Abstract—Recent years witnessed popular use of various mobile devices, e.g., smart phones, vehicle networks and wearable watches. Such mobile devices generate massive trajectory data, and literature have proposed various algorithms to leverage the trajectory data for map inference. Unfortunately, such algorithms are hard to achieve both high map quality and computation efficiency. In this paper, we propose a solution framework to infer road network maps with high quality and efficiency. The key of our map inference is to divide map extent into smaller cells and maintain a binary cell-trajectory matrix. The binary matrix determines whether or not a trajectory passes a cell. We infer the importance of each cell from the matrix using a popular topic model (e.g., LDA [13] and pLSA [8]). Based on such computed importance, we next infer representative points and road segments to derive a road network map. Our extensive experiments on real data sets verify that the proposed inference algorithm can achieve higher map quality and meanwhile $1.5 \times$, $6.8 \times$ and $280 \times$ shorter running time, when compared with state of the arts including three representative work [4], [7], [14].

I. Introduction

Recent years witnessed popular use of various mobile devices, e.g., smart phones, vehicle networks and wearable watches. Such devices are frequently equipped with GPS sensors to record their geographical positions, generating massive spatial trajectory data. The trajectory data has been widely used to infer and update (existing) road network maps. For example, in urban computing application, inferred maps directly benefit those areas where no manually produced maps exist, and are also useful to timely update existing maps (e.g., detecting changes or pinpointing errors in such existing maps) [3], [6].

Though many previous work including [4], [7], [14] has proposed map inference algorithms, such algorithms typically involve the trade-off between map quality and inference efficiency. In particular, when processing massive trajectories, some work may infer highly qualified maps but at cost of very long running time. Others can offer efficient inference algorithms yet sacrificing map quality. It is hard to achieve both highly qualified maps and efficient inference.

In this paper, we propose a solution framework to infer road network maps. In our framework, a map consists of two important parts: (i) representative points and (ii) road segments (a.k.a lines). The key of our map inference is to leverage a binary cell-trajectory matrix. That is, by dividing map extent into smaller disjointed grid cells, we maintain a boolean matrix to determine whether or not a trajectory passes a cell. We infer the importance of each cell from the matrix using a popular topic model (e.g., LDA [8] and pLSA [13]). Based on such computed importance, we can carefully select some cells as representative points in the inferred map. We next construct road graphs. By computing shortest paths on the road graphs, we can infer road segments in the map. After that, we refine the points and road segments to form a final road network map.

As a summary, we make the following contributions.

- Our solution provides higher quality than state of the art. It is mainly because the used topic model can cluster the representative cells together to form road segments.
- Besides the high map quality, our proposed solution can greatly outperform state of the art by much shorter running time. The high efficiency is mainly caused by the fact: our inference solution works mainly based on the computation granularity of divided cells. The amount of cells is significantly fewer than the number of GPS points. Previous work instead performs map inference using such GPS points as the computation granularity, thus incurring long running time.
- Extensive evaluation on real data sets verifies the advantages of the proposed map inference solution over state of the art in terms of both map quality and computation efficiency. For example, the proposed inference algorithm can achieve reasonably higher map quality than three representative work [4], [7], [14] meanwhile with $1.5 \times$, $6.8 \times$ and $280 \times$ shorter running time.

The rest of the paper is organized as follows. Section
II first gives an overview. Next, Section III describes the detail to infer road network maps. After that, we perform the evaluation in Section IV, and review related work in Section V. Finally Section VI concludes the paper.

Table I summarizes the main symbols and associated meanings used in the paper.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$M$</td>
<td>Road network map</td>
</tr>
<tr>
<td>$T, t, p$</td>
<td>Trajectory database, a trajectory and a trajectory point</td>
</tr>
<tr>
<td>$C, c_i$</td>
<td>Set of all divided cells, a cell</td>
</tr>
<tr>
<td>$X$</td>
<td>Cell-trajectory matrix</td>
</tr>
<tr>
<td>$X_{i,j}$</td>
<td>an element w.r.t trajectory $t_j$ and cell $c_i$</td>
</tr>
<tr>
<td>$\tau_k,</td>
<td>K</td>
</tr>
<tr>
<td>$\Phi_k, \rho_k, M_k$</td>
<td>road vector, road rank, road graph w.r.t road $\tau_k$</td>
</tr>
<tr>
<td>$N(c, h)$</td>
<td>$h$-hop neighbors of cell $c$</td>
</tr>
<tr>
<td>$w_{c,c'}$</td>
<td>Edge weight between two cells $c$ and $c'$</td>
</tr>
<tr>
<td>$cw$</td>
<td>cell width</td>
</tr>
<tr>
<td>$min$</td>
<td>threshold of road rank</td>
</tr>
</tbody>
</table>

**II. Overview**

In this section, we first give the problem definition, next introduce the solution overview, and finally highlight the technique idea.

**A. Problem Definition:**

**DEFINITION 1. (TRAJECTORIES) Consider a trajectory database $T$. Each trajectory $t \in T$ consists of a set of quadruples $p = \langle id, x, y, s \rangle$. Here, $id$ is a unique ID of trajectory $t$, and $x$ (resp. $y$) represents the longitude (resp. latitude) of a specific location at timestamp $s$. Typically the quadruples $p \in t$ are sorted by ascending order of timestamp $s$.**

**PROBLEM 1. (ROAD NETWORK MAP INFERENCE) Road network map inference problem is to infer a road network map $M$ on which those trajectories in a given trajectory database $T$ are moving. A road network map $M$ is comprised of (1) a set of representative points and (2) intersected road segments.**

In Problem 1, we have to tackle two following challenges. (i) Trajectory data could contain noisy GPS points, leading to falsely inferred but non-existing road segments. How to tolerate the noisy GPS points and avoid falsely inferred road segments is non-trivial. (ii) Due to the disparity of trajectories made on different road segments, it is possible that a very few number of real GPS points in trajectories truly appear in a road segment. It is hard to differentiate such real GPS points from noisy ones.

Fig. 1 shows an example of the trajectories from Chicago campus shuttles dataset [1]. The area highlighted by red dashed square contains high buildings and involves plenty of noisy GPS points. In addition, the area highlighted by a red dashed oval shows some rarely traveled (only by one or two trajectories) roads. Fig. 1 clearly illustrates the above challenges.

**B. Solution Overview:** We give the basic idea to infer the road network map $M$ as follows (see Fig. 2). Consider that the map $M$ is comprised of a set of (representative) points and lines (i.e., road segments). We would like to first infer (representative) points appearing in road segments, and connect such points (using shortest paths) as lines (road segments). Such points and lines become the road map $M$. We highlight the steps as follows.

First, by dividing map extent into disjoint smaller square cells, we maintain the membership between divided cells and trajectories by a cell-trajectory matrix $X$. The binary element, either 0, in the matrix $X$, depends upon whether or not a trajectory passes a cell. We next apply the topic model, e.g., Latent Dirichlet Allocation (LDA) [8] or Probabilistic Latent Semantic Analysis (pLSA) [13], onto the matrix $X$, and thus generate road vectors. The vector elements indicate the importance of cells. Those important cells are then selected as candidate cells which will be used to select the following representative points.

Second, with the help of road vectors, we construct weighted road graphs and carefully select a small amount of representative points among the above candidate points. Such points can indicate the skeleton of the road network maps. By performing shortest paths between representative points on road graphs, we can infer road segments, which form a map skeleton $M^-$. After that, we refine the $M^-$ to form the final map $M$. 

![Fig. 2. Framework Overview](image-url)
C. Main Technical Idea:

**Topic Model**: Probabilistic topic models [8], [13] have been successfully used to extract hidden thematic structure in large archives of documents. In the topic model, each document in a corpus exhibits multiple topics, and each word in a document supports a certain topic. Given all the words of each document in a corpus as observations, we can train a topic model to infer the hidden thematic structure behind the observations. LDA and pLSA are two most widely used topic models. The intuition behind these two models is that documents are represented as random mixtures over latent topics and each topic is characterized by a distribution over words.

\[
P(c|t) = \sum_{z=1}^{K} P(c|z)P(z|t) \quad (2.1)
\]

\[
P(c|t, \phi, \theta) = \sum_{z=1}^{K} P(c|z, \phi)P(z|t, \theta) \quad (2.2)
\]

For pLSA, the probability of word \(c\) from a document \(t\) is calculated by Equation (2.1) by EM algorithm to learn \(P(c|z)\) and \(P(z|t)\). LDA infers probability of word \(c\) from document \(t\) by Equation (2.2) by using the inference techniques including Gibbs sampling and expectation propagation. Different from pLSA, there is a multinomial distribution \(\theta_d\) over topics for document and a topic-specific multinomial \(\phi_z\) in LDA. To state our problem more clearly, we use a document-word matrix \(X\) denoting \(P(c|z)\), a word-topic matrix \(\Phi\) denoting \(P(c|t)\) and a topic-document matrix \(\Theta\) denoting \(P(z|t)\).

Assume there are \(K\) topics \(\tau_0, ..., \tau_{K-1}\), a set of words \(C\) and a set of documents \(T\), a topic model receives a document-word matrix \(X|C|\times|T|\) as input and generate two output matrices: \(\Phi|C|\times|K|\) as word-topic matrix and \(\Theta|K|\times|T|\) as topic-document matrix. In the word-topic matrix \(\Phi|C|\times|K|\), an element \(\Phi_{ct}\) denotes the probability of word \(c\) belonging to topic \(t\); in the topic-document matrix \(\Theta|K|\times|T|\), an element \(\Theta_{\tau t}\) denotes the proportion of topic \(\tau\) in document \(t\). See [8], [13] for a detailed discussion of LDA and pLSA. Either topic model can be generally applied to our framework.

**Intuition of Using Topic Model for Map Inference**: To understand our technical solution, we highlight the intuition of using topic model (See Table II). When a document contains multiple words and a trajectory goes through multiple cells, we equivalently map the concepts of document and word in the topic model to trajectory and cell in our problem, respectively.

Next, in the topic model, a document is with multiple hidden topics and multiple words together decide a topic; equivalently in our problem, a trajectory could involve multiple road segments, and multiple cells are connected together to form a road segment. Thus, we can again map the road segment in our problem to the topic concept in topic model. It makes sense when we consider the following scenario. Those cells in road intersections in our problem could appear in different road segments. This is equivalent to the polysemy phenomenon in topic model: a word indicates significantly different meanings in various topics.

<table>
<thead>
<tr>
<th>Document</th>
<th>Trajectory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Word</td>
<td>Cell</td>
</tr>
<tr>
<td>Topic</td>
<td>Road Segment</td>
</tr>
<tr>
<td>Polysemy</td>
<td>Cells in intersected road segments</td>
</tr>
<tr>
<td>A doc. with topics</td>
<td>A traj. passing road segments</td>
</tr>
</tbody>
</table>

**TABLE II**

**ANALOGY FROM TRAJECTORY-CELL MODEL TO TRADITIONAL DOCUMENT-WORD MODEL**

As a summary, we map the concepts of document, word and topic in topic model to trajectory, cell and road segment (which means a combination of multiple roads) defined in our problem. Based on the intuition, Section III adopts the topic models on the binary cell-trajectory matrix \(X\) and computes the importance of each cell. The computed importance becomes the base to infer road network maps.

### III. Road Network Inference

In this section, we first describe two important data structures, namely **cell-trajectory matrix** and **Road graph**, and next present the algorithm used to infer road network maps \(M\).

**A. Cell-trajectory matrix**: We first preprocess a trajectory database \(T\) as follows. By dividing the map extent into square cells of equal size (denote the cell width to be \(cw\)), we have a set \(C\) of cells. With the cells in set \(C\), we transform trajectories into a binary cell-trajectory matrix \(X|C|\times|T|\), where \(|C|\) is the number of cells and \(|T|\) is the number of trajectories.

**DEFINITION 2.** [Cell-trajectory matrix] Consider a trajectory database \(T\) and cell set \(C = \{c_0, c_1, ..., c_{n-1}\}\). A cell-trajectory matrix \(X|C|\times|T|\) is a boolean matrix consisting of \(|C|\) rows and \(|T|\) columns. The matrix element \(X_{i,j}\) is defined as follows:

\[
X_{i,j} = \begin{cases} 
1 & \text{if trajectory } t_j \in T \text{ passes cell } c_i \in C \\
0 & \text{otherwise}
\end{cases}
\]

Fig. 3 shows an example of map division. In this figure, the 5 trajectories \(t_0...t_4\) are moving on the two intersected roads \(R_0\) and \(R_1\). We divide the map into \(4 \times 5 = 20\) cells \(c_0...c_{19}\). Based on the map division, the associated binary matrix \(X\) is given by the left-most matrix in Fig. 4 consisting of 20 rows and 5 columns.
Beyond the condition above (i.e., trajectory \( t_j \in T \) passes cell \( c_i \in C \)), we set a stricter condition as follows. When \( t_j \) passes \( c_i \), \( t_j \) then divides \( c_i \) into two sub-areas. In an ideal case, we expect that \( t_j \) cuts the cell \( c_i \) into two sub-areas with roughly equal size. However, it is possible that a trajectory may only pass across a small portion of a cell. Generally, we define a parameter \( r \) (e.g., \( r = 1.5 \)) to make sure that the cut ratio between the size of two sub-areas is inside the range of \([1/r, r]\). As shown in Fig. 3(c), the trajectory \( t_1 \) divides \( c_9 \) into two portions with a division ratio much beyond the range \([1/1.5, 1.5]\). Thus, \( t_1 \) fails to pass \( c_9 \). With the stricter condition, we will use the center point of a cell to represent the trajectory segment passing the cell.

With a specific road number \( K \) (the way to tune this parameter will be discussed in Section 3.4), we adopt the topic model discussed on matrix \( X \) and learn two matrices: a cell-road matrix \( \Phi \) and a road-trajectory matrix \( \Theta \) (here roads indicate a set of continuous road segments). For a given road \( \tau_k \) (with \( 0 \leq k \leq K-1 \)), we have an associated column \( \Phi_k \) in matrix \( \Phi \). The column \( \Phi_k \) is a \( |C| \)-dimensional vector \( \Phi_k = (\Phi_{0,k}, \Phi_{1,k}, \ldots, \Phi_{|C|-1,k}) \), where \( \Phi_{i,k} \) is the probability of cell \( c_i \) in road \( \tau_k \) and \( \sum_{i=0}^{|C|-1} \Phi_{i,k} = 1 \). Next, in the road-trajectory matrix \( \Theta \), we also have an associated \( K \)-dimensional vector \( \Theta_j = (\Theta_{j,0}, \Theta_{j,1}, \ldots, \Theta_{j,K-1}) \), where \( \Theta_{j,k} \) indicates the proportion of road \( \tau_k \) in trajectory \( t_j \). For example, for trajectories \( t_0 \) and \( t_1 \), the road \( \tau_0 \) is with the largest element 0.8; for trajectory \( t_2 \), the road \( \tau_2 \) is with the largest element 0.6. Now, following the definition of road rank \( \rho_k \), we have \( \rho_{\tau_0} = 2.1, \rho_{\tau_1} = 1.9 \) and \( \rho_{\tau_2} = 1.0 \). Thus, among the three latent topics, \( \tau_0 \) is the one with the most significance, and \( \tau_2 \) is instead trivial.

Given the road-trajectory matrix \( \Theta \) in Fig. 4, we can find the proportion of the 3 road to the 5 trajectories. For example, for trajectories \( t_0 \) and \( t_1 \), the road \( \tau_0 \) is with the largest element 0.8; for trajectory \( t_2 \), the road \( \tau_2 \) is with the largest element 0.6. Now, following the definition of road rank \( \rho_k \), we have \( \rho_{\tau_0} = 2.1, \rho_{\tau_1} = 1.9 \) and \( \rho_{\tau_2} = 1.0 \). Thus, among the three latent topics, \( \tau_0 \) is the one with the most significance, and \( \tau_2 \) is instead trivial.

For illustration, Fig. 6 plots raw trajectories of our Shanghai dataset and 1st, 2nd, 5th, 50th, 100th road vectors.
computed by pLSA according to the sorted road ranks $\rho_k$. The cells with darker color represent a higher weight or importance. By checking a real Shanghai map, we find that the road segments of all top-5 ranks are with the inner ring road in Shanghai. Next, by comparing Fig. 6(c) and Fig. 6(d), we find that the two road segments are nearly duplicate. Thus, we would like to focus on the roads with higher importance (because the majority of trajectories are real GPS points). It is consistent with the classic road duplication problem [21]. Finally, in terms of data disparity, we can still find that a rarely visited road can produce a road, e.g., in Fig. 6(f).

**B. Road Graph:** For a specific road $\tau_k$, we need to build an associated road graph $M_k$. In the graph $M_k$, vertices are those cells relevant to this road (road segments), and edges denote the connectivity of cells. In the rest of this section, we first define *candidate cell*, then introduce the construction of $M_k$ with help of such candidate cells, and next present the inference of road segments by the shortest paths on $M_k$.

When processing a specific road $\tau_k$, we have an associated $|C|$-dimensional road vector $\Phi_k = (\Phi_{0,k}, \Phi_{1,k} \ldots \Phi_{|C|-1,k})$. Here, we normalize the elements $\Phi_{i,k}$ in road vector $\Phi_k$ by

$$\Phi_{i,k} = \frac{\Phi_{i,k} - \min(\Phi_k)}{\max(\Phi_k) - \min(\Phi_k)} \quad (3.3)$$

In the normalization above, $\min(\Phi_k)$ (resp. $\max(\Phi_k)$) indicates the minimal (resp. maximal) element value in vector $\Phi_k$. By default, we say that the element $\Phi_{i,k}$ is a normalized value.

In the vector $\Phi_k$, a cell $c_i$ is a *candidate cell* if its normalized element $\Phi_{i,k}$ satisfies $\Phi_{i,k} \geq \min$, where $\min$ is a minimal threshold. In this way, we prune those cells with low $\Phi_{i,k}$ which are passed by noisy GPS points. In our experiment, the threshold $\min = 0.1$ can comfortably prune the noisy cells.

**Construction of Road Graph $M_k$:** For each road $\tau_k$, we have a set of candidate cells $c$, which correspond to the vertices of $M_k$. Here, we use the center points of cells $c$ as the vertices of $M_k$. For each cell $c \in M_k$, we define the $h$-hop neighbors of cell $c$ as follows.

**Definition 3.** [h-hop neighbors]: For two vertices $c \neq c' \in M_k$, the hop distance $\text{dist}(c, c')$ is the nearest hop count between $c$ and $c'$ in the map extent divided by cells. The $h$-hop neighbors $N(c, h)$ of $c$ is a set of all candidate cells $c' \in M_k$ satisfying the condition $0 < \text{dist}(c, c') \leq h$.

Introducing the $h$-hop neighbors $N(c, h)$ is motivated as follows. Consider that the GPS errors typically follow a normal distribution [16]. Thus, the GPS points that are truly located inside cell $c$ could shift to another cell $c'$. To this end, we introduce the $N(c, h)$ in order to cover the possible cells besides the cell $c$ itself.

With the definition $N(c, h)$, we have a set of neighbours $c' \in N(c, h) = 1$ of a cell $c$. Such neighbours indicate that the GPS points in $c$ have high chance to shift to those in $c'$. Thus, we use an edge to connect a cell $c$ to the neighbours $c'$. For an edge between $c$ and $c'$, we compute the edge weight $w_{c,c'}$ by

$$w_{c,c'} = (1 - \Phi_{c,k})^2 + (1 - \Phi_{c',k})^2 \quad (3.4)$$

Fig. 7 shows an example road graph. In this Figure, we have 8 example candidate cells (given $\min = 0.1$): $c_6, c_7 \ldots c_{14}$. Fig. 7(a) highlights such cells by grey color, and Fig. 7(b) lists all cells with the computed importance and normalized value. Next, for each of the candidate cells $c$, we find its adjacent cells. After that, we connect each candidate cell to its adjacent cells by edges and compute the associated weights. Fig. 7(d) gives the road graph consisting of candidate cells and weighted edges.

![Road Inference From Road Vector](image)

Equation (3.4) indicates that two adjacent cells $c$ and $c'$, if having large $\Phi_{c,k}$ and $\Phi_{c',k}$, are with a small edge weight $w_{c,c'}$. It is not hard to understand the behind rationale why we use shortest paths on road graph $M_k$ to infer road segments: shortest paths indicate the least edge weights $w_{c,c'}$. It means that the cells $c$ and $c'$ inside the shortest paths are with high importance $\Phi_{c,k}$ and $\Phi_{c',k}$. Thus, the majority of trajectories are moving through such cells $c$ and $c'$. Thus the cells $c$ and $c'$ have high possibility to be on real road segments, and the shortest path between $c$ and $c'$ has the high chance to become real road segments.

**C. Inference of Map Skeleton $M^{-}$:** After a road graph $M_k$ is constructed, we adopt shortest paths on $M_k$ to infer road segments in $M^{-}$. For example, given the two candidate cells $c_8$ and $c_{11}$ in Fig. 7(d), we compute the shortest path between them: $c_8 - c_9 - c_{10} - c_{11}$. The shortest distance path is then as a road segment in $M^{-}$. From Fig. 7(b), we find that the cells $c_8, c_9, c_{10}, c_{11}$ in the path are with high importance, consistent with the rationale that we have mentioned previously.
Until now, a followup question is which cells inside road graph \( M_k \) should be selected to perform the computation of shortest paths. Note that a cell \( c' \) is possible to redundantly become the neighbors of multiple other cells \( c \). To this end, we choose a small amount of representative cells among the above candidate cells to perform shortest paths.

**Definition 4.** [Representative cells] Given a road graph \( M_k \) of road \( \tau_k \), a candidate cell \( c \) becomes the representative one, if no representative cell \( c' \in N(c, h) \) satisfies \( \Phi_{c,k} < \Phi_{c',k} \).

In Fig. 7(d) with \( h = 1 \), the cell \( c_8 \) is with the largest \( \Phi_{c_8,k} = 1.0 \) and becomes the first representative cell; for cell \( c_{11} \) with the 2nd largest \( \Phi_{c_{11},k} = 1.0 \), it also becomes a representative cell due to \( c_{11} \notin N(c_8, h = 1) \); any other cell \( c' \) cannot be a representative cell because either \( c' \in N(c_8, h = 1) \) or \( c' \notin N(c_{11}, h = 1) \) holds. We finally have two representative cells: \( c_8, c_{11} \).

Consider that we have \( K \) roads \( \tau_k \) and road graphs \( M_k \), and \( 1 \leq k \leq K \). Following the above Definition 4, we have a set of representative cells for each road graph \( M_k \). It is possible for a specific cell \( c \) to redundantly become a representative cell for multiple road graphs \( M_k \). Thus, in Alg. 1, we will slightly extend Definition 4 to define global and local representative cells.

**Algorithm 1:** Inference of Map Skeleton \( M^- \)

| Input: Matrices: \( \Phi, \Theta \); thresholds: \( \text{min}, h \); road count \( K \) |
| Output: Map Skeleton \( M^- \) |
| 1 | initiate two global sets: representative cells \( S \) and non-representative cells \( D \); |
| 2 | compute road ranks \( \rho_k \) and sort \( \tau_k \) by \( \rho_0 \geq \cdots \geq \rho_{K-1} \); |
| 3 | for \( 0 \leq k \leq K - 1 \) do |
| 4 | initiate two local sets: cells \( S_k \) and road graph \( M_k \); |
| 5 | select a vector \( \Phi_k \) from matrix \( \Phi \); normalize & sort \( \Phi_k \); |
| 6 | for each sorted cell \( c \in \Phi_k \) and \( \Phi_{c,k} > \text{min} \) do |
| 7 | add \( c \) to \( M_k \); |
| 8 | for each \( c' \in N(c, h = 1) \) and \( \Phi_{c',k} > \text{min} \) do |
| 9 | if \( \{c - c'\} \in M^- \) then |
| 10 | compute weight \( w_{c,c'} \) by Equation (3.4) ; |
| 11 | add edge \( \{c - c'\} \) to \( M_k \); weight \( w_{c,c'} \) by Equation (3.4) ; |
| 12 | if \( c \notin (S \cup D) \) then |
| 13 | add \( c \) to \( S \) and \( S_k \); add \( c' \in N(c, h) \) to \( D \); |
| 14 | for each \( c \in S_k \) do |
| 15 | for each \( c' \in N(c, 2 \times h + 1) \) and \( c' \in S \) do |
| 16 | in local graph \( M_k \), find shortest path \( sp \) between \( c \) and \( c' \); |
| 17 | add every edge and point in \( sp \) to \( M^- \) and \( S \); |
| 18 | remove those points \( c \in sp \) from \( D \); |

In Alg. 1, we process the roads \( \tau_k \) by descending order of road ranks \( \rho_0 \leq \cdots \leq \rho_{K-1} \) (line 2). For each road \( \tau_k \), in lines 4-13, we construct a local road graph \( M_k \) and select local representative cells \( S_k \); in lines 14-18, we perform shortest paths between two representative cells \( c \) and \( c' \) if \( 0 < dist(c, c') < 2h+1 \). Meanwhile, given totally \( K \) roads, we have to maintain global representative cells \( S \) (line 1), such that we do not select those cells \( c \) which redundantly appear in multiple roads as representative cells (see line 12). For each road \( \tau_k \), by the sorted elements \( \Phi_{ck} \) in vector \( \Phi_k \), we can safely select representative cells and add them to set \( S_k \) (lines 5-13). The members in \( S_k \) thus indicate the local representative cells. Next, given an edge \( \{c - c'\} \), the edge weight \( w_{c,c'} \) depends upon whether or not the edge has previously added to the map skeleton \( M^- \) (lines 9-11). If \( true \), we simply set the weight to be \( 0.0 \), and otherwise weight \( w_{c,c'} \) by Equation (3.4).

**D. Parameter Tuning:** The quality of \( M^- \) depends on the parameters of \( K, cw \) and \( h \). The number \( K \) depends on the total number of roads in the map. However, without the knowledge of real road network map, it is hard to determine the total number of roads from trajectory data. To this end, we alternatively compute the cell coverage ratio \( \sigma = \frac{|S|+|D|}{|G|} \), where the parameters \( S \) and \( D \) are defined in line 1 of Alg. 1, and \( C \) indicates all divided cells (see Definition 2). This parameter \( \sigma \) measures the percentage of total cells passed by trajectories that have been selected by the \( K \) roads in Alg. 1. It determines whether or not the \( K \) selected roads are enough to infer all the road segments in the dataset. Our experiment indicates that \( 0.85 \leq \sigma \leq 0.95 \) is helpful to achieve a good quality of the inferred map \( M^- \). Otherwise, a smaller \( \sigma \) means a lower cell coverage ratio, and we have to tune a larger \( K \); and a very high \( \sigma > 0.95 \), say 1.0, indicates that all trajectories including those noisy GPS points are selected, leading to a low map quality, too.

The parameter \( cw \) also affects the quality of \( M^- \). Because we use the center point of a cell to represent points in \( M^- \), a smaller \( cw \) leads to a higher quality of \( M^- \) and meanwhile much longer running time. Our experiment will verify that a road width, e.g., \( cw = 30 \) meters, comfortably achieves a good quality of \( M^- \).

In addition, the parameters \( h \) and \( cw \) work together to filter out nearby noisy data. A larger \( h \cdot cw \) could filter out more noisy data, and meanwhile combines more nearby parallel roads into a single road. Thus, we maintain \( h \cdot cw \approx 50 \) in our experiment and achieve a good quality of \( M^- \).
E. Final Refinement: As the final step, the refinement involves two operations on map $M^{-}$ to generate the final map $M$: merge and replacement. For example, in Fig. 8(b), we merge the roads $r_1$, $r_2$ and $r_3$ because the three transit roads share the same direction. Next, Fig. 8(c) gives an example of replacement. Here, the replacement requires that (i) the length of roads (e.g., $r_3$) to be replaced is smaller than $\sqrt{2} \times cw$ and (ii) the two cells $c_4$ and $c_5$ are with degree 2. Once the two requirements are met, we first replace the road $r_3$ by a new cell $c_7$, next connect the old cells $c_3$ and $c_6$ to the new one $c_7$ by two new roads $r_8$ and $r_9$, respectively, and finally remove the three roads $r_4$, $r_5$, $r_6$. We choose the center point between $c_4$ and $c_5$ as the location of $c_7$.

IV. Evaluation

<table>
<thead>
<tr>
<th>Data Set</th>
<th>Area</th>
<th>Number of Trajectories</th>
<th>Sampling Rates (s)</th>
<th>Average GPS Points</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chicago</td>
<td>14.5 km$^2$</td>
<td>889</td>
<td>3.6</td>
<td>133.1</td>
</tr>
<tr>
<td>Shanghai Small</td>
<td>25 km$^2$</td>
<td>10000</td>
<td>10.4</td>
<td>33.1</td>
</tr>
<tr>
<td>Shanghai Large</td>
<td>625 km$^2$</td>
<td>100000</td>
<td>10.3</td>
<td>96.7</td>
</tr>
</tbody>
</table>

TABLE III
Statistics of three used datasets

A. Data sets: As shown in Table III, we use three datasets in two cities:

Chicago campus shuttle bus dataset [1]: Fig. 1 plots the associated trajectories. These shuttle buses serve several regular routes around the campus. For illustration, in Fig. 1, we plot some frequently traveled roads by a red dashed square, and some rarely traveled (only by one or two trips) by a red dashed oval. It is not hard to find that the datasets contain the problem of data disparity and GPS error.

Two Shanghai taxi datasets: We use the Shanghai small and large datasets mainly for the scalability comparison. These two datasets are divided from the same Shanghai taxi trajectory dataset collected from April 1st to 5th 2015. Due to space limit, we do not plot the Shanghai dataset as Chicago campus data set does.

B. Evaluation Metrics: First, we map the trajectories of three data sets to the recent version of road network from OpenStreetMap [2] and choose those road segments that are traversed by at least one trajectory. Such road segments are then regarded as the ground truth. Next, we follow the method in the previous work [15] to evaluate different approaches. According to the evaluation method, the full set of segments of each map are first sampled at 5-meter intervals. Then, a bipartite matching is computed between the two sets. If a sample road can be matched to the other map up to a maximum match distance threshold (20 meter in both [15] and our experiment), this sample road is matched rightly. Remaining samples are spurious or missing.

We measure the quality of a map inference algorithm by three metrics: precision, recall and F-score. Precision represents the percentage of accurately inferred roads against all inferred roads, recall stands for the percentage of accurately inferred roads against the ground truth map, and F-score is a comprehensive index of both precision and recall. Finally, to study the efficiency of a map inference algorithm, we measure the running time used to infer a map when given a data set.

C. Counterparts: We implement three state-of-art algorithms (the detail refers to Section V): (1) KDE algorithm by Biagioni etc. [7], (2) an incremental track insertion algorithm by Ahmed etc. [4], and (3) cluster-based algorithm by Karagiorgou[14].

Our evaluation result is consistent with a recent survey [3]: Biagioni etc. [7] and Karagiorgou[14] can achieve better quality and accuracy than Ahmed etc. [4] but at cost of inefficiency issue. Thus, due to the inefficiency issue of [14] and [7], we only evaluate [14] on Chicago data set and [7] on two small data sets (Chicago and Shanghai small data set) only.

We measure the performance of our algorithms, a cell-based road network map inference framework (in short CRIF) and fairly compare our work with the three representative algorithms. We implement our algorithm by Python and measure all four approaches on a Debian Linux 8.3 machine with 24 2.5GHZ CPUs, 120GB RAM and 1 TB hard disk.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Algorithm</th>
<th>Precision</th>
<th>Recall</th>
<th>F-score</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chicago</td>
<td>Amhed[4]</td>
<td>0.81</td>
<td>0.66</td>
<td>0.73</td>
<td>7.2 min</td>
</tr>
<tr>
<td></td>
<td>Biagioni[7]</td>
<td>0.97</td>
<td>0.73</td>
<td>0.83</td>
<td>32.7 min</td>
</tr>
<tr>
<td></td>
<td>Karagiorgou[14]</td>
<td>0.92</td>
<td>0.76</td>
<td>0.83</td>
<td>22.4 hours</td>
</tr>
<tr>
<td></td>
<td>CRIF</td>
<td>0.94</td>
<td>0.85</td>
<td>0.89</td>
<td>4.8 min</td>
</tr>
<tr>
<td>Shanghai Small</td>
<td>Amhed[4]</td>
<td>0.64</td>
<td>0.72</td>
<td>0.68</td>
<td>31.9 min</td>
</tr>
<tr>
<td></td>
<td>Biagioni[7]</td>
<td>0.91</td>
<td>0.85</td>
<td>0.88</td>
<td>339.3 min</td>
</tr>
<tr>
<td></td>
<td>CRIF</td>
<td>0.92</td>
<td>0.89</td>
<td>0.91</td>
<td>16.2 min</td>
</tr>
<tr>
<td>Shanghai Large</td>
<td>Amhed[4]</td>
<td>0.60</td>
<td>0.73</td>
<td>0.65</td>
<td>2.8 weeks</td>
</tr>
<tr>
<td></td>
<td>CRIF</td>
<td>0.89</td>
<td>0.84</td>
<td>0.86</td>
<td>3.9 hours</td>
</tr>
</tbody>
</table>

TABLE IV
Main results on three datasets

D. Baseline Experiment: Table IV first gives the baseline results of four algorithms. In this experiment, we choose pLSA as the topic model, and the running time of CRIF includes three parts: pLSA, Alg. 1 and final refinement. In terms of cell width $cw$, hop threshold $h$, and number $K$ of roads, we set $cw = 15$, $h = 3$ and $K = 50$ for the Chicago and Shanghai small datasets (resp. $cw = 20$, $h = 2$ and $K = 400$ for Shanghai large dataset).

In general, among all four algorithms, CRIF almost achieves the best quality and efficiency on three data sets, only except the precision on Chicago dataset. In particular, CRIF can achieve much shorter running time than all other three algorithms. For example, as shown in Chicago dataset, CRIF is $1.5 \times$, $6.8 \times$ and $280 \times$ shorter than the
three counterpart algorithms. In addition, when we use the Shanghai large data set, Amhed [4] even used around 2.8 weeks to process the 100000 trajectories in the data set and instead CRIF uses only 3.9 hours.

High efficiency of CRIF is mainly because CRIF performs map inference using divided cells. The number of cells is significantly smaller than the number of GPS points. Instead, Biagioni [7] needs to generate several versions of skeleton maps and two times of map-matching to recognize roads visited by different number of vehicles. Karagiorgou [14] performs complex spatial computation to merge trajectories. Finally, Amhed’s algorithm [4] is much more efficient than the two algorithms [7], [14], but failed to achieve high quality.

As a summary, Table IV validates that CRIF can achieve much better efficiency than three previous algorithms [4], [7], [14], but without compromising the quality of inferred maps.

E. Sensitivity Study: Scalability: In order to evaluate the scalability of four algorithms, we vary the number of used trajectories in two Shanghai data sets. As shown in Figure 10, when the number of trajectories grows, CRIF can lead to better quality and meanwhile shorter running time than the other two algorithms.

Topic Model: To study CRIF’s sensitivity on the used topic model, we use both LDA and pLSA and vary the number $K$ of roads on Shanghai small dataset. In Fig. 11, a larger number $K$ leads to lower precision but higher recall rises on both LDA and pLSA. This is because with a larger $K$, more roads will be inferred into the final map $M$, leading to more false positives. When the number $K$ is around 50, F-score reaches the peak value. In addition, CRIF using pLSA can much shorter running time than CRIF using LDA, meanwhile with slightly better map quality. In Shanghai large dataset, we use a much larger $K = 400$ because there are much more roads in the larger dataset. As mentioned in Section III-C, by using $\sigma = \frac{|C|}{|S| + |D|}$, we can judge whether the selected $K$ is enough to infer all the road segments in the dataset.

Cell Width: As shown in Fig. 12, map quality goes down as cell width $cw$ grows. The result is consistent on both LDA and pLSA. This is because CRIF uses the central point of a cell to represent a point on roads. Thus, a larger $cw$ leads to a coarser granularity of CRIF to infer maps. Nevertheless, when $cw$ grows, the total number of cells is decreased, leading to a smaller size of cell-Trajectory Matrix $X$. Thus, the running time becomes shorter. For every $cw$, we select the corresponding parameter $h$ by setting $h \cdot cw \approx 50$ (see Section III-C for more detail).

F. Correction of Existing Maps: Besides the inference of a new road network map, our algorithm can be used to detect the changes in existing maps. It is particularly useful to correct an old map which is not timely updated to show the changes. To this end, we compare the generated road network produced by using our large Shanghai data set with a 2013 version road network from OpenStreetMap. Figure 13 shows an example of a suburb area in Shanghai’s Jia-Ding district. As shown in the figure, we find that the OpenStreetMap road network missed the S6 Highway (which was built in late 2014). Instead, the Shanghai data
set is generated by taxis in 2015. The inferred map by the data set clearly indicates the road S6 Highway (the highway from west to east labeled by the text \( S_6 \) inside three green color rectangle boxes). This example validates that this road network inference algorithm is practically useful to update old road networks.

V. Related Work

Literature has studied the road network map inference problem for years [3], [6]. In general, the previous work to solve the problem can be classified into three categories: clustering-based, KDE-based and incremental track insertion approaches.

**Clustering based approaches** cluster GPS points in trajectories in order to generate intersections, and next compute the edges to connect such intersections. K-means clustering has been widely used as the first step in these approaches. In these clustering-based approaches, the distance measurement involves both the geometric distance of GPS points to cluster centers as well as GPS bearing. Representative algorithms in this category include Karagiorgou and Pfoser [14], Edelkamp and Schrödl [12], Schrödl et al. [18], Worrall and Nebot [22]. Liu et al. in a recent work [15] proposed to cluster line segments based on proximity and direction, and next apply the resulting clusters to fit polylines. Chen et al. [10] recently proposed a framework involving a novel graph-based clustering techniques and a supervised junction recognition algorithm which achieves high efficiency and accuracy. However, the high accuracy relies on the supervised learning step which requires an existing map and manual labeled data involving all kinds of junctions. Instead, these additional data is not required in our algorithm.

**KDE-based approaches** employ KDE (kernel density estimation) method to transform the GPS trajectories into a discretized image representing the density of samples at each pixel. A representative work was done by Biagioni and Eriksson [7] who proposed an algorithm using the KDE approach with various thresholds to compute successive versions of a skeleton map. Then, they performed a map matching technique [17] to associate GPS trajectories with skeleton map and finally refined the skeleton map. Other algorithms in this class include the work by Davies et al. [11], Steiner and Leonhardt [19]. This kind of approach can produce a map with high quality and accuracy, but at cost
of high running time on map refinement.

**Incremental track insertion algorithms** construct a street map by incrementally inserting GPS trajectories into an empty map. Cao and Krumm [9] proposed to group nearby GPS traces together by simulating physical attraction between them. In this way, they can clarify the GPS traces to minimize the effect of the GPS errors. Ahmed and Wenk [4] presented an incremental method that employs a partial map-matching based on a variant of the Fréchet distance to identify matched portions and unmatched portions. The unmatched portions of trajectories are then inserted into the partially constructed map; and the matched edges in the map are updated using a minimum-link algorithm to compute a new representative edge. This category of algorithms process every single trajectory with the current constructed map each time and thus reduce the computation complexity. However it is affected by the error of specific single trajectory largely, leading to a low quality and accuracy of final road network maps.

Finally, Besides extracting topic from documents, topic model is also used to discover hidden thematic structure of other kinds of data. Bhattacharya and Getoor [5] use topic model to solve entity resolution problem. Zhang et al. [23] combine topic model with online analytical processing (OLAP) techniques on the dimension of text data in a multidimensional text database. Based on topic model, Tang et al. [20] effectively extract a multi-topic summary from a document collection.

**VI. Conclusion**

In this paper, we propose an effective and efficient map inference solution framework. With help of the popular topic model (LDA and pLSA), our algorithm can efficiently infer road network maps with high map quality. Our experiments on three data sets have successfully validated the advantages of our algorithm over three representative algorithms in terms of both inference efficiency and map quality.

While the results of our solution framework using topic model are promising, much work remains to improve our solution. For example, instead of evenly dividing map extend to cells of equal size, we could adopt a quadtree to divide the map extent, such that the number of GPS points are evenly distributed among the quadtree’s leaf cells. In addition, we are interested in the adaptation of our solution to incrementally process incoming trajectories and to update existing maps.

**References**