Automatic Generation and Validation of Road Maps from GPS Trajectory Data Sets

Hengfeng Li, Lars Kulik, Kotagiri Ramamohanarao
Department of Computing and Information Systems
The University of Melbourne
Victoria 3010, Australia
henli@student.unimelb.edu.au, {lkulik, kotagiri}@unimelb.edu.au

ABSTRACT

With the popularity of mobile GPS devices such as on-board navigation systems and smart phones, users can contribute their GPS trajectory data for creating geo-volunteered road maps. However, the quality of these road maps cannot be guaranteed due to the lack of expertise among contributing users. Therefore, important challenges are (i) to automatically generate accurate roads from GPS traces and (ii) to validate the correctness of existing road maps. To address these challenges, we propose a novel Spatial-Linear Clustering (SLC) technique to infer road segments from GPS traces. In our algorithm, we propose the use of spatial-linear clusters to appropriately represent the linear nature of GPS points collected from the same road segment. Through inferring road segments our algorithm can detect missing roads and checking the correctness of existing road network. For our evaluation, we conduct extensive experiments that compare our method to the state-of-the-art methods on two real data sets. The experimental results show that the F_1 score of our algorithm is on average 10.7% higher than the best state-of-the-art method.

CCS Concepts

•**Information systems** → **Spatial-temporal systems; Geographic information systems;** *Data mining; Clustering;*

Keywords

GPS traces; Map inference; Spatial-linear clustering

1. INTRODUCTION

Map data is of utmost importance in many LBSs (Location Based Services) and traffic applications, such as route planning and traffic monitoring. In the future, real-time traffic advisory systems urgently need the continuously updated map data to ensure safety of vehicles. However, traditional ways of building road maps are labor-intensive, involving specialized vehicles and skilled operators. Furthermore, dynamically updating road maps in real time can be very expensive and time consuming as roads need to be visited and maps accordingly updated. Therefore, quality and timeliness are major concerns when collecting the map data. We propose a solution to gather real-time data from a fleet of vehicles, and automatically infers maps from GPS traces, which results in intervention-free up-to-date road maps.

Automatic validation of existing road maps is another important application. To get up-to-date map data, OpenStreetMap (a well-known example of community-built road maps) allows users to edit maps from GPS traces and satellite images. However, this still requires editors to check tiny details visually, and errors may arise from ignorant or inexperienced users. Automated map inference has advantages compared to crowdsourcing such as high efficiency, easy scalability and comparatively low cost.

In addition, automatic and real-time map generation can be used for detecting abnormal traffic events such as temporary closure of roads due to construction or repairs. In addition, monitoring changes of inferred maps periodically enables us to analyze the cause of continuing traffic problems. Map inference from GPS traces can be used to create many customized maps for commonly travelled paths of pedestrians, bicycle riders, or even animals, without any human interference or involvement. The creation of such customized maps through humans are labor-intense and expensive, which requires specialized devices and professionals.

Challenges

In order to successfully infer accurate road maps, we have to address the following challenges:

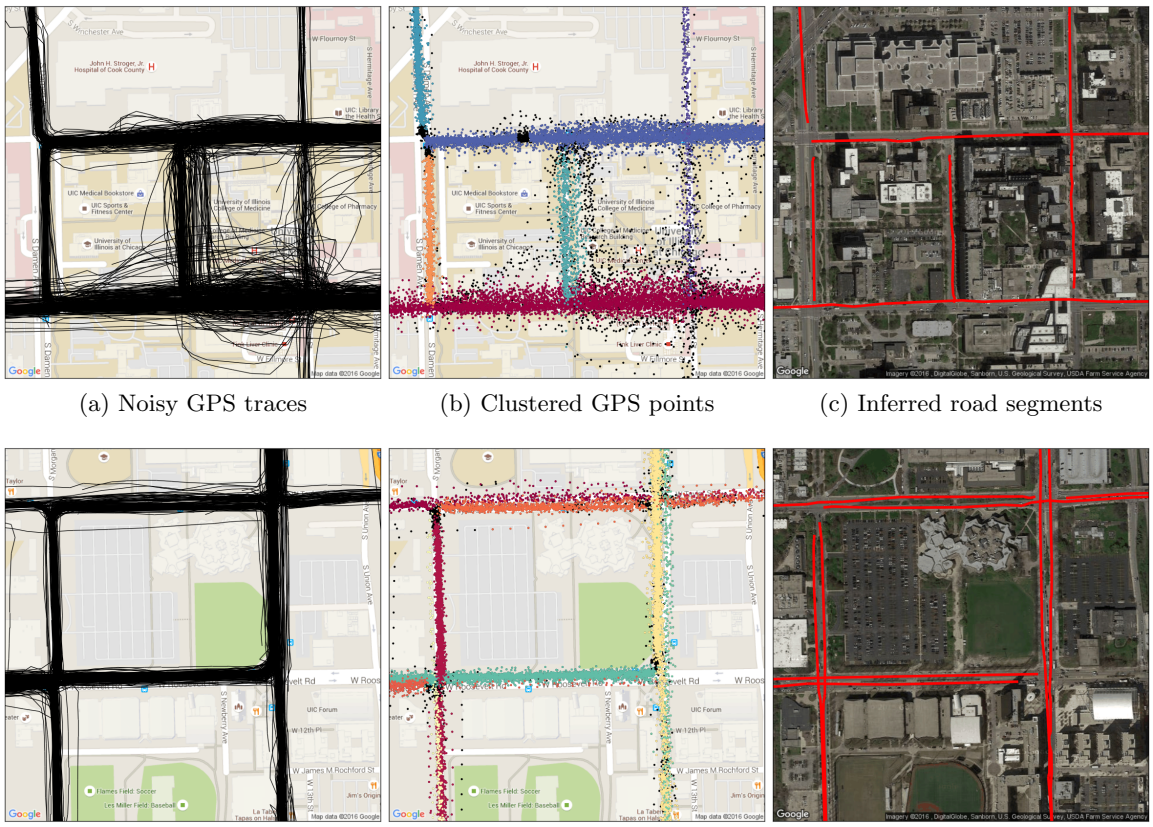
(a) *GPS noise:* GPS measurement noise is the unavoidable result of imprecision in sensed GPS location data. The noise level varies for different conditions, e.g., a satellite visibility is very poor if a GPS receiver is surrounded by several skyscrapers. Under noisy conditions, the recorded coordinates of a GPS receiver can be far from its actual locations and may not be located on the roads (in Fig. 1(a)(d)). This makes the map inference difficult in matching or finding the correct roads.

(b) *Low sampling rate:* GPS traces are sampled at a low frequency rate to save energy and computation resources for mobile devices. For a trajectory with a low sampling rate, the line segment of two consecutive points can be long and may not match the actual route between them (see Fig. 1(a)(d)). Especially when the travelled path is not a straight line, the sampled GPS line segment crosses residen-

CIKM'16 , October 24 - 28, 2016, Indianapolis, IN, USA

ISBN 978-1-4503-4073-1/16/10...$15.00

DOI: http://dx.doi.org/10.1145/2983323.2983797



(a) Noisy GPS traces (b) Clustered GPS points (c) Inferred road segments

(d) Noisy GPS traces (e) Clustered GPS points (f) Inferred road segments

Figure 1: An overview of our map inference approach for GPS traces.

tial areas rather than following the actual roads. Therefore, instead of using a single trace, we have to use knowledge from multiple nearby GPS traces to correctly generate the actually travelled roads.

Contributions

To address above challenges, we propose a clustering-based algorithm for map generation, called Spatial-Linear Clustering (SLC), to improve the inference accuracy. We group a set of GPS points that are spatial-linearly distributed because of the linear nature of a road network. In the algorithm, we use spatial-linear clusters to approximate the geometry of roads. A spatial-linear cluster starts with a randomly chosen point, and then the cluster searches similar points that might belong to the same road. The algorithm considers the range of GPS measurement error and the moving direction between two consecutive points when searching for similar points. The algorithm generates new clusters incrementally along the two opposite directions of the initial cluster. After combing all generated clusters, we obtain a long and thin cluster, called a spatial-linear cluster (see Fig. 1(b)(e)). We will show that this is a more effective approach to generate a road segment under highly noisy conditions, because our algorithm focuses on building clusters for road segments that are learned from the entire set of GPS points.

After forming the spatial-linear clusters, we propose a geometric approach to estimate road segments for each cluster (see Fig. 1(c)(f)). We use the knowledge of trajectories that are completely or partially covered by a cluster. Those sub-trajectories are used for generating road segments for each cluster. With sufficient support of GPS samples, our algorithm can achieve high inference accuracy for road generation. In addition, our algorithm can identify unidirectional roads, such as the parallel roads in Fig. 1(f), because we consider the moving direction of a GPS trace and create clusters for each direction of a road.

In summary, our contributions in this paper are as follows:

- We propose an effective clustering-based algorithm for map inference in Section 3. We spatially cluster GPS points that might originate from the same road segment. These spatial-linear clusters can effectively indicate the existence of road segments.

- We propose a geometric approach to estimate roads for each cluster in Section 4. We consider line segments between two consecutive points in the same trajectory. These line segments represent the partial movement of a trajectory, which has a high likelihood of an actual trip via a road segment.

- We demonstrate the effectiveness of our algorithm on two real data sets in Section 5. We also compare our algorithm with two state-of-the-art methods. Our algorithm can improve on average the F_1 score by 10.7% compared to the best state-of-the-art method.

2. PROBLEM STATEMENT

In this section, we first present preliminary concepts used for describing and defining the map inference problem accurately. On this basis we give a problem definition of map inference. Finally, we provide a system overview of our proposed approach.

2.1 Preliminary Concepts

Definition 1. (ROAD MAP). $RM = (V, E)$ is a graph representing the road network. The vertices V represent joint or end points in a road network and the edges E correspond to directed road segments.

Thus, a vertex $v \in V$ can only be a start or an end point of an edge $e \in E$. Compared to the graph structure in computer science, an important difference is that a vertex v in a road map has its own geographic location represented by the spherical coordinates.

Definition 2. (GPS TRAJECTORY). The raw trajectory T consists of measured GPS points along the traveling path during a fixed or varied time intervals. A trajectory function f_T can be defined as:

$$\text{for } t \in [t_1, t_2, ..., t_n]: \qquad t \mapsto f_T(t) = (\lambda, \delta)$$

Each GPS point is represented by the spherical coordinates with its measured longitude λ ($0 \leq \lambda < 2\pi$) and latitude δ ($-\pi \leq \delta \leq \pi$) at a certain timestamp t. Due to GPS measurement noise, the location of each point is uncertain.

Definition 3. (MAP INFERENCE). Map inference computes a road map RM through mining road segments from a set of GPS traces, $TS = \{T_1, T_2, ..., T_m\}$ and m is the number of traces. The process can be defined as a function $Infer$ to infer maps from TS to RM:

$$Infer : TS \mapsto RM$$

Definition 4. (MAP SIMILARITY MEASURE). A map similarity measure $Sim(RM_1, RM_2)$ calculates the similarity of two road maps, RM_1 and RM_2. A higher value between two road maps refers to a higher similarity.

$$\forall RM_1, RM_2: \quad RM_1, RM_2 \mapsto Sim(RM_1, RM_2) \in [0, 1]$$

2.2 Problem Definition

In this paper, we compute a road map RM from a set of GPS traces TS. The computed road map RM should be as similar as possible to the actual road network RM_{truth}. GPS measurement errors cause imprecision in GPS location data. Low sampling intervals increase the uncertainty of the actual travelled path between two consecutive points. Thus, a GPS trace does not perfectly lie on a travelled path on the actual road map. The map inference problem can be defined as how to transform a set of raw GPS traces with noise and spareness issues to a geographical graph structure (a road map).

Next, we need to define a measure to compute the similarity between the inferred road map and the actual map. This similarity measure determines which algorithm has a better map inference accuracy. We assume that RM_{A_1} and RM_{A_2} are inferred maps from two different map inference algorithms A_1 and A_2, respectively. The similarities to the actual map are calculated, i.e., $Sim(RM_{A_1}, RM_{truth})$ and $Sim(RM_{A_2}, RM_{truth})$. If $Sim(RM_{A_1}, RM_{truth})$ is greater than $Sim(RM_{A_2}, RM_{truth})$, then A_1 is better than A_2 in terms of map inference accuracy.

Overall, we want to validate the effectiveness of our proposed algorithm to address the map inference problem and also show its superiority to the state-of-the-art methods in terms of inference accuracy.

3. SPATIAL-LINEAR CLUSTERING

Most existing clustering-based methods address the map inference problem through the k-means algorithm [11, 12, 25]. They randomly generate clusters for the set of GPS points, build cluster centers and identify road segments by connecting these centers. However, we show that this way of finding cluster centers is not suitable for map inference, because simple line connections between cluster centers are not precise for representing road segments. We propose spatial-linear clusters, which are more suitable to match a set of GPS points recorded from a road segment or an entire street. In addition, our clusters are linearly growing to include as many points as possible. Therefore, we have more support for estimating accurate representation of road segments. This significantly improves the performance of map inference.

Now, we first give definitions of *anchor point*, *directly reachable*, *indirectly reachable*, and *spatial-linear cluster*.

Definition 5. (ANCHOR POINT). A point p is an anchor point if $|N_{\alpha,\beta}(p)| \geq minpts$, where $N_{\alpha,\beta}(p)$ computes the set of all the similar points to p and $minpts$ refers to the minimum number of points. p_1 and p_2 are considered similar if: angle_diff($p1$, $p2$) $\leq \alpha$ and dist($p1$, $p2$) $\leq \beta$, where angle_diff() computes the difference of moving direction and dist() calculates the proximity of two points.

The moving direction of each point is represented by computing the angle from the current point to its next point in the same trajectory. For example, p_i and p_{i+1} are two consecutive points in the same trajectory T. The moving direction at p_i is calculated by the angle of the line segment $p_i p_{i+1}$.

Definition 6. (DIRECTLY REACHABLE). Two anchor points, $<p_i, p_j>$, are *directly reachable* if $p_i \in N_{\alpha,\beta}(p_j)$ and $p_j \in N_{\alpha,\beta}(p_i)$.

Definition 7. (INDIRECTLY REACHABLE). Two anchor points, $<p_i, p_j>$, are *indirectly reachable* if a path $p_i p_0 p_1 ... p_{n-1} p_j$ exists where $<p_i, p_0>$ and $<p_{n-1}, p_j>$ are *directly reachable*, and $\forall k \in [0, 1, ..., n-2]$ $<p_k, p_{k+1}>$ are *directly reachable*.

Definition 8. (SPATIAL-LINEAR CLUSTER). A spatial-linear cluster C is determined by a set of anchor points P_{anchor} that lie on a line, $C = \sum_{p \in P_{anchor}} N_{\alpha,\beta}(p)$, where $<p_i, p_j>$ are either *directly reachable* or *indirectly reachable* $\forall p_i, p_j \in P_{anchor}$.

The goal of our algorithm is to cluster GPS points spatial-linearly for inferring road segments. The clustering algorithm has no knowledge of the road network. The algorithm generate spatial-linear clusters as follows: (a) building an initial spatial-linear cluster; (b) computing the moving direction of points in this cluster; and (c) growing the cluster forward and backward along with its moving direction.

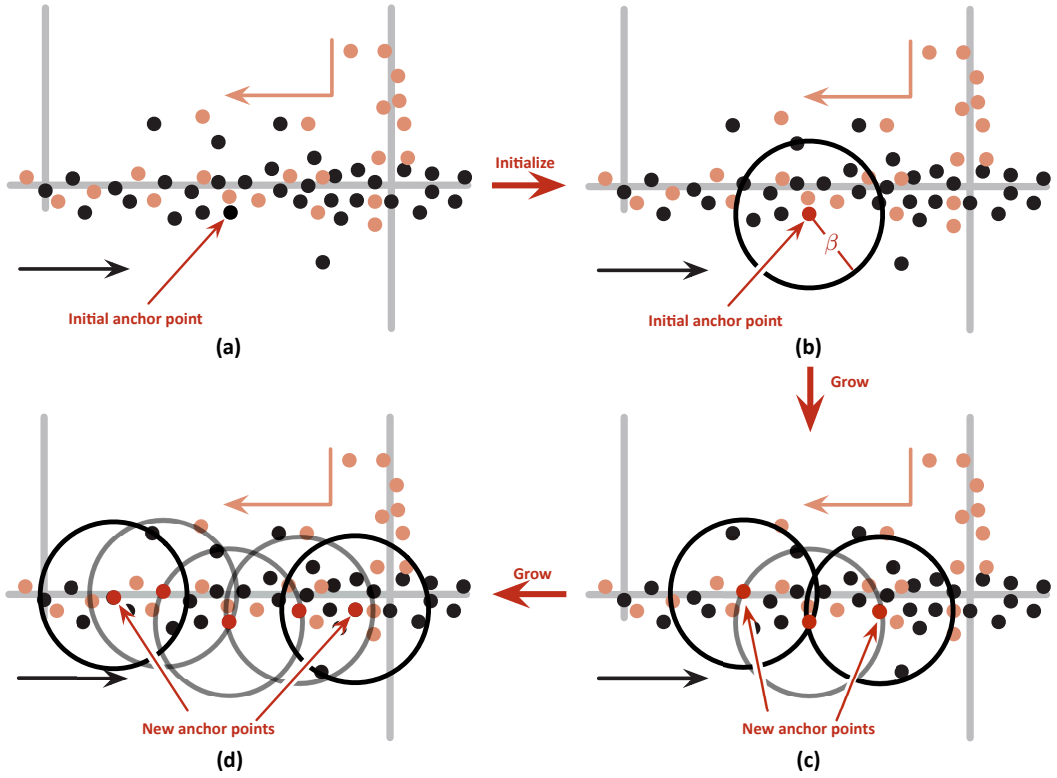


Figure 2: An illustrating example of generating and growing an initial spatial-linear cluster. (GPS points, ● and ●, move along with the underlying roads ▬ on two different directions and anchor points ● are chosen.)

We propose a heuristic search approach for building spatial-linear clusters. An illustrating example of generating and growing spatial-linear clusters is shown in Fig. 2. GPS points (● and ●) represent trajectories with different directions. They are generated by multiple traces and do not perfectly lie on the road segments (▬) due to the measurement error. The algorithm starts by choosing a random point (● in Fig. 2(a)) as the initial *anchor point* ●. The algorithm locates the nearest neighbors of that point around certain distance β and filters points that do not share similar moving direction (assume that the difference between moving directions of ● and ● is greater than α in Fig. 2(b)). Therefore, we only add points (●) into our cluster. For the growing step, the algorithm computes the average moving direction of points in current cluster. The next *anchor point* ● is chosen from neighbor points, which has the maximum projected distance to the current *anchor point* on the line of average moving direction as shown in Fig. 2(c)(d). This is a heuristic procedure, which adds as many point as it can until it meets two following cases: (i) no more new anchor points can be discovered; (ii) the growing side overlaps with existing clusters (points are already visited).

The pseudocode of our spatial-linear clustering algorithm is presented in Algorithm 1. It initializes a counter of generated clusters and creates a set of points that will be visited (Lines 1–2). We choose a random point p_{rand} as the initial anchor point and compute its similar neighbors N_{rand} (Lines 3–4). If the size of similar neighbors $|N_{rand}|$ is greater than the minimum number of points *minpts*, we consider them as an initial cluster and grow it based on its average moving

Algorithm 1 Spatial-Linear Clustering Algorithm

Input: (1) A set of GPS traces $TS = \{T_1, T_2, ..., T_m\}$, (2) Three parameters α, β, and *minpts*.
Output: A set of clusters $CS = \{C_1, C_2, ..., C_k\}$

1: $clusterId \leftarrow 1$;
2: $PS \leftarrow \{p \mid \forall p \in T_i \text{ and } \forall T_i \in TS\}$;
3: **while** $PS \neq \varnothing$ **do**
4: $p_{rand} \leftarrow$ Pop out a point from PS randomly;
5: $N_{rand} \leftarrow$ Compute $N_{\alpha,\beta}(p_{rand})$;
6: **if** $|N_{rand}| \geq minpts$ **then**
7: Mark $\forall p \in N_{rand}$ as $clusterId$;
8: Remove N_{rand} from PS;
9: $a_{avg} \leftarrow (\sum_{p \in N_{rand}} angle(p))/|N_{rand}|$;
10: GrowCluster($p_{rand}, N_{rand}, a_{avg}, clusterId, true$);
11: GrowCluster($p_{rand}, N_{rand}, a_{avg}, clusterId, false$);
12: $clusterId \leftarrow clusterId + 1$;
13: **else**
14: Mark p_{rand} as a noisy point;
15: $CS \leftarrow$ Assign $\forall p \in T_i$ and $\forall T_i \in TS$ to its $clusterId$;
16: **return** CS

17: **function** GrowCluster(p, N_p, a_{avg}, $clusterId$, $flag$)
18: $p_{anchor} \leftarrow$ Choose next anchor point from N_p, $flag$ determines either identical or opposite to a_{avg};
19: $N_{anchor} \leftarrow$ Compute $N_{\alpha,\beta}(p_{anchor})$;
20: **if** $p_{anchor} \neq \varnothing$ and $|N_{anchor}| \geq minpts$ **then**
21: Mark $\forall p \in N_{anchor}$ as $clusterId$;
22: Remove N_{anchor} from PS;
23: GrowCluster($p_{anchor}, N_{anchor}, a, clusterId, flag$);

direction a_{avg}; otherwise, we consider this point p_{rand} as a noisy point (Lines 6–14). We assign each point to a cluster according to its $clusterId$ and return the set of clusters (Lines 15–16). The cluster grows along its average direction a_{avg} if we can find next an anchor point, and $flag$ refers to whether the growing direction should be either identical or opposite to a_{avg} (Lines 17–23).

We analyze the time complexity of our SLC algorithm.

LEMMA 1. *The spatial-linear clustering in Algorithm 1 can be performed in $O(n \log n)$ time with using a spatial index, where n is the number of points.*

PROOF. The key operation, computing similar neighbors $N_{\alpha,\beta}$, determines the time complexity of the algorithm. Making rectangular range queries on GPS points is a frequent operation in our algorithm for growing clusters. If a spatial index is used such as R*-tree [5], computing $N_{\alpha,\beta}$ can be done in $O(\log n + k)$, where k is the number of reported points. Assume that the function GrowCluster is performed $2c$ times for building each cluster (Lines 10–11). The time of creating a cluster is $2c(\log n + k)$. It means that each cluster has on average $2ck$ points. So the algorithm can be performed in time $O(n/(2ck) \times 2c(\log n + k)) = O(n/k \times (\log n + k))$. The query region is expected to be far smaller than the entire area so that $k \ll n$ and $k \in O(1)$. Thus, the average total time is $O(n \log n)$. □

4. GENERATION OF ROAD SEGMENTS

After creating the clusters, we generate road segments for each cluster. We incorporate the trajectory information by considering line segments that are either completely or partially covered by each cluster. We propose a geometric approach for map inference based on the idea of producing a representative trajectory from a set of line segments in [16].

The goal of our algorithm is to generate a minimum trajectory from a set of similar traces in the same cluster. The minimum trajectory is our estimation of existence of road segments. An illustrating example of our algorithm is shown in Fig. 3. There are three GPS traces (●), T_1, T_2 and T_3. In the algorithm, we compute the average moving direction a_{avg} and transform the coordinate system by rotating a_{avg} to ensure that the x-axis is parallel to a_{avg}. Assume that the threshold $minsegs = 2$, which means that the dashed lines, from x_1 to x_6, need to intersect at least two segments in order to generate an average point (●) for the minimum trajectory. The example in Fig. 3 illustrates that the algorithm has generated a partial minimum trajectory from x_2 to x_6. There is no average point at x_1, because line x_1 only intersects with the trace T_2 and the number of intersecting segments is less than $minsegs$. The new average point p_{avg} is computed at line x_6, because the number of its intersecting segments is greater than $minsegs$. The y-value of p_{avg} is computed from the average y-values of p_1, p_2 and p_3, where p_1 (●) is a reported GPS point and p_2 and p_3 (●) are intersections.

The pseudocode of our road segment generation is presented in Algorithm 2. The algorithm starts with computing the average direction a_{avg} of all line segments (Line 1–2). Then, it transforms LS to LS' by rotating the axes and the transformed x-axis is parallel to a_{avg} (Line 3–4). The algorithm sorts all transformed points based their x-value (Line 5). For each transformed point, we get its x-value and compute intersecting segments from LS'. If the number of intersecting segments is greater than $minsegs$, we need to generate a new average point and append it to the polyline after transforming it back to the original coordinate (Line 6–12).

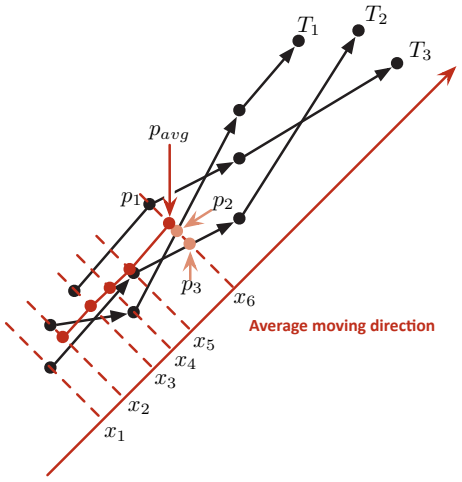


Figure 3: An illustrating example of generating road segments for a cluster. Some segments of three GPS traces (●) T_1, T_2 and T_3 are covered in the cluster. We generate a minimum trajectory (●) from these three traces. p_2 and p_3 (●) are intersections of line x_6 at T_1 and T_2, respectively. The new average point p_{avg} is computed from p_1, p_2 and p_3.

Algorithm 2 Road Segment Generation

Input: (1) A set of line segments $LS = \{S_1, S_2, ..., S_m\}$, (2) a threshold $minsegs$.
Output: A generated polyline PL (a sequence of points).
1: // Compute the average direction
2: $a_{avg} \leftarrow \sum_{S \in LS} angle(S)/|LS|$;
3: // Rotate the coordinate system by a_{avg} degree
4: Transform LS to LS' by rotating the axes and x-axis is parallel to a_{avg};
5: $PS \leftarrow \forall p \in S$ and $S \in LS'$; Sort PS on x-axis;
6: **for** $p \in PS$ **do**
7: $x \leftarrow p'$s x-value;
8: $pts \leftarrow$ Compute intersections of line x to $\forall S \in LS'$;
9: **if** $|pts| \geq minsegs$ **then**
10: $p_{avg} \leftarrow$ compute the average y-value from pts;
11: Transform p_{avg} back to the original coordinate;
12: Add p_{avg} to the polyline PL;
13: **return** PL

We now analyze the time complexity of the road generation algorithm.

LEMMA 2. *The road generation algorithm in Algorithm 2 can be run in $O(n(\log n + m))$ time with using a spatial index, where n is the number of distinct points in the set of line segments and m is the average number of intersections at each point.*

PROOF. The key operation is to compute the intersections between line x and any line segment in LS if they intersect (Line 8). We only need to check whether a line segment contains x-value. With a spatial index, this range search can be performed in $O(\log n)$ time. Another key operation

Table 1: Description of experimental data sets

Location	Chicago [6]	Porto [18]
Sampling (secs)	3.6	15
Trips	889	10,000
Number of GPS Points	118,364	296,573
Duration (hr)	∼118	∼1194
Length (km)	2,863	15,150

is to calculate the average point p_{avg} from the reported intersections pts (Line 10). Assume that the average number of intersections at each point is m. The algorithm can be performed in $O(n(\log n + m))$. □

5. EXPERIMENTS

In this section, we will evaluate the performance of our proposed algorithm. An existing measure is used for evaluating our inferred maps. We compare our algorithm with the state-of-the-art methods on two real data sets. All experiments are conducted on a 2011 MacBook Pro with 2.3 GHz Intel Core i5 and 16 GB RAM. In addition, we make our code publicly available to the community[1].

5.1 Experiment Setting

5.1.1 Data Sets

We evaluate our algorithm on two real data sets described in Table 1:

Chicago Shuttle Bus Data. This benchmark data set [6] consists of 889 shuttle bus GPS traces in Chicago for an approximate duration of 118 hours. It has been commonly used in some existing work to evaluate map inference algorithms [3, 6, 7, 17]. The data set has an average sampling interval of approximately 3.6 seconds, and its range varies from 1 second to 29 seconds. The traces contain 118,364 GPS data points that cover a region between $-87.687150°$ and $-87.639896°$ longitude and $41.861952°$ and $41.883565°$ latitude. The ground truth map is also provided.

Porto Taxi Data. 10,000 trajectories are chosen from a data set performed by 442 taxis in Porto, Portugal [18]. The trajectories contain 296,573 GPS data points that cover a region between $-8.612244°$ and $-8.588826°$ longitude and $41.13711°$ and $41.158098°$ latitude. These taxi trajectories are recorded for 1,194 hours in total, and its sampling interval is about 15 seconds. We use the map data from the OpenStreetMap[2] as ground truth.

5.1.2 Competing Algorithms

Some competing inference algorithms are implemented in the work [6, 7]. The authors also made their code publicly available. Thus, we can compare our proposed algorithm with their work. The experimental results in [3, 6, 7] show that the KDE-MM is the best state-of-the-art method. Therefore, in our experiments, we compare our algorithm with the state-of-the-art methods, KDE and KDE-MM.

KDE: The Kernel Density Estimation (KDE) method [10] cleans the GPS traces by processing the generated image. After computing the contours of cleaned traces, a Voronoi digram is used for inferring road segments.

[1]http://people.eng.unimelb.edu.au/henli/projects/map-inference/

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

KDE-MM: The hybrid approach [7] combines the KDE algorithm with map matching techniques. It first uses the KDE method to generate an initial map, and then matches GPS traces to the generated map. With map-matched traces, the approach refines the initial map based on its geometry and topology.

SLC: This represents our proposed Spatial-Linear Clustering (SLC) algorithm. Different from existing methods, we assume a set of GPS points from the same road segment can be better represented by a spatial-linear cluster.

5.2 Evaluation Methodology

In our experiments, we mainly use the F_1-measure for evaluating the map inference algorithms as proposed in [17]. F_1 score is the harmonic mean of *precision* and *recall*. Assume that we have an inferred map $M_{inferred}$ and a ground truth map M_{true}, *precision* refers to the ratio of comparing the length of matched edges $||M_{inferred} \cap M_{true}||$ to the length of inferred map $||M_{inferred}||$. *Recall* calculates the ratio of comparing the length of matched edges $||M_{inferred} \cap M_{true}||$ to the length of ground truth map $||M_{true}||$.

$$Precision = \frac{||M_{inferred} \cap M_{true}||}{||M_{inferred}||} \quad (1)$$

$$Recall = \frac{||M_{inferred} \cap M_{true}||}{||M_{true}||} \quad (2)$$

$$F_1\ score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \quad (3)$$

where $||M||$ returns the total length of edges $e \in M$.

The computation of $M_{inferred} \cap M_{true}$ is the key part of this F_1 measure. The approach is to sample every edge $e \in M_{true}$ on a distance interval. For each sampled point, if an edge $e' \in M_{inferred}$ is within a matching threshold distance and the turning angle between e and e' is less than a specified degree α, 60° given in [17], this sampled point is counted as a matching point. The length of matched points in edge e is added into $||M_{inferred} \cap M_{true}||$.

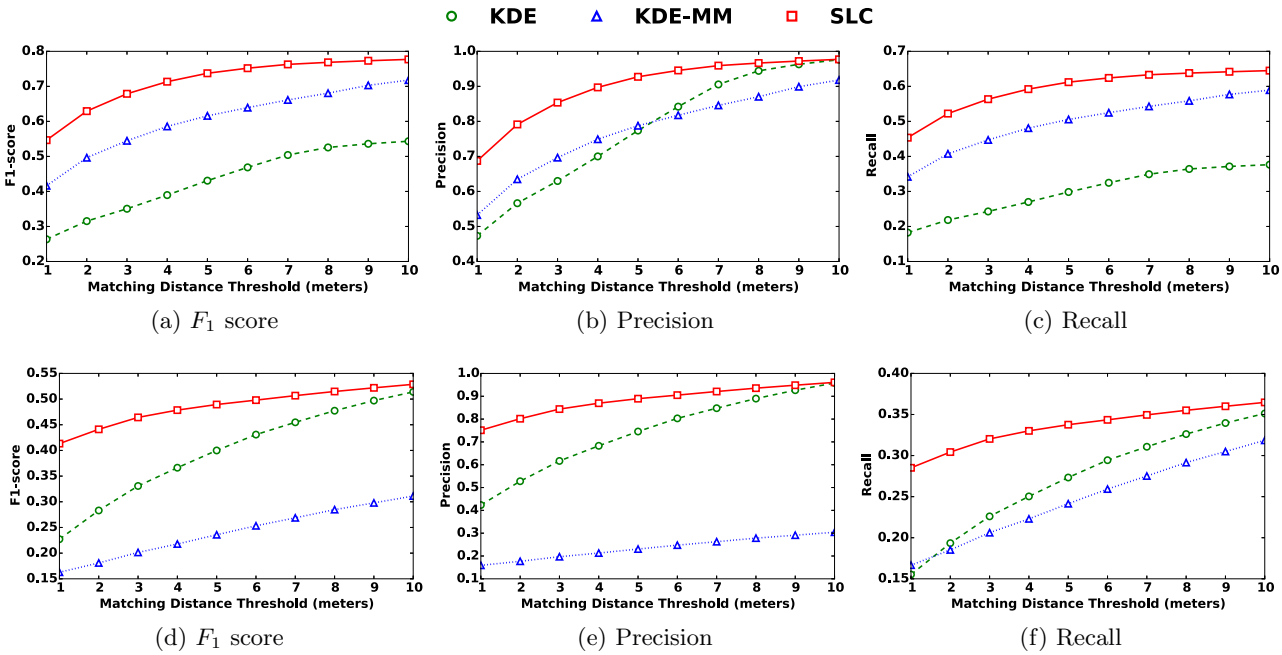
5.3 Experimental Results

5.3.1 Inference accuracy

The experiments show that our proposed algorithm significantly improves the map inference accuracy. In Fig. 4(a), our method always outperforms other methods in terms of the F_1 score, increasing from approximately 50% to over 70%. In our experiments, we focus on small matching distance from 1m to 10m, which is more important for generating highly accurate road maps. Existing works, such as [6, 7], use larger matching distance up to 30m to achieve a high F_1 score, which is not practical for generating accurate and usable road maps.

With the small matching thresholds from 1m to 5m, our method can achieve much higher F_1 score than other competing methods. Especially, Fig. 4(a) shows that within the 5-meter matching threshold, our algorithm can have F_1 scores over 10% higher than the state-of-the-art method.

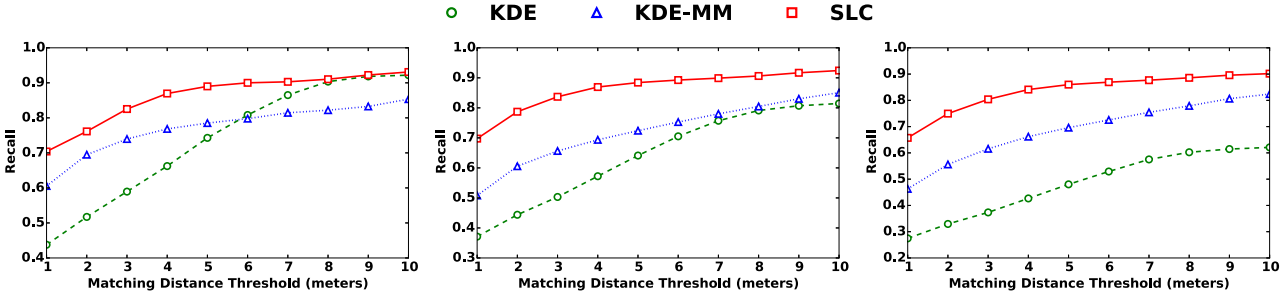
Fig. 4(b) and Fig. 4(c) show that compared to KDE and KDE-MM, our method maintains both high precision and high recall. The precision of our method is almost 15% higher than that of KDE-MM. The pure KDE method has a high precision but with the lowest recall in map inference. In other words, its inference is highly precise but only covers a small part of the ground truth map.



(a) F_1 score (b) Precision (c) Recall

(d) F_1 score (e) Precision (f) Recall

Figure 4: (a)(b)(c) shows the comparison of F_1 measure on the Chicago data set. (d)(e)(f) shows the same comparison on the Porto data set.



(a) Top 10% (b) Top 30% (c) Top 50%

Figure 5: Recalls (coverage) of frequently travelled roads with dense GPS samples on the Chicago data set.

In addition to the Chicago data set, we also run the experiments on a large data set, as shown in Fig. 4(d). When we run the experiments on Porto data set, the KDE-MM method only can give a partial result (generated by KDE) due to the errors of map matching. We suspect that KDE-MM works well on large city-scale blocks, because the road network is more complicated and map matching for all GPS trips is expensive and error-prone. Therefore, on Porto data set, we focus on the comparison between our algorithm and pure KDE method. Fig. 4(d) shows that the F_1 score of our algorithm is much higher than that of KDE method. In Fig. 4(e) and Fig. 4(f), our algorithm achieve higher precision and recall compared to KDE.

Fig. 5 shows the comparison of map inference algorithms in terms of coverage of frequently travelled roads. We associate a sampled GPS point to its closest road segment. Then, we rank all road segments in terms of the number of sampled GPS points in oder to select top 10%, top 30% and top 50% roads. Those roads have more sampled points and should be easy to discover. Therefore, it is important to check the coverage of those frequently travelled roads in our inferred maps. In Fig. 5, our algorithm outperforms the state-of-the-art methods in terms of the coverage of frequently travelled roads. The KDE method decreases its coverage when increasing the percent from top 10% to top 50%. KDE is based on the density of sampled GPS points and frequently travelled roads can be accurately inferred with enough support. However, with introducing more less frequent roads, the coverage of KDE decreases significantly, because KDE has a low recall in total. Overall, our SLC algorithm outperforms the state-of-the-art methods for inferring frequently travelled roads.

5.3.2 *Close-up comparison*

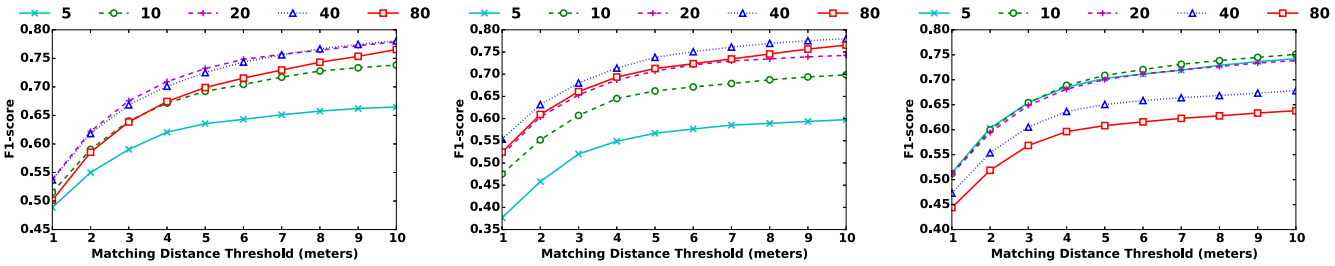
Fig. 6 shows a close-up comparison of inferred road segments. Fig. 6(a)(e)(i) show that raw trajectory data (black lines) can be extremely noisy and the underlying road segments (red lines) are difficult to discover from those trajectory data. All road segments inferred by the KDE method

(a) Raw trajectory data (b) KDE (c) KDE-MM (d) SLC

(e) Raw trajectory data (f) KDE (g) KDE-MM (h) SLC

(i) Raw trajectory data (j) KDE (k) KDE-MM (l) SLC

Figure 6: Close-up comparison of inferred road segments on the Chicago data set (red lines are actual underlying roads).



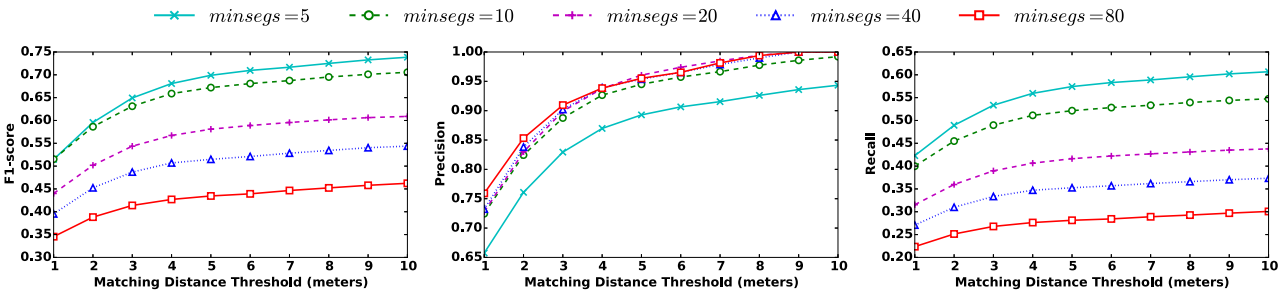
(a) α (degrees) (b) β (meters) (c) *minpts*

Figure 7: A comparison of different values of α, β and *minpts* in our SLC algorithm.

follow a zig-zag pattern. This leads to a poor quality route when navigating on the inferred map, see the red circle in Fig. 6(b). Although inferred roads of KDE-MM has better quality than that of KDE, KDE-MM causes some serious errors, such as red circles in Fig. 6(c)(g). Fig. 6(c) shows an error that KDE-MM adds a nonexistent road and Fig. 6(g) shows that an erroneous intersection is created. Our algorithm shows a high inference accuracy of road segments. In addition, our algorithm can precisely discover the unidirectional roads, see green circles in Fig. 6(h)(l), which cannot be detected from the state-of-the-art methods.

5.3.3 *Tuning parameters*

The performance of our method is affected by two processes: clustering of GPS points and generating of road segments. For the clustering, it has three parameters, the angle threshold α, the distance threshold β, and the minimum number of points *minpts*. Fig. 7 shows that the choices of these parameters affect the performance of our SLC algorithm. In Fig. 7(a), a low angle threshold limits the number of points that can be added to the cluster, because sampled GPS points are noisy and they do not perfectly lie on the road. If the angle threshold is too narrow, there is a low likelihood that two consecutive GPS points are added into the cluster. If α is too large, the precision decreases leading to the poor inference accuracy. The choice of the distance threshold β also highly affects the performance of the algorithm, see Fig. 7(b). Fig. 7(b) shows that when $\beta = 40$ meters, the algorithm has the best performance among all testing values. A small distance threshold limits the number of points that can be found in the range. A high distance threshold may introduce more noisy points leading to low



(a) F_1 score (b) Precision (c) Recall

Figure 8: A comparison of different values of *minsegs* in the generation of road segments.

performance. Fig. 7(c) demonstrates that a high *minpts* value lowers the F_1 score, because points on a sparse area are treated as noisy points leading to a low recall.

For the generation of road segment, *minsegs* indicates the minimum number of segments that we need to ensure that there is a potential road. In other words, *minsegs* refers to the minimum support that we need to generate a road segment. Fig. 8(a) shows that with decreasing *minsegs*, our algorithm achieves higher F_1 score. This is because with a low *minsegs*, the algorithm can generate more road segments leading to a high recall (in Fig. 8(c)). However, with a low support threshold *minsegs*, the geographical location of road segments are not accurate. For example, in Fig. 8(b), when $minsegs = 5$, the algorithm has the lowest precision. Therefore, there is a tradeoff between precision and recall. Our algorithm can be tuned to meet the user's preference.

6. RELATED WORK

The map inference problem was proposed by an early work in [11] about how to transform GPS data into a weighted and directed graph structure. Recently, some surveys and comparisons of existing map inference algorithms have been published in [3, 17, 6]. Existing map inference algorithms can be categorized based on the use of techniques [6]: *K-means*, *Trace Merging*, and *Kernel Density Estimation (KDE)*.

K-means. The k-means approaches first build clusters based on GPS points alone or detected intersections by taking their locations and bearings into considerations. By connecting those created clusters, the approaches consider the links between clusters as potential roads. In order to get close to the actual roads, tuning the positions of cluster centers is an important step this type of approaches. Existing algorithms in this class include work in [11, 12, 25, 13, 1, 14, 24]. The k-means approaches can achieve a high recall because they distribute clusters globally. However, when applying on a high noise-level data set, those clusters become unreliable and their generated links lead to many incorrect estimations. Thus, they have low precision on practical data sets under high noise-level conditions.

Trace Merging. Trace merging approaches are straightforward by starting with a single trace as an initial graph and incrementally adding new traces into the graph. Gradually, a complete graph is formed after merging all GPS traces. Typical algorithms in this class include work in [8, 19, 4, 22]. This type of map inference approaches also requires an accurate GPS data set to get high precision. Under high noise-level or low sampling interval conditions, trajectory segments between two consecutive points are not accurate, e.g., they may cross some areas that do not exist any roads. The trace merging approaches are not robust enough to deal with real data sets with high noise levels or low sampling intervals.

Kernel Density Estimation (KDE). The KDE approaches adopt the image processing technique to plot GPS points or trajectory segments on a discrete image and to calculate the density of each pixel. They remove low-density pixels to reduce the impact of noisy points. High-density pixels are more likely to have potential roads. The approaches build contour lines for high-density areas and compute road centerlines by creating a Voronoi diagram on contours. Existing KDE algorithms are published in work [10, 9, 20, 21, 7, 2, 23, 15]. The KDE algorithms can achieve a really good precision on real data sets with high noise levels. Due to the spareness of sampled GPS points on some residential roads, the algorithms overlook these roads on the image processing, therefore leading to a low recall.

Our proposed algorithm is a clustering-based approach by grouping GPS points globally contributing to a high recall. However, we infer road segments by creating and tuning spatial-linear clusters, which group potential GPS points recorded from the same road segment. Our generated clusters are closer to the linear property of most roads. To the best of our knowledge, the KDE-MM is the best state-of-the-art method as shown by the experimental results in [3, 6, 7]. In our experiments, we show that our proposed algorithm can achieve much better accuracy than KDE-MM.

7. CONCLUSIONS AND FUTURE WORK

We propose the spatial-linear clustering algorithm to improve the map inference accuracy. Our algorithm adopts a spatial-linear cluster to group a set of GPS points potentially recorded from the same road segment. We demonstrate that this is a better clustering technique compared to the state-of-the-art methods through conducting comprehensive experiments on real data sets. Experimental results show that our proposed algorithm improves the F_1 score on average 10.7% compared to the best state-of-the-art method. Furthermore, we demonstrate the advantages of our algorithm in terms of discovering unidirectional roads, which are difficult to detect via the state-of-the-art methods. In addition, we demonstrate that our algorithm performs significantly better than existing methods for inferring frequently travelled road segments with dense GPS sampled points.

Future directions:

- *Mining road features:* We can discover more useful information about road map data from GPS traces such as the number of lanes, speed limits and type of roads.
- *Complex intersection modeling:* The modeling of intersections can be more sophisticated such as no right-turn at an intersection. This is beneficial to giving driving instructions in a navigation system.
- *Traffic events monitoring:* Trajectory segments in a cluster will be useful to identify average speeds of a road. This information can be exploited for detecting the changes of traffic flow under certain events such as accidents or road constructions.
- *Customized maps for moving objects:* Building customized maps for moving objects is a challenging problem such as the migration of animals. Map inference automates the process of constructing maps for the discovery of their moving patterns.

8. REFERENCES

[1] G. Agamennoni, J. I. Nieto, and E. M. Nebot. Robust inference of principal road paths for intelligent transportation systems. *IEEE Transactions on Intelligent Transportation Systems*, 12(1):298–308, 2011.

[2] M. Ahmed, B. T. Fasy, M. Gibson, and C. Wenk. Choosing thresholds for density-based map construction algorithms. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems, Bellevue, WA, USA, November 3-6, 2015*, pages 24:1–24:10, 2015.

[3] M. Ahmed, S. Karagiorgou, D. Pfoser, and C. Wenk. A comparison and evaluation of map construction algorithms using vehicle tracking data. *GeoInformatica*, 19(3):601–632, 2015.

[4] M. Ahmed and C. Wenk. Constructing street networks from gps trajectories. In *Algorithms - ESA 2012*, volume 7501 of *Lecture Notes in Computer Science*, pages 60–71. Springer Berlin Heidelberg, 2012.

[5] N. Beckmann, H. Kriegel, R. Schneider, and B. Seeger. The r*-tree: An efficient and robust access method for points and rectangles. In *Proceedings of the 1990 ACM SIGMOD International Conference on Management of Data, Atlantic City, NJ, May 23-25, 1990.*, pages 322–331. ACM Press, 1990.

[6] J. Biagioni and J. Eriksson. Inferring road maps from global positioning system traces. *Transportation Research Record: Journal of the Transportation Research Board*, 2291:61–71, 2012.

[7] J. Biagioni and J. Eriksson. Map inference in the face of noise and disparity. In *SIGSPATIAL 2012 International Conference on Advances in Geographic Information Systems (formerly known as GIS), SIGSPATIAL'12, Redondo Beach, CA, USA, November 7-9, 2012*, pages 79–88, 2012.

[8] L. Cao and J. Krumm. From GPS traces to a routable road map. In *17th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, ACM-GIS 2009, November 4-6, 2009, Seattle, Washington, USA, Proceedings*, pages 3–12, 2009.

[9] C. Chen and Y. Cheng. Roads digital map generation with multi-track gps data. In *International Workshop on Education Technology and Training and Geoscience and Remote Sensing. ETT and GRS 2008.*, volume 1, pages 508–511, Dec 2008.

[10] J. J. Davies, A. R. Beresford, and A. Hopper. Scalable, distributed, real-time map generation. *IEEE Pervasive Computing*, 5(4):47–54, 2006.

[11] S. Edelkamp and S. Schrödl. Route planning and map inference with global positioning traces. In *Computer Science in Perspective, Essays Dedicated to Thomas Ottmann*, pages 128–151, 2003.

[12] T. Guo, K. Iwamura, and M. Koga. Towards high accuracy road maps generation from massive GPS traces data. In *Proceedings of IEEE International Geoscience & Remote Sensing Symposium, IGARSS, July 23-28, 2007, Barcelona, Spain*, pages 667–670, 2007.

[13] S. Jang, T. Kim, and S. Lee. Map generation system with lightweight gps trace data. In *The 12th International Conference on Advanced Communication Technology (ICACT)*, volume 2, pages 1489–1493, Feb 2010.

[14] S. Karagiorgou and D. Pfoser. On vehicle tracking data-based road network generation. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '12, pages 89–98, New York, NY, USA, 2012. ACM.

[15] C. Kuntzsch, M. Sester, and C. Brenner. Generative models for road network reconstruction. *International Journal of Geographical Information Science*, 30(5):1012–1039, 2016.

[16] J. Lee, J. Han, and K. Whang. Trajectory clustering: a partition-and-group framework. In *Proceedings of the ACM SIGMOD International Conference on Management of Data, Beijing, China, June 12-14, 2007*, pages 593–604, 2007.

[17] X. Liu, J. Biagioni, J. Eriksson, Y. Wang, G. Forman, and Y. Zhu. Mining large-scale, sparse GPS traces for map inference: comparison of approaches. In *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012*, pages 669–677, 2012.

[18] L. Moreira-Matias, J. Gama, M. Ferreira, J. Mendes-Moreira, and L. Damas. Predicting taxi–passenger demand using streaming data. *IEEE Transactions on Intelligent Transportation Systems*, 14(3):1393–1402, 2013.

[19] B. Niehoefer, R. Burda, C. Wietfeld, F. Bauer, and O. Lueert. Gps community map generation for enhanced routing methods based on trace-collection by mobile phones. In *First International Conference on Advances in Satellite and Space Communications, 2009. SPACOMM 2009.*, pages 156–161, July 2009.

[20] W. Shi, S. Shen, and Y. Liu. Automatic generation of road network map from massive gps, vehicle trajectories. In *Intelligent Transportation Systems, 2009. ITSC '09. 12th International IEEE Conference on*, pages 1–6, Oct 2009.

[21] A. Steiner and A. Leonhardt. Map-generation algorithm using low-frequency vehicle position data. In *Transportation Research Board 90th Annual Meeting, 23-27 January 2011, Washington, D.C., USA*, number 11-0486, pages 1–17, 2011.

[22] J. Wang, X. Rui, X. Song, X. Tan, C. Wang, and V. Raghavan. A novel approach for generating routable road maps from vehicle GPS traces. *International Journal of Geographical Information Science*, 29(1):69–91, 2015.

[23] S. Wang, Y. Wang, and Y. Li. Efficient map reconstruction and augmentation via topological methods. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems, Bellevue, WA, USA, November 3-6, 2015*, pages 25:1–25:10, 2015.

[24] Y. Wang, X. Liu, H. Wei, G. Forman, C. Chen, and Y. Zhu. Crowdatlas: self-updating maps for cloud and personal use. In *The 11th Annual International Conference on Mobile Systems, Applications, and Services, MobiSys'13, Taipei, Taiwan, June 25-28, 2013*, pages 27–40, 2013.

[25] S. Worrall and E. Nebot. Automated process for generating digitised maps through gps data compression. In *Australasian Conference on Robotics and Automation, Brisbane, Australia*, 2007.