points (50 m in our experiments).

Taking the interpolated points on the straight line between the two cellular locations achieves accurate map-matching of low-sampled coarse-grained locations without performing expensive shortest path computations.

IV. The *SnapNet* Map Matcher

The input to the *Map Matcher* module is the filtered and interpolated location data as well as the digital map. We model the map matching problem as a Hidden-Markov Model (HMM) [35]. Our extended incremental HMM can effectively fuse the noisy input location data and the provided road network constraints in a sound way to provide accurate map matching. The *Map Matcher* module has three sub-modules: the *Candidate Extraction and Filtering* module, the *Incremental Map Matching* module, and an *Online Viterbi Algorithm*. This section provides the details of each of these sub modules and the extensions we propose to the standard HMM algorithm to provide accurate and real-time map matching.

A. Candidate Extraction and Filtering Module

For each input location $(\text{lat}_t, \text{lon}_t, \text{err}_t)$ at time t to the *Map Matcher* module, we extract all candidate road segments that intersect with the circle centered at $(\text{lat}_t, \text{lon}_t)$ with radius err_t using the digital map. We use err_t as, most probably, the actual location will lie inside the error circle. To speed up the candidate extraction, we build an R-tree spatial index [22] on all possible road segments in the road network.

We further filter these candidate road segments by removing any segment that is not connected to at least one of the candidate road segments from the previous estimation step. The unfiltered road segments represent the candidates for the current observation. An observation point with an empty candidates set is considered as an outlier.

B. Incremental Map Matching

SnapNet employs an extended HMM to address the noisy input data. In particular, we provide (a) dynamic parameter estimation based on the noise in the input data, (b) a novel road weighting technique to leverage hints from the road network to enhance accuracy, and (c) apply a heuristic for road transitions to handle the false transitions.

Traditional HMM-based map matching approaches use a window of input locations to estimate the corresponding map matched locations. This technique leads to a good accuracy, however, it increases the latency of the estimated location as the system has to wait for a full window of samples before it

can produce the output. This is even worse with the coarse-grained and sparse cellular location information as the number of candidate road segments is high, leading to more calculations per iteration. To address this latency issue, *SnapNet* uses an incremental map matching approach, where a sliding window is used for estimation: at each time instant, a new location sample is added and the oldest sample is removed from the window.

A modified version of the online Viterbi algorithm [13], [37] is then applied to compute the maximum likelihood sequence (MLS) of hidden states for the current window using dynamic programming by extending the current solution with the estimated HMM parameters to get the best path.

The incremental HMM algorithm works as follows: When a new location enters the system, the HMM parameters are calculated for this point and a new row is added to the online Viterbi matrix while removing the oldest row (sliding window). The MLS estimation is applied on this matrix to get the best road segment for the newly added point. The system is updated with the new segment in real time. The Viterbi matrix starts empty and increases incrementally when a new point is added until it reaches a maximum size (e.g. 100 rows). After that it works using a sliding window approach with the new point replacing the oldest point. Note that unlike [13], [37], our algorithm does not wait for coalescence points to produce the output. This guarantees that the output of any new input point is always in real time, without any latency except for computing the HMM parameters of this point. However, the disadvantage of this scheme is that the maximum likelihood sequence is computed for each point separately regardless of the output sequence of the previous points. This might result in apparently broken topological rules [e.g., see Fig. 8(a)]. The transition weights, which prefer staying on the same road, reduces this effect as discussed in Section IV-B3.

We start this section by providing the mathematical model and notations followed by the details of how *SnapNet* estimates the different model parameters.

1) Mathematical Model and Notations: The input to the *Map Matching* module is a set of T cellular locations $Z = (z_1, \ldots, z_T)$, where each $z_t, 1 < t < T$ is a triplet in the form (latitude, longitude, error) representing the location information for the t^{th} location sample. Let $S_t = \{s_{1,t}, s_{2,t}, \ldots, s_{N_t,t}\}$ be the set of possible states, i.e. road segments, at time $1 < t < T$ obtained by the *Candidate Extraction and Filtering* module, where $N_t = |S_t|$.

For each road segment i, The *Map Matching* module uses two probability distributions:

1) The state transition probability distribution between road segments i and j, $A = \{a_{ij}\}$, where $a_{ij} = P[s_{j,t}|s_{i,t-1}]$
2) The observation probability distribution in state i, $B = P[z_t|s_{i,t}]$, i.e. the probability of observing a certain input location given the user is actually on road segment i.

In addition, the module calculates the initial state distribution $\pi = \{\pi_i\}$, where $\pi_i = P[s_{i,1}]$.

Therefore, the problem becomes, given a sequence of location observations $Z = (z_1, \ldots, z_T)$, we want to find the most probable sequence of road segments (states) $Q = (q_1, \ldots, q_T)$, where each $q_t \in S_t, 1 < t < T$.

Fig. 7. Using weighted distance based on the road type. Cross is the initial input location. Circles are the apparent locations for each road after applying the road weights.

In the next subsections, we show how the HMM distributions are calculated in *SnapNet* and how the optimal state sequence is obtained.

2) Observation Probability: The observation probability $p(z_t|s_{i,t})$ is the probability that state (i.e. road segment) $s_{i,t}$ emits the observation (i.e. input location) z_t. To estimate the observation probability, we take it as a function of the distance between the observed location and the projected location on the corresponding road segment. The intuition is that the closer the observed location to the road segment, it is more probable that this road is the user's actual segment. It has been shown that this distance can be modeled accurately using a Gaussian distribution for GPS trajectories [19], [32]. We extend the Gaussian model in literature to fit our problem. Specifically, the observation probability in *SnapNet* is modeled as:

$$p(z_t|s_{i,t}) = \frac{1}{\sqrt{2\pi}\sigma_t} e^{-0.5\left(\frac{\text{dist}(z_t, s_{i,t})}{\sigma_t}\right)^2} \tag{4}$$

where $\text{dist}(z_t, s_{i,t})$ is the geodesic distance between z_t and $s_{i,t}$, and σ_t of the standard deviation of a Gaussian random variable that corresponds to the error in the sensor measurements. *SnapNet* has **three novel contributions** to this observation probability model compared to literature: **First**, different from literature that use a fixed value for σ_t, and since the error in the input location in our case varies with time, e.g. based on the range of the cell-tower the user is associated with, we set the value of sigma dynamically based on the input location error. Specifically, we estimate the value of σ_t as the average location error over a time window. We found that the accuracy of the system is not sensitive for window sizes larger than 10, so we use it in our evaluation.

Second, to further handle the "ping-pong" effect and errors in the digital map, we also consider the relation between the direction of the candidate road segment and the direction of the input trajectory in calculating the observation probability: roads in the opposite direction of the input trajectory are assigned a small probability.

Third, due to the large error in the input locations (order of kilometers), even with these extensions, the model gave poor map matching accuracy (see Section V). To further enhance accuracy, we exploit the hints from the digital map. The idea is that major roads are more likely to be traversed than side roads. Therefore, we leverage this observation to give different weights to different roads based on their type. Fig. 7 shows

the basic idea: Instead of using the real distance between the observation and the road, we multiply this distance by a weight < 1 that is proportional to the road type. This weighted distance is then fed into the Gaussian distribution to estimate the observation probability. Note that more important roads are given smaller weights, reflecting smaller distances, which in turn implies favoring them. The road importance can be obtained from the digital map; for example, OpenStreetMaps have eight pre-defined road types. Alternatively, the number of lanes per road can be used to estimate the road importance.

To map the road type to a weight (w_r), we experimented with different functions and found the following linear function to be efficient to implement and give good performance:

$$w_r = F(r_t) = 1.0 - c(r_t - 1) \tag{5}$$

where c is a constant discount factor and r_t is the road type index (≥ 1) with 1 being the lowest major type.

Note that if the user is moving on a side road, using road weights may lead to a wrong estimate. However, this happens with a lower probability compared to moving on a major road as discussed before.

3) Transition Probability: The transition probability is the probability of moving to the next road segment (state $s_{j,t}$) given that the current state is road segment $s_{i,t-1}$. The transition probability can be estimated as a function based on the difference between the distance on the input trajectory and the distance between the projections on the road segments. More formally, this distance difference (d_t) is given by the following equation:

$$d_t = \|\text{dist}(z_t, z_{t-1}) - \text{dist}(s_{j,t}, s_{i,t-1})\| \tag{6}$$

where $\text{dist}(z_t, z_{t-1})$ is the geodesic distance between the location observations z_t and z_{t-1} and $\text{dist}(s_{j,t}, s_{i,t-1})$ is the geodesic distance between the projections of z_t and z_{t-1} on the $s_{j,t}$ and $s_{i,t-1}$ roads respectively. The smaller this distance (d_t) is, the larger the transition probability should be. We assume that this distance difference has an exponential distribution [32]. However, with the much noisier locations in our case and using an online incremental algorithm, it gives a large number of false transitions.

To reduce the abrupt transitions, we bias the transition probability to favor staying on the current road. This is achieved by weighting the exponential probability according to whether the transition is on the same road segment or not. Therefore, the transition probability used in *SnapNet* is:

$$p(s_{j,t}|s_{i,t-1}) = \begin{cases} p_{\text{same}} \cdot \frac{1}{\beta} e^{-\frac{d_t}{\beta}} & \text{when } s_{j,t} = s_{i,t-1} \\ (1 - p_{\text{same}}) \cdot \frac{1}{\beta} e^{-\frac{d_t}{\beta}} & \text{otherwise} \end{cases} \tag{7}$$

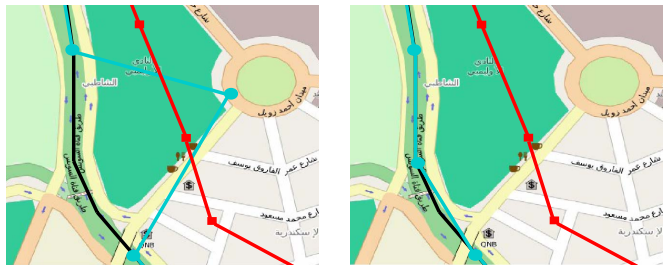
where $p_{\text{same}} \geq 0.5$ is the probability (weight) of staying on the same road between transitions.

Fig. 8 shows an example of a user trajectory with and without the transition weights.

4) The Initial State Distribution: Initially, all road segments selected by the *Candidate Extraction and Filtering* module are assigned equal weights. After each estimation step, the new initial state distribution is obtained as the product of the current state distribution by the transition probability matrix to the new candidate road segments (obtained by the *Candidate Extraction and Filtering* module). More formally

$$\pi_{t+1} = \pi_t . P_t \tag{8}$$

where π_t is the initial state distribution at time t and P_t is the state transition matrix at time t calculated as in Section IV-B3.



(a) (b)

Fig. 8. Effect of using transition weights. Using the unweighted transition, the map matcher deviates from the actual path as it favors the nearest road. (a) Unweighted transition. (b) Weighted transition.

TABLE II
SAMPLES OF THE TRACES USED IN THE EVALUATION

Trace ID	Len. (Km)	#Samp.	Time (mins)	GT Seg.	Major%	Prec.	Recall
1	121.8	87	81.517	35	94.12%	0.8	0.806
2	116.5	60	68.983	12	100%	0.997	0.934
3	94.9	112	55.9	25	92 %	0.784	0.913
4	54.2	62	26.883	24	100%	0.992	0.931
5	10.3	75	30.417	6	71.5 %	0.995	0.992
6	3.9	59	22.216	10	0 %	0.745	0.875
7	4.6	73	34.88	5	0 %	0.845	0.984

V. EVALUATION

In this section, we show the evaluation results of SnapNet. We tested SnapNet on real data traces collected from two cities in Egypt. Our data consists of 407 km of car driving traces. These traces cover different representative cases including: urban, rural, main and side-roads, among others (see Table II). The data was collected using different Android phones including Samsung Galaxy S3 Mini, S4, and S5 phones as well as an LG Nexus 4. We used OpenStreetMaps for the road network information.

We implemented the system using Java on a Lenovo Ideapad S400 laptop with 4 GB RAM, 1.70 GHz core i5 processor, running the Windows 8 OS. To get cellular-based location estimates, we use the Google Android Location API with WiFi turned off. This way, the returned location is estimated based on the associated cell tower information only as the used phones do not provide the neighbouring cell towers information. This provides a low-accuracy location with a median error of 2 km, as shown in Fig. 1. The location update is reported when a cell-tower change occurs, which happens at a rate of 2 minutes on average. For ground truth, we also collected accurate location data at high sampling rate (1 sample/sec) using a combined GPS/GLONASS receiver.

The rest of this section is organized as follows: We start by describing our evaluation methodology and metrics. Then, we show the effect of the different filters on the system accuracy. After that, we quantify the effect of the different HMM parameters on performance. Finally, we compare *SnapNet* with three baseline HMM map matching techniques.

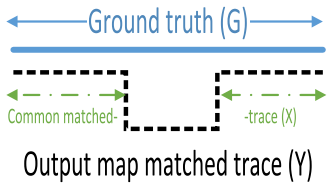


Fig. 9. Definition of precision X/Y and recall X/G.

TABLE III
DEFAULT PARAMETER VALUES

Param.	Def. val.	Tested Range	Meaning
w_s	7	$0-20$	Window size of the speed filter.
w_a	5	$0-20$	Window size of the α-trimmed filter.
α_a	0.2	$0-0.5$	α value of the α-trimmed filter.
p_{same}	0.75	$0.5-0.9$	Bias for trans. prob. to the same road.
c	0.08	$0-0.1$	Disc. factor for the linear road weigh. fn.

A. Evaluation Methodology and Metrics

One of the challenges in evaluating a map matching system is to construct the ground truth path. Manual construction of the path is labor intensive. Instead, we apply map matching to the high accuracy ground truth data collected using the GPS/GLONASS receiver to obtain the actual driving path. It has been shown in literature that map matched GPS traces with a one second sampling rate almost yields the ground truth path [32].

To evaluate *SnapNet*, first we find the common matching sequence of roads X between the map matched output trajectory Y and the ground-truth trajectory G. Based on this sequence, we compute two evaluation metrics: precision and recall [32] (see Fig. 9). Precision is defined as the ratio between the distance traversed on the matching sequence X and the total distance of the map matched trajectory Y. Recall is defined as the ratio between the distance traversed on the matching sequence X and the total distance of the ground truth trajectory G.

In the following subsections, we show the effect of different parameters on the system performance. The default parameters are shown in Table III.

B. Effect of Filters

1) Speed Filter: Fig. 10 shows the effect of changing the window size of the speed filter on the overall accuracy. At $w_s = 0$, the speed filter is turned off. At small values of w_s, the accuracy is low due to the effect of noisy points. Increasing the window size gradually increases the accuracy till it reaches a maximum value then decreases again due to spanning multiple speeds within the window. We use the optimal value at $w_s = 7$.

2) α-Trimmed Filter: The α-trimmed filter has two parameters: the window size and α. Fig. 11(a) shows the effect of changing the window size (w_a). The figure shows that at a small window size, the system has low accuracy as the effect of noise is not completely removed. As the window size increases, the accuracy increases, then it decreases again due to the loss of information introduced by smoothing. At $w_a = 5$, the system has the best accuracy in terms of precision and recall.

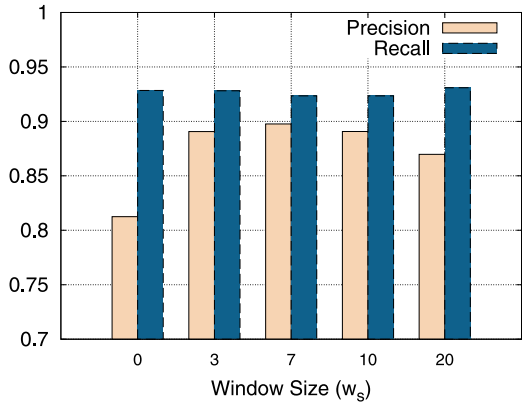


Fig. 10. Effects of the speed filter on the system accuracy.

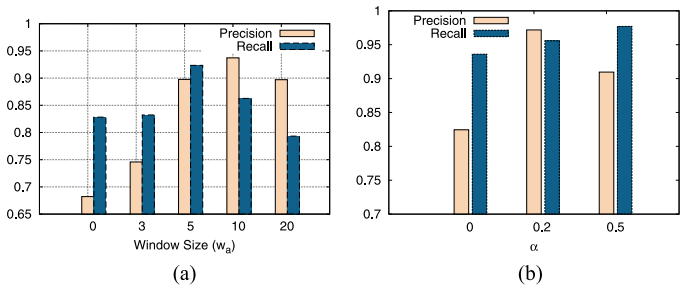


Fig. 11. Effects of the α-trimmed filter parameters on accuracy. (a) Window size w_a. (b) α.

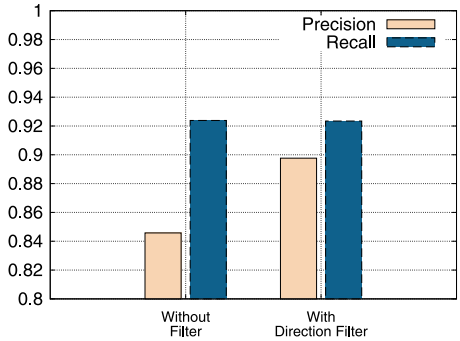


Fig. 12. Effects of the direction filter on the performance.

Fig. 11(b) shows the effect of α on performance. The figure shows that a value of $\alpha = 0.2$ balances the performance between the average and median filters and hence gives the best accuracy.

3) Direction Filter: Fig. 12 shows the effect of the direction filter on accuracy. The figure shows that the filter increases the precision by 7% while maintaining the recall value.

C. Effect of HMM Parameters

1) Road Weighting: Fig. 13 shows the effect of changing the c parameter of the linear weighting function discussed in Section IV-B2. The figure shows that increasing the differentiation between roads leads to better accuracy till it saturates at $c = 0.08$. Using road weights simultaneously enhances precision by 15.4% and recall by 13.6%.

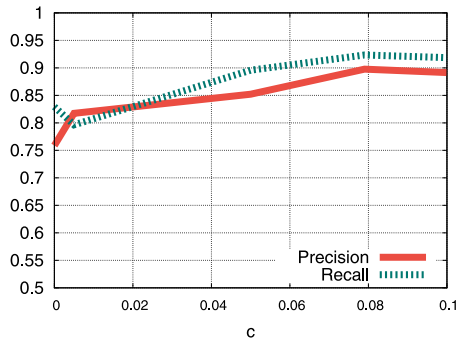


Fig. 13. Effects of the road-type weighting factor on the performance.

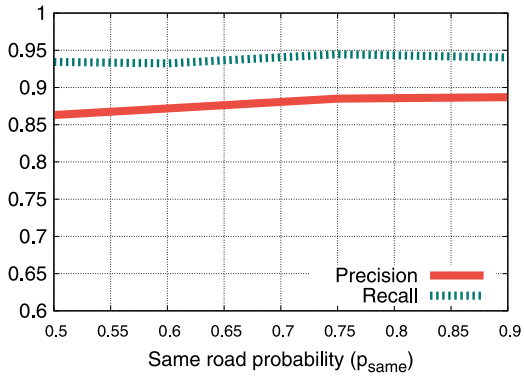


Fig. 14. Effects of same road probability biasing on the performance.

2) Road Transition Weighting: Fig. 14 shows the effect of biasing the transition probability of staying on the same road discussed in Section IV-B3. The value at $p_{\text{same}} = 0.5$ indicates no bias. The figure shows that favoring staying at the same road enhances both the recall and precision concurrently. The performance saturates at $p_{\text{same}} = 0.75$. We note that while the effect of the transition weights on the overall accuracy is not large, due to amortizing the advantage of the weighting over the entire trace, we can see that the transition weights are still effective for real-time path visualization as it removes the small jumps (see Fig. 8).

D. Effect of Localization Error

In this section, we evaluate the effect of the error in the input locations on the map matcher performance. For this, we interpolate between the ground truth location (zero error) and the cellular-based location (maximum error). More specifically, the test location coordinates $\text{loc}_{\text{test}} = q \times \text{loc}_{\text{GPS}} + (1 - q) \times \text{loc}_{\text{Network}}$, where q is a location quality factor ranging from 0 and 1 that determines how close the test location is to the ground truth (no errors).

Fig. 15 shows the effect of changing the location quality parameter q on *SnapNet* precision and recall. The figure shows that increasing the quality of the input locations leads to better accuracy. Nevertheless, even with the lowest quality, i.e., $q = 0$, SnapNet can achieve more than 90% precision and recall.

E. Overall System Performance

We compare the overall accuracy of *SnapNet* with three other systems: (a) NearRd: Snapping to the nearest road to

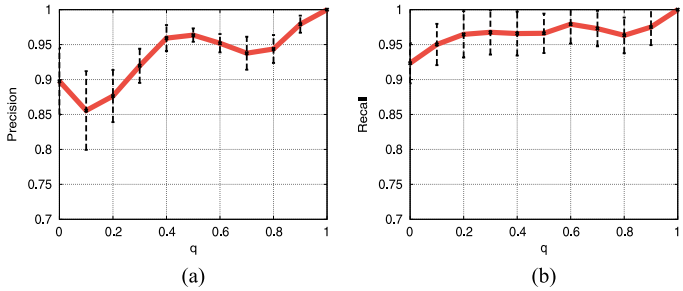


Fig. 15. Effects of the input location quality factor q. (a) Precision. (b) Recall.

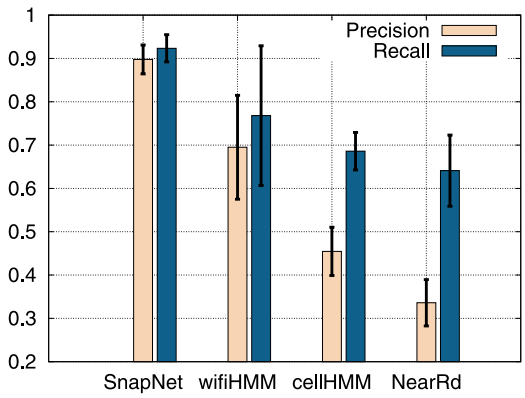


Fig. 16. Accuracy comparison between *SnapNet* and other map-matching techniques.

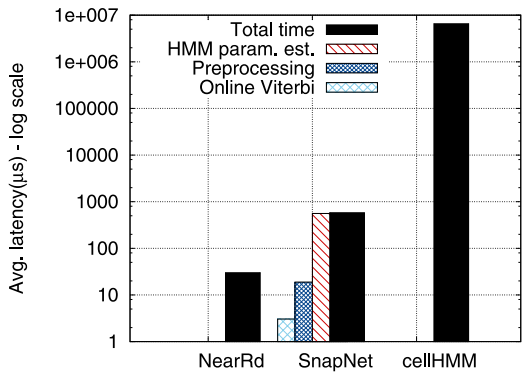


Fig. 17. Running time of the components of *SnapNet*.

show the effect of map-matching. (b) cellHMM: A plain HMM technique that map-matches cellular-based locations without using any of the additional filters or our HMM modifications. (c) wiHMM: The plain HMM technique using locations from a WiFi localization technique. Fig. 16 shows that *SnapNet* has the highest accuracy of all techniques. It achieves an enhancement of 97.4% and 34% in precision and recall respectively over the plain HMM with cellular data, highlighting the combined advantage of the different modules of the system.

Fig. 17 shows the average latency per input location for each module (log scale). The latency is defined as the time taken from the new point arrival in the system until its best match is calculated. This time includes adding interpolated points and map-matching them, calculating the HMM parameters of the new point, adding a new row to the Viterbi matrix, and estimating the best match for the new point. The figure shows that the overall latency of *SnapNet* is about 0.58 ms. Most

TABLE IV
EFFECTS OF USING ROAD WEIGHTS ON TYPICAL AND NON-MAJOR ROAD TRACES

	Non-major Roads		Typical	
	Prec.	Recall	Prec.	Recall
W. weights	0.71	0.70	**0.90**	**0.92**
W/o weights	0.83	0.82	0.76	0.83

of the running time is consumed by the HMM parameters estimation, which involves calculating complex functions over a large number of candidates, due to the large error of the input cellular-based locations. The online Viterbi algorithm has the least running time as it runs on the unfiltered points only. Compared to the window-based plain HMM technique, the online Viterbi algorithm and the incremental HMM module reduce the latency by more than multiple orders of magnitude (latency of milliseconds compared to seconds). The NearestRoad technique has the least running time as it only requires calculating the distance between each point and its candidate roads then selecting the road with the least distance.

F. Effect of Violating Assumptions

Since *SnapNet* gives higher observation probability to major roads (see Section IV-B2), it is likely that its performance will degrade when the whole trip is traveled over non-major roads. In order to measure this degradation, we evaluate the system with traces over non-major roads only and typical traces mostly traveled on major roads. Table IV summarizes the results. For the non-major roads case, adding the road weights (when this assumption is not correct) decreases the accuracy by about 12%. The overall performance in this case is still better than other state-of-the-art systems that use the cellular data for localization due to the other modules of *SnapNet*.

Observing the second row of Table IV, when the weights are not used, there is a slight difference in accuracy between major and non-major roads, which reflects the system worst-case performance. Using road weights, however, leads to a significant enhancement in performance for the **typical case** of driving over major roads most of the time (see Section IV-B2).

VI. CONCLUSION

We presented *SnapNet*, a real-time map matcher for noisy cellular-based trajectory traces. *SnapNet* is unique in targeting mainstream cell phones which can only provide its associated cell-info for localization systems; opening up new possibilities for location-based services from both network and client side. We provided the *SnapNet*'s system architecture and the different filtering and preprocessing modules that help it reduce the noise in the input data and handle the data sparsity. We also presented the details of our novel incremental HMM algorithm that leverages the digital map information and different heuristics to enhance the estimation accuracy and maintain real-time operation and online Viterbi to provide online map matching.

Evaluation of *SnapNet* on actual cellular-based traces collected using Google's network-based localization, shows that *SnapNet* can achieve a map matching precision and recall of more than 90% which maps to over 97% and 34% enhancement in precision and recall respectively when compared to traditional HMM map matching algorithms. In addition, *SnapNet* has an average running time of 0.58 ms per location estimate, which is three orders of magnitudes better than traditional window-based HMM algorithms, making it suitable for online applications.

REFERENCES

[1] A. Al-Husseiny and M. Youssef, "RF-based traffic detection and identification," in *Proc. IEEE VTC—Fall*, 2012, pp. 1–5.

[2] H. Alt, A. Efrat, G. Rote, and C. Wenk, "Matching planar maps," in *Proc. ACM-SIAM Symp. Discr. Algorithms*, 2003, pp. 589–598.

[3] H. Aly, A. Basalamah, and M. Youssef, "LaneQuest: An accurate and energy-efficient lane detection system," in *Proc. IEEE PerCom*, 2015, pp. 163–171.

[4] H. Aly, A. Basalamah, and M. Youssef, "Robust and ubiquitous smartphone-based lane detection," *Pervasive Mobile Comput.*, vol. 26, pp. 35–56, 2016.

[5] H. Aly, A. Basalamah, and M. Youssef, "Map++: A crowd-sensing system for automatic map semantics identification," in *Proc. IEEE SECON*, 2014, pp. 546–554.

[6] H. Aly and M. Youssef, "semMatch: Road semantics-based accurate map matching for challenging positioning data," in *Proc. ACM SIGSPATIAL GIA*, 2015, pp. 1–10.

[7] H. Aly and M. Youssef, "Dejavu: An accurate energy-efficient outdoor localization system," in *Proc. ACM SIGSPATIAL*, 2013, pp. 154–163.

[8] M. Alzantot and M. Youssef, "UPTIME: Ubiquitous pedestrian tracking using mobile phones," in *Proc. IEEE WCNC*, 2012, pp. 3204–3209.

[9] M. Bader, *Space-Filling Curves—An Introduction with Applications in Scientific Computing*, ser. Texts in Computer Science and Engineering. Berlin, Germany: Springer-Verlag, 2013.

[10] J. Bednar and T. Watt, "Alpha-trimmed means and their relationship to median filters," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 32, no. 1, pp. 145–153, Feb. 1984.

[11] D. Bernstein and A. Kornhauser, "An introduction to map matching for personal navigation assistants," New Jersey Inst. Technol., Newark, NJ, USA, Tech. Rep., 1996.

[12] C. A. Blazquez and A. P. Vonderohe, "Simple map-matching algorithm applied to intelligent winter maintenance vehicle data," *Transp. Res. Rec., J. Transp. Res. Board*, vol. 1935, pp. 68–76, 2005.

[13] J. Bloit and X. Rodet, "Short-time Viterbi for online HMM decoding: Evaluation on a real-time phone recognition task," in *Proc. IEEE ICASSP*, 2008, pp. 2121–2124.

[14] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk, "On map-matching vehicle tracking data," in *Proc. VLDB*, 2005, pp. 853–864.

[15] Y. Cui and S. S. Ge, "Autonomous vehicle positioning with GPS in urban canyon environments," *IEEE Trans. Robot. Autom.*, vol. 19, no. 1, pp. 15–25, Feb. 2003.

[16] N. Ekiz, T. Salih, S. Küçüköner, and K. Fidanboylu, "An overview of handoff techniques in cellular networks," *Int. J. Inf. Technol.*, vol. 2, pp. 132–136, 2005.

[17] M. Elhamshary and M. Youssef, "CheckInside: A fine-grained indoor location-based social network," in *Proc. ACM UbiComp*, 2014, pp. 607–618.

[18] M. Elhamshary and M. Youssef, "SemSense: Automatic construction of semantic indoor floorplans," in *Proc. IPIN*, 2015, pp. 1–11.

[19] C. Y. Goh, J. Dauwels, N. Mitrovic, M. T. Asif, A. Oran, and P. Jaillet, "Online map-matching based on hidden Markov model for real-time traffic sensing applications," in *Proc. IEEE ITSC*, 2012, pp. 776–781.

[20] J. S. Greenfeld, "Matching GPS observations to locations on a digital map," presented at the 81st Annual Meeting Transportation Research Board, Washington, DC, USA, 2002.

[21] S. Guha *et al.*, "Autowitness: Locating and tracking stolen property while tolerating GPS and radio outages," *ACM Trans. Sens. Netw.*, vol. 8, no. 4, pp. 29–42, 2012.

[22] A. Guttman, "R-trees: A dynamic index structure for spatial searching," in *Proc. ACM SIGMOD*, 1984, pp. 47–57.

[23] B. Hummel, "Map matching for vehicle guidance," in *Dynamic and Mobile GIS: Investigating Space and Time*. Boca Raton, FL, USA: CRC Press, 2006.

[24] M. Ibrahim and M. Youssef, "A hidden Markov model for localization using low-end GSM cell phones," in *Proc. IEEE ICC*, 2011, pp. 1–5.

[25] E. Isaacson, *Analysis of numerical methods*. New York, NY, USA: Dover, 1994.

[26] N. Kassem, A. E. Kosba, and M. Youssef, "RF-based vehicle detection and speed estimation," in *Proc. IEEE VTC—Spring*, 2012, pp. 1–5.
[27] J. Krumm, J. Letchner, and E. Horvitz, "Map matching with travel time constraints," in *Proc. SAE World Congr.*, 2007, pp. 1–7.
[28] S. Lee, K. Ho, G. Huazhi, and Y. Yung, "Base station association in wireless cellular networks: An emulation based approach," *IEEE Trans. Wireless Commun.*, vol. 11, no. 8, pp. 2720–2729, Aug. 2012.
[29] X. Li, M. Li, W. Shu, and M. Wu, "A practical map-matching algorithm for GPS-based vehicular networks in Shanghai urban area," in *Proc. IET CCWMSN*, 2007, pp. 454–457.
[30] R. Mohamed, H. Aly, and M. Youssef, "Accurate and efficient map matching for challenging environments," in *Proc. ACM SIGSPATIAL*, 2014, pp. 401–404.
[31] N. Mohssen, R. Momtaz, H. Aly, and M. Youssef, "It's the human that matters: Accurate user orientation estimation for mobile computing applications," in *Proc. ICST Mobiquitous*, 2014, pp. 70–79.
[32] P. Newson and J. Krumm, "Hidden Markov map matching through noise and sparseness," in *Proc. ACM SIGSPATIAL*, 2009, pp. 336–343.
[33] B. P. Phuyal, "Method and use of aggregated dead reckoning sensor and GPS data for map matching," in *Proc. Int. Tech. Meet. Satell. Division ION GPS*, 2002, pp. 430–437.
[34] M. A. Quddus, W. Y. Ochieng, and R. B. Noland, "A general map matching algorithm for transport telematics applications," *GPS Solutions*, vol. 7, no. 3, pp. 157–167, 2003.
[35] L. Rabiner and B.-H. Juang, "An introduction to hidden Markov models," *IEEE ASSP Mag.*, vol. 3, no. 1, pp. 4–16, Jan. 1986.
[36] D. Schultes, "Route planning in road networks," M.S. thesis, Inst. Theoretical Comput. Sci., Univ. Karlsruhe, Karlsruhe, Germany, 2008.
[37] R. Šrámek, B. Brejová, and T. Vinař, "On-line Viterbi algorithm and its relationship to random walks," *arXiv preprint arXiv:0704.0062*, 2007.
[38] A. Thiagarajan *et al.*, "VTrack: Accurate, energy-aware road traffic delay estimation using mobile phones," in *Proc. ACM SenSys*, 2009, pp. 85–98.
[39] A. Thiagarajan, L. Ravindranath, H. Balakrishnan, S. Madden, and L. Girod, "Accurate, low-energy trajectory mapping for mobile devices," in *Proc. USENIX*, 2011, pp. 1–14.
[40] H. Wang *et al.*, "WheelLoc: Enabling continuous location service on mobile phone for outdoor scenarios," in *Proc. IEEE INFOCOM*, 2013, pp. 2733–2741.
[41] C. E. White, D. Bernstein, and A. L. Kornhauser, "Some map matching algorithms for personal navigation assistants," *Transp. Res. C, Emerging Technol.*, vol. 8, no. 6, pp. 91–108, Feb.–Dec. 2000.
[42] J. Yang, S. Kang, and K. Chon, "The map matching algorithm of GPS data with relatively long polling time intervals," *J. East Asia Soc. Transp. Stud.*, vol. 6, pp. 2561–2573, 2005.

Reham Mohamed received the B.Sc. degree in computer and systems engineering, in 2013, from Alexandria University, Alexandria, Egypt, where she is currently working toward the M.Sc. degree in computer science.

She is currently with the Wireless Research Center, Egypt-Japan University of Science and Technology, Alexandria. Her research interests include mobile computing, location determination systems, and machine learning.

Heba Aly received the B.Sc. and M.Sc. degrees in computer science from Alexandria University, Alexandria, Egypt, in 2011 and 2015, respectively. She is currently working toward the Ph.D. degree with the University of Maryland, College Park, MD, USA.

Her research interests include mobile computing, location determination technologies, pervasive computing, and data mining.

Ms. Heba is the recipient of the 2014 Common Market for Eastern and Southern Africa (COMESA) Innovation Award and the 2013 ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems Best Paper Award, among others.

Moustafa Youssef is an Associate Professor with Alexandria University, Alexandria, Egypt and with Egypt-Japan University of Science and Technology, Alexandria. He has fifteen issued and pending patents. His research interests include mobile wireless networks, mobile computing, location determination technologies, pervasive computing, and network security.

Prof. Youssef is an Associate Editor for the Association for Computing Machinery (ACM) Transactions on Spatial Algorithms and Systems and was an Area Editor of the ACM Mobile Computing and Communications Review. He has served on the organizing and technical committees of numerous prestigious conferences. He was the recipient of the 2003 University of Maryland Invention of the Year award, the 2010 The World Academy of Sciences–African Academy of Sciences–Microsoft Award for Young Scientists, the 2012 Egyptian State Award, the 2013 and 2014 Common Market for Eastern and Southern Africa (COMESA) Innovation Award, multiple Google Research Awards, the 2013 ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems Best Paper Award, among others. He is also an ACM Distinguished Scientist.