Robust Road Map Inference through Network Alignment of Trajectories

Rade Stanojevic, Sofiane Abbar, Saravanan Thirumuruganathan, Sanjay Chawla[*]
Fethi Filali, Ahid Aleimat[†]

Abstract

In this paper we address the challenge of inferring the road network of a city from crowd-sourced GPS traces. While the problem has been addressed before, our solution has the following unique characteristics: (i) we formulate the road network inference problem as a network alignment optimization problem where both the nodes and edges of the network have to be inferred, (ii) we propose both an offline (`Kharita`) and an online (`Kharita`*) algorithm which are intuitive and capture the key aspects of the optimization formulation but are scalable and accurate. The `Kharita`* in particular is, to the best of our knowledge, the first known online algorithm for map inference, (iii) we test our approach on two real data sets and both our code and data sets have been made available for research reproducibility.

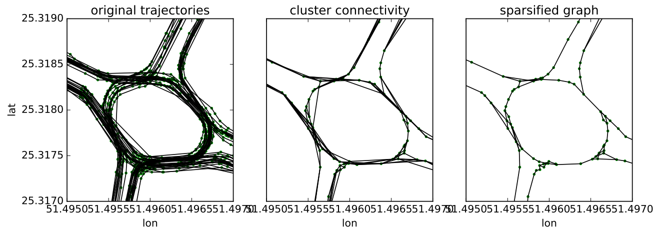


Figure 1: `Kharita` **process:** All raw trajectories passing through a roundabout (left). Centroids of the clusters and graph $G = (V, E_C)$ obtained from the raw trajectories projected onto clusters (middle). Final Kharita map obtained after sparsifying G (using a spanner). Map inference can be conceptualized as a multiple network alignment task.

1 Introduction

With the impending revolution of autonomous vehicles, interest in creating highly accurate geographical maps has come to the forefront. Large commercial efforts to build accurate maps have been announced[1]. In the near future, the most accurate maps may not be a public good but a property of private stakeholders. An important scientific (and societal) question is: *can we algorithmically construct accurate road network maps from crowd-sourced GPS traces in a cost efficient manner ?* Such an approach is likely to complement the popular collaborative efforts embodied in the Open Street Map (OSM) community.

Challenges for Accurate Map Inference: There are a number of technical challenges that a map construction algorithm using crowdsourced data needs to overcome including: (i) GPS sensors in smartphones, while reliable in general, can have non-trivial *GPS errors*. The errors are often acute in "urban canyons" with dense buildings or other structures; (ii) There is a substantial *disparity in data* due to the opportunistic data collection from smartphones. For example, while popular highways might have large number of data points, residential areas might only have handful of points; (iii) There exist a wide variety of *sampling rates* in which the GPS information is collected, often due to power concerns; (iv) Existing algorithms tend to overfit on the data sets on which they were tested. For example, many existing algorithms make some *implicit assumptions* that are specific to the road structures common to the United States and Europe. For instance, many countries in Asia and Middle East have circular intersections (roundabouts) that are notoriously hard for existing algorithms to capture. These challenges often results in a substantial quality gap between the maps produced by previous map construction algorithms and those obtained from road surveys.

Figure 1 provides an illustration of the two major steps - node inference through clustering and edge inference through graph spanners - of our offline algorithm. While our approach is conceptually simple, it significantly outperforms the state-of-the-art by improving the Biagioni TOPO score [5], the de-facto standard metric for measuring map quality, by up to 20%.

The rest of the paper is structured as follows. In Section 2 we formally define the map inference task. In Section 3 we associate the map inference task with the multiple network alignment problem. Section 4 in-

[*]Qatar Computing Research Institute, HBKU.

[†]Qatar Mobility Innovation Center.

[1]https://www.cnet.com/au/news/toyota-wants-to-make-google-mapping-tech-obsolete/

troduces `Kharita`, our two-step offline algorithm while Section 5 introduces the corresponding online variant `Kharita`*. Experimental details can be found in Section 6. We discuss the related work in Section 7 and conclude in Section 8 with a summary and directions for future work.

2 Definitions and Problem Statement

Definition 1. *A GPS point is a five-tuple* (lat, lon, t, s, a)*, where lat is the latitude, lon the longitude, t the timestamp, s the speed and a the angle.*

Definition 2. *A trajectory tr is a chronologically ordered collection of GPS points. We represent a trajectory tr as* $\{x_1, x_2, \ldots, x_{|tr|}\}$ *where each* x_i *is a GPS point and* $|tr|$ *is number of GPS measurements in trajectory tr.*

2.1 Problem Definition:

Given: A collection T of GPS trajectories.

Objective: Infer a geometric, directed G which represents the underlying road network. Each node of a graph represents a triplet $(lat, long, angle)$ and each directed edge represents a road segment.

3 Optimization Formulation for Map Inference

We briefly digress to view the road map inference task as a multiple network alignment problem (MNAP). In an MNAP setting, we are given a set of observable graphs (GPS traces) and the aim is to infer the underlying latent graph (road network). The network alignment problem was initally introduced in bioinformatics for aligning protein networks and in computer vision for image matching [15]. Much of the effort however was in pairwise alignment but there has been a recent interest in carrying out multiple network alignment [15]. We can cast the MNAP problem in terms of a quadratic extension of the facility location problem. The integer program formulation is shown in Figure 2.

$$\min_{x,y,S} \overbrace{\sum_j f_j y_j + \sum_{ij} d_{ij} x_{ij}}^{\substack{\text{Standard Facility Location} \\ \text{Objective:} \\ \textbf{Kharita: Node Inference}}} - g \overbrace{\sum_{ijk} O_{ik} S_{jl} x_{ij} x_{kl}}^{\substack{\text{Term to infer the edges of the} \\ \text{underlying network:} \\ \textbf{Kharita: Edge Inference}}} + \overbrace{\gamma \sum_{jl} S_{jl}^2}^{\substack{\text{Encourage Sparsity:} \\ \textbf{Kharita: Spanner}}}$$

$$\sum_j x_{ij} = 1 \ \ \forall i$$

$$x_{ij} \leq y_j \ \ \forall i, j$$

Figure 2: Relationship between the MNAP optimization formulation and the Kharita algorithm.

Here y_j is binary variable which is set to one when location j is set as a facility. This is equivalent to creating the location j as the vertex of the map graph. Similarly, x_{ij} is a binary variable indicating whether client i (GPS location) is assigned to location j. The traditional facility location problem generalizes the k-median problem where the number of facilities (clusters) opened are part of the objective. This corresponds to our situation as the number of nodes of the underlying graph are not known a priori but are part of the objective.

However, it is the term $\sum_{i,j,k,\ell} O_{ik} S_{j\ell} x_{ij} x_{k\ell}$ which highlights the complexity of the map inference problem. This part of the objective is trying to encourage an edge between two latent nodes in the underlying graph using the following rule: if client i is assigned to location j and client k to location l and if i and k are consecutive points on a GPS trace ($O_{ik} = 1$), then there should be an edge between location j and l and S_{jl} will be set to one. The regularizer term forces the matrix S to be sparse.

Despite tremendous progress in integer program solvers and even using heuristics (like Lagrangian solvers), the MNAP problem can only be solved for relatively small instance sizes [15]. `Kharita` resolves the complexity by decoupling the node and edge creation process.

4 `Kharita`: Offline Algorithm

We will now propose a two phased approach that first seeks to infer the vertices of the underlying graph through clustering and in the second step seeks to infer the edges between the vertices through graph spanners. A major advantage of this approach is that it can be easily modified to design an online version that can produce a routable map given one trajectory at a time.

4.1 Distance Metric A key novelty of our paper is the extensive use of the `angle` information that is widely available as part of GPS output. One can measure the distance between two (latitude, longitude) pairs (L_1, L_2) using Vincenty distance formula which we denote by $v(L_1, L_2)$ and it provides the distance in meters. The distance between two angles can be computed using the unit circle metric defined as $d_\circ(\alpha_1, \alpha_2) = min(|\alpha_1 - \alpha_2|, 360° - |\alpha_1 - \alpha_2|)$. For example, the distance between 350° and 10° is 20°. In order to compute the distance between two GPS points, we should consider both the location and the heading (angle of movement). Given two GPS points (L_1, α_1) and (L_2, α_2), we combine the aforementioned metrics as

$$d_\theta((L_1,\alpha_1),(L_2,\alpha_2)) = \sqrt{v(L_1,L_2)^2 + (\theta\frac{d_\circ(\alpha_1,\alpha_2)}{180^\circ})^2}, \tag{4.1}$$

We denote the heading penalty parameter through θ. Intuitively, we want to penalize points that are very close to each other based on location but have diametrically opposite direction. This is often the case for parallel roads where each lane corresponds to the traffic in opposite directions.

LEMMA 4.1. *The distance measure $d_\theta(.,.)$ is a metric.*

Proof. This follows directly from Cauchy-Schwarz inequality and the fact that both v and $d_\circ$ are metrics.

4.2 Node Inference Recall that our objective is to infer a directed graph from GPS trajectories. The first step is thus to infer the nodes of the graph which is carried out using a form of clustering. We cluster the GPS data points in (T) while ignoring the trajectory information. The cluster centers obtained will form the nodes V of the inferred road network G and are then connected using the trajectory information. In this paper, we use k-Means algorithm for clustering that has been shown to work well in prior work even without usage of the angle information [1, 11]. However, we have designed a new centroid initialization strategy customized for our problem.

Initialization: We start with an empty centroid list and go through the GPS points sequentially. A GPS point is added to the centroid list if there are no other centroid within a specified radius using the distance function $d_\theta(\cdot)$. This process ensures that every point is within a fixed distance to some centroid point. Our initialization approach results in the centroids being uniformly spread throughout the space of the map by making sure there are no two initial centroids within the specified distance of each other. It also determines the density of the clusters. For example using a small value, say $3m$, can result in inferring each lane in a highway as an independent road segment, while choosing large value may result into merging points from close (parallel) streets into the same cluster.

k-Means Clustering: After the initial cluster centroids are selected, we run the standard k-Means algorithm (where k is the number of centroids selected). In the assignment step, a GPS point is assigned to the closest centroid using the distance metric d_θ defined in Section 4.1. In the update step, the cluster centroid is updated using standard mean along the lat/lon coordinates and the mean of circular quantities for the heading coordinate. Formally, the centroid for a cluster of points S_i is:

$$m_i = (\overline{lat}, \overline{lon}, \overline{\alpha})$$

where:

$$\overline{lat} = \frac{1}{|S_i|}\sum_{x\in S_i} x.lat. \quad \overline{lon} = \frac{1}{|S_i|}\sum_{x\in S_i} x.lon$$

and

$$\overline{\alpha} = atan2(\frac{1}{|S_i|}\sum_{x\in S_i}\sin x.\alpha, \frac{1}{|S_i|}\sum_{x\in S_i}\cos x.\alpha).$$

It is well known that $\overline{\alpha}$ is a maximum likelihood estimator of the Von-Mises distribution (the spherical Gaussian) [13].

4.3 Edge Inference The output of the clustering stage is a set of cluster centroids $\mathcal{V} = \{v_1, v_2, \ldots, v_k\}$ where each centroid is a triple of $\langle lat, lon, a\rangle$. We now construct a cluster connectivity graph $G_C = (V, E_C)$ that integrates the clustering information and trajectory information. To create the edges we use the set of trajectories T_D as follows: Consider a trajectory $tr = \{x_1, x_2, \ldots, x_{|tr|}\}$. We transform the tr from a sequence of GPS points into a sequence of cluster centroids $tr_v = \{v_{i,1}, v_{i,2}, \ldots, v_{i,|tr|}\}$ where $v_{i,j}$ is the closest centroid to point $x_i \in tr$. For each pair of consecutive centroids $(v_{i,j}, v_{i,j+1}) \in tr_c$, we add an edge between them if they are distinct, i.e., $v_{i,j} \neq v_{i,j+1}$.

4.4 Graph Sparsification through Spanners The inferred graph $G = (V, E_C)$ from the previous section has potentially several redundant edges which makes G unusable for routing. Redundancy occurs when two nodes (centroids), v_a and v_b are directly connected by an edge even though they are far apart and there is path between v_a and v_b through other nodes. The cause of redundancy is due to the differential sampling rates of GPS trajectories and, more importantly, due to the fact that two points on a curved road that are physically close to each other may not be so in the combined metric d_θ which takes angular information into account.

Our objective is to construct a sparsified graph that retains the same connectivity information but also connects clusters that are near each other maintaining the road shapes. Consider a trajectory based edge $(v_i, v_j) \in E_C$. This implies that v_j is reachable from v_i through the underlying road network. However, due to the sampling rate, they might not necessarily be adjacent to each other. Hence, we would like to identify a smooth path between v_i and v_j that connects them through clusters that are near each other. Let us now formalize this problem. Given candidate graph

$G = (V, E_C)$, our objective is to obtain a sparsified graph $H = (V, E)$ where:

- $E \subseteq E_C$. i.e. H is a subgraph of G
- $\forall e = (v_i, v_j) \in E_C$, $d_H(v_i, v_j) \leq \alpha \times d_G(v_i, v_j)$. This constraint ensures that the sparsification preserves approximate distance between each pair of vertices in G. In other words, for any pair of vertices $(v_i, v_j) \in V$, their distance $d_H(v_i, v_j)$ in H is at most α times their distance $d_G(v_i, v_j)$ in G. Setting larger values of α generates sparser graphs. Throughout the paper we use $\alpha = \sqrt{2} \approx 1.41$ which removes the cross edges in orthogonal streets.

This problem can be abstracted as a well studied combinatorial problem of graph spanners [17]. Given a graph $G = (V, E_C)$, a subgraph $H = (V, E)$ is called a α-spanner of G, if for every $u, v \in V$, the distance from u to v in G is at most α times longer than the corresponding distance in G. There has been extensive prior work on developing efficient algorithms for spanner construction. For the purpose of our paper, we use a simple greedy spanner algorithm depicted in Algorithm 1. A naive implementation has a time complexity of $O(n^3 \log n)$ while it can be improved to $O(n^2 \log^2 n)$ using advanced data structures [17]. However, we empirically observe that by not recomputing the paths we can obtain a sub-quadratic runtime complexity while retaining full connectivity.

Algorithm 1 Greedy Spanner Algorithm

```
1: Input: $G = (V, E_C)$, $\alpha$
2: $E \leftarrow \emptyset$ , $H = (V, E)$
3: for each edge $(v_i, v_j) \in E_C$ in the order of decreasing weight do
4:     if $d_G(v_i, v_j) > \alpha d_H(v_i, v_j)$ then
5:         $E \leftarrow E \cup (v_i, v_j)$
6: return $H$
```

The pseudocode for Kharita is depicted in Algorithm 2.

Algorithm 2 Two-Phase Kharita Algorithm

```
1: Input: A collection $T$ of trajectories, $\alpha$
2: Output: A directed planar graph $G = (V, E)$.
3: Phase 1:
4:     $G_D \leftarrow$ DistinctPoints($T_D$)
5:     Let $S \leftarrow$ SelectSeeds($G_D$)
6:     $V \leftarrow k$-Means($|S|$, $G_D$)
7: Phase 2:
8:     Let $E_C \leftarrow$ EdgeAssignment($V, T_D$)
9:     $G(V, E') \leftarrow$ Spanner($V, E_C, \alpha$)
```

5 Kharita*: Online Algorithm

In this section, we present Kharita*, an online algorithm that can update the map as new trajectories arrive. In this algorithm, the two phases of the offline algorithm - node and edge inference (including sparsification) - are combined into one phase. This enables us to process trajectories that arrive in streaming fashion.

Streaming Setting: We consider the following streaming setting: our algorithm is provided a pair of GPS data points, x_i, x_{i+1}, that are taken consecutively. Our algorithm is generic enough to handle various GPS streaming models that have been previously studied. As an example, it generalizes the streaming model where the algorithm is provided one trajectory at a time. Given a trajectory tr, we can convert it to our streaming model by considering consecutive pair of points.

Challenges: Recall that our offline algorithm has two phases: (i) node and (ii) edge inference (including sparsification). Both of them required access to the entire data. The node inference phase consists of seed selection and running the k-Means algorithm. The graph sparsification required that the edges are provided in decreasing order of weight. However, in the streaming setting, we have to process each pair of GPS points immediately and it is not feasible to conduct expensive operations. Our solution combines online adaptation of k-Means clustering algorithm and online spanner algorithm.

Online Algorithm for k-Means and Spanners: We begin by providing a brief intuition behind the online k-Means algorithm [14]. In this problem, data points arrive one at a time and the objective is to provide a clustering that is competitive with the offline variant of k-Means that has access to all the data points. The algorithm assigns the first k points as cluster centroids. When a new data point arrives, it is assigned to the nearest centroid if the distance is less than some threshold f_i. If not, the data point becomes a new cluster on its own. As the number of clusters becomes larger, the threshold is periodically doubled thereby reducing the likelihood that a new cluster is created for new points. The online version of spanner algorithm is adapted from [16]. Given an α-spanner graph G and a new edge $e = (u, v)$, we add the edge to G if $\alpha \times w(e)$ is less than the distance $d_G(u, v)$.

Online Map Construction Algorithm – Kharita*: We begin with an empty graph $G = (V, E)$. When a pair of GPS points (x_i, x_{i+1}) arrives, we first densify them by creating an (artificial) set of equidistant points $P = \{p_1, \ldots, p_l\}$ where $p_1 = x_i$ and $p_l = x_{i+1}$ and each consecutive pair of points differ by a fixed densification threshold of sr meters. Each point p_i is assigned to the closest node (cluster) $v^* \in V$ if it is within a radius

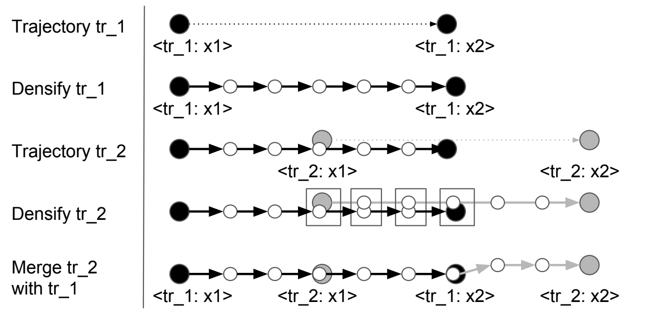


Figure 3: Example of streaming densification and merging of trajectories. The first trajectory tr_1 is densified and all its points are considered as new clusters, whereas the densification of the second trajectory tr_2 results in the assignment of some of its points to the clusters generated for tr_1

distance cr and the difference in angle is less than some heading angle tolerance ha. If not, we create a new node and assign p_i to it. Next, the algorithm checks whether an edge should be created from node v^*_{prev} (which correspond to the node to which the point pt_{i-1} was assigned) and nodes v^*. This edge is created if and only if the angle difference between v^*_{prev} and $v*$ as well as that between v^*_{prev} and the vector $\overrightarrow{(pt_i, pt_{i+1})}$ are both lower or equal to ha. In addition, we also ensure that the graph remains an α-spanner. Given the online and streaming nature of `Kharita`*, we had to give up on the angular distance $d_\theta(.,.)$ and replace it with a hard angle based threshold filtering to better control for angle differences. Figure 3 shows how the online algorithm proceeds step by step. Algorithm 3 provides the pseudocode for `Kharita`*.

Algorithm 3 `Kharita`* for online map inference

```
1:  Input: Trajectory $tr = \{x_1, x_2, \dots x_{|tr|}\}$, Road
    graph $G = (V, E)$, that can be empty
2:  Parameters: clustering radius ($cr$, in meters),
    sampling rate ($sr$, in meters), heading angle toler-
    ance ($ha$, in degrees)
3:  for each consecutive pair of points $e(x_i, x_{i+1}) \in tr$
    do
4:    $tr_D \leftarrow$ Densification(($x_i, x_{i+1}$), $sr$)
5:    for each point $p_i \in tr_D$ do
6:      $v^* = argmin(d(p_i, v_i)|v_i \in V)$
7:      if $v^*$ is within the distance tolerance $cr$ and
        angle tolerance $ha$ then
8:        Assign $p_i$ to cluster $v^*$
9:      else
10:       Create a new node $v^*$ for $p_t$
11:       $V \leftarrow V \cup \{v^*\}$
12:     if $d_G(v^*_{prev}, v^*) > \alpha \times d(edge(v^*_{prev}, v^*))$ then
13:       $E \leftarrow E \cup \{(v^*_{prev}, v^*)\}$
14:     $v^*_{prev} \leftarrow v^*$
15: return $G$
```

6 Evaluation

We have carried out a comprehensive set of experiments to evaluate both `Kharita` and `Kharita`* against a number of representative algorithms. We begin by introducing the datasets and evaluation metrics used. Next, we report how `Kharita` and `Kharita`* perform on the basis of these metrics relative to other approaches. Finally, we assess the robustness of our solution to different parameter settings.

6.1 Data and methodology As we discussed earlier, our map inference process uses data generated by a fleet of vehicles with GPS-enabled devices. In this paper we utilize two datasets from Doha (Qatar) and UIC (Chicago) with basic statistics reported in Table 1. The *Doha* dataset includes speed and heading information while the *UIC* dataset only has the location. For *UIC* we infer heading and speed information from two consecutive points.

Doha dataset covers a rectangle (in lat, lon coordinates) of 6km×8km in an urban region in the city of Doha with a mixture of highways, high and medium volume roads, capillary streets, and roundabouts. The *UIC* dataset covers an area of approximately 2km×3km in downtown Chicago and is generated by a fleet of University buses with relatively regular routes.

`Kharita` has two main parameters. In the Doha dataset we set $seed\_radius = 20m$ to cover 3-lane roads with minimal noise, while in UIC dataset we set $seed\_radius = 80m$ as we observe much higher noise levels, which go as high as $60m$ for a single road segment. In both cases we use $\theta = 2 \cdot seed\_radius$ to ensure that two datapoints from the same street going into opposite directions are not assigned to the same cluster. We also experimented with other choice of parameters and observe low sensitivity of final maps to the parameter choice. However, a more detailed examination of parameter space is left for future work.

6.2 Geometric and Topological Metrics Evaluating the quality of an automatically generated map is a challenging question. The de-facto standard for measuring the quality of the map is the *holes and marbles* method introduced by Biagioni and Eriksson [5].

GEO method evaluates how well the given map geometrically matches a ground truth map. Throughout the paper we use the OpenStreetMap (OSM) snapshot

Table 1: Characteristics of Datasets

Dataset	# Days	# GPS points	# Vehicles	Covered roads (km)
Doha	30	5.5M	432	300
UIC	29	200K	13	60

of the same region as the ground truth map[2]. GEO method samples points every five meters from both maps and puts a marble in each sampled point of the *inferred* map and puts a hole in each sampled point of the *ground truth* map. We say that marble (hole) is matched if there is a hole (marble) within a matching threshold, mt. We vary mt in the range of $5m - 30m$ and evaluate precision, recall and f_{score} where precision and recall are defined as :

$$precision = \frac{\text{\#matched-marbles}}{\text{\#all-marbles}}, \;\; recall = \frac{\text{\#matched-holes}}{\text{\#all-holes}}$$

TOPO method evaluates the topological (connectivity) characteristics of the map. Namely from a randomly sampled marble (hole) we first find all the marbles (holes) which could be reached from the starting point within a *radius*; throughout the paper we set the $radius = 2000m$. For the neighborhoods of each sampled starting point we calculate f_{score} as above and report the mean f_{score} over a sample of 200 randomly selected starting points. Note, that starting marble and starting hole belong to two different maps and are never match exactly; hence we enforce them to be within $1m$ distance and belong to the roads with the same direction (within a small angle difference).

6.3 Comparison with state-of-the-art In this section we compare Kharita with a representative set of state of the art (SOTA) map-inference algorithms that covers all the major approaches for map inference. We choose Edelkamp [10] for k-means based techniques, Cao [6] for trace merging based techniques, Biagioni [5] for KDE based techniques, and a recent work Chen [8] which is the closest to our work in that it uses data with similar features to ours (e.g., angle, speed.). For Cao, Biagioni and Edelkamp we use the source code developed and provided by the authors, while for Chen we use our implementation of the algorithm as the original implementation was not made available to us.

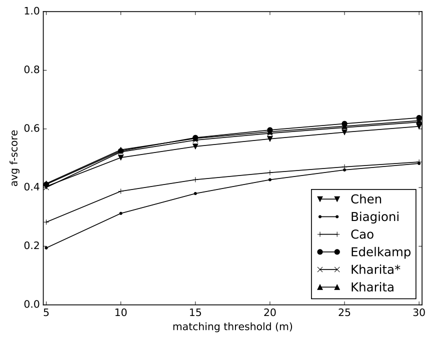
An important characteristics of some of the existing solutions is poor scalability. Namely it takes 3,558 and 6,621 seconds to run Cao/Edelkamp on a single day worth of data ($\approx 200K$ data points), with only one iteration of Cao's clarification step. Hence we report the comparison of the different algorithms using one day worth of data from Doha. Furthermore, trying to scale these algorithms to one week of data has resulted in "out of memory" problems in the machines we used for the evaluation.

6.3.1 GEO Comparison In Figure 4 we report the GEO f_{score} for the four map inference solutions, measured against the underlying OSM map in the rectangles of interest for both Doha and UIC datasets. The GEO f_{score} is determined by the amount of the data and their coverage of the underlying road network, but also depends on the edge-creation process. Namely some algorithms (such as Cao and Biagioni) are more conservative in the way they infer a road segment and hence have lower f_{score}; Other algorithms require a small number of trajectories to infer a particular road segment and consequently have larger f_{score}. Note that difference in f_{score} among those (Chen, Edelkamp, Kharita, Kharita*) is relatively small which is a result of the fact that geometrically each one of them closely covers the street segments which have at least one trajectory passing by. However, GEO method is oblivious to the connectivity structure of the road network captured by TOPO method.

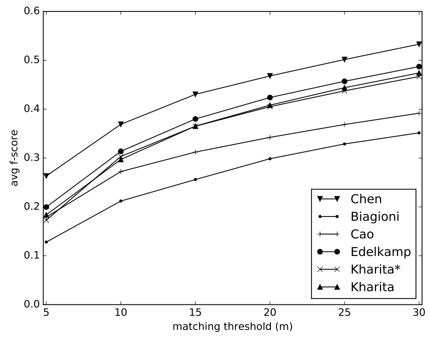
6.3.2 TOPO Comparison The true test of the map quality is the TOPO method. It measures how accurately intersection are inferred with the corresponding connectivity among different road segments. In Figure 4 we report the TOPO f_{score} for the same six maps and both datasets. We observe that Kharita and Kharita* manage to achieve largest f_{score} among the examined algorithms. This improvement comes from correctly inferring the connections (at the intersections) between the road segments; a task at which most of the previous methods fail, especially when oblivious of incoming trajectory information.

Kharita achieves TOPO f_{score} (with matching radius of $30m$) of 0.91 on the UIC dataset and 0.8 on the Doha dataset which improves state-of-the-art by 20% and 10%, respectively. In general, TOPO

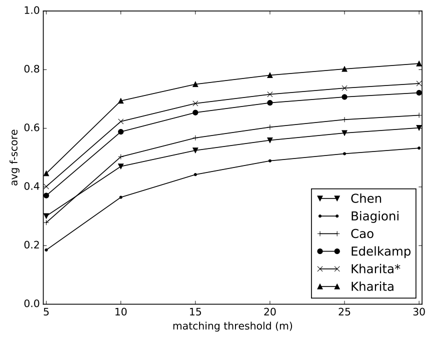
[2]As far as we can tell several intersections are not accurately represented by the OSM in our region of interest due to construction works, but this has relatively small impact on the matching scores.



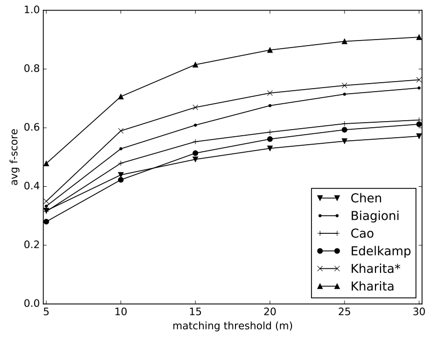
(a) Doha GEO



(b) UIC GEO



(c) Doha TOPO



(d) UC TOPO

Figure 4: GEO and TOPO f-scores for different map-inference solutions. On the more important TOPO scores both Kharita and Kharita* produce significantly more accurate results.

f_{score} are greater on the UIC dataset compared to Doha. This is the result of the fact that data is generated by buses following (mostly) regular routes, hence the network structure is fairly regular with many trajectories covering most inferred road-segments. In contrast, Doha dataset is much more diverse both in terms of routes and intersection types, and hence more difficult to infer accurately. Another important observation is that despite the fact that Kharita* is presented with a very limited future information regarding the incoming trajectories, due to the online streaming constraints, it was able to achieve higher f_{score} (up to 0.75) than all other algorithms except Kharita which acts offline. Finally, note that some solutions perform well on one of the datasets and not so well on the other suggesting that their approach is data-dependent, while Kharita effectively handles both datasets.

6.3.3 Visual Comparison Figure 5 shows the output of the six mapping algorithms on the same roundabout in Doha . Note that each map has its own way of creating edges which determines the final quality of the map. Both Kharita and Kharita* faithfully map the underlying road geometry and connectivity, with very few redundant edges. A visual inspection of the roundabout generated by Biagioni algorithm explains its poor GEO f_{score} performance as it misses too many road segments. Likewise, the fact that Chen's algorithm generates many extra undesirable edges increases its GEO f_{score}. However, because the edges are not correctly connected, its TOPO f_{score} performance is poor. It is important to notice that for navigation purposes, the most important aspect of the map to be captured is the connectivity, which translates in the case of complex intersections and roundabouts to being able to correctly infer all in/out links and turns.

6.3.4 Time performance of SOTA algorithms We next explore the time performance of the different solutions. Table 2 reports the execution time of the six algorithms to process 1 days of data from Doha and one month of data from UIC. Note that the two datasets are very similar in terms of number of GPS points ($\approx 200K$.) However, the rate at which points are generated is different, which is reflected on the times achieved. Doha data being denser, requires generally more time to process compared to the sparse UIC data.

As previously reported in [5][8], Edelkamp, Biogioni and Cao are expensive in terms of execution time. For Edelkamp, the algorithm runs 10 iterations in the case of Doha data and 19 iterations in the case of UIC data before it converges. The time we report in

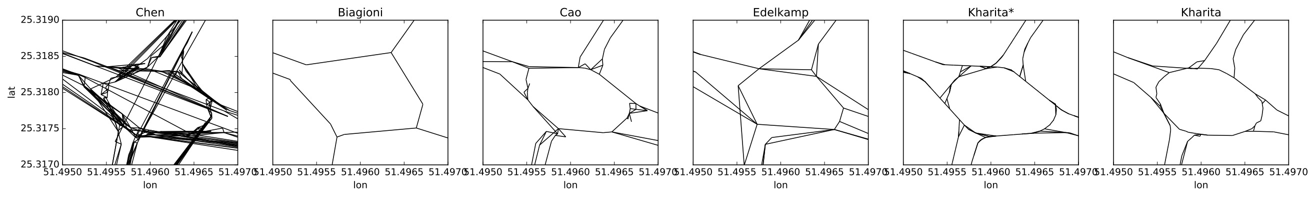


Figure 5: The six maps at prominent TV roundabout in Doha.

Table 2 as well as the f_{score} reported previously are all the results of these iterations. Cao needs almost one hour in the case of Doha to make one iteration of its clarification step. Thus, it was unrealistic to aim for a larger number of iterations. Due to its myriad steps, Biagioni algorithm required 01h20min to digest one day of Doha data. Unsurprisingly, our two solutions `Kharita` and `Kharita`* outperformed all existing algorithms. Note that `Kharita` runs in less time than its online counterpart `Kharita`* due to the use of the efficient "bash" ball-search in the KD-tree index to find nearest neighbors, that is not possible in the online setting as the data comes one point at a time.

Table 2: Comparative results for time performance

Algorithm	Doha (1 day)	UIC (1 month)
Edelkamp	6,621sec	1,254sec
Cao	3,558sec	1,637sec
Biagioni	4,719sec	1,280sec
Chen	1,078sec	381sec
`Kharita`	167sec	73sec
`Kharita`*	217sec	64sec

6.4 `Kharita` at scale In previous paragraphs we compared `Kharita` against other map inference solutions. In this section we will examine how the quality of the maps changes when more data is available and look into computational scalability of our algorithms.

We run both `Kharita` and `Kharita`* on slices of 1 day, 7 days, and 30 days of Doha data. We report for each slice the amount of GPS points processed, the execution time, and both GEO and TOPO f_{scores}. We empirically observed that the execution times scale in a near linear fashion with the size of the input dataset making the proposed algorithms highly scalable.

In terms of f_{score}, with more data GEO f_{score} improves for both `Kharita` and `Kharita`* as simply more data implies better coverage of the map. More interestingly, we observe that `Kharita` TOPO f_{score} also improves implying `Kharita`'s ability to improve the topological accuracy when additional data is available. Oddly, `Kharita`* TOPO performance suffers with more data due to the creation of spurious edges. Since `Kharita`* is online in nature, it is designed to work over short time windows and scalability is not a major concern of `Kharita`*.

Table 3: `Kharita` at scale

$\rightarrow$ **# days**	1	7	30
Data			
# GPS points	195,283	1,295,360	5,570,806
# Trajectories	2,834	22,907	77,314
Time performance (seconds)			
`Kharita`	167	1,543	8,190
`Kharita`*	217	1,798	5,512
GEO F1 scores - 30m matching radius			
`Kharita`	0.63	0.72	0.8
`Kharita`*	0.63	0.73	0.8
TOPO F1 scores - 30m matching radius			
`Kharita`	0.8	0.85	0.86
`Kharita`*	0.76	0.74	0.73

7 Related Work

Due to the widespread availability of smartphones and the advent of autonomous cars, the problem of map construction from opportunistically collected GPS traces have been extensively studied by various communities including data mining, geo spatial computing, transportation and computational geometry [10, 9, 6, 4, 5, 8, 3]. In this section, we provide a representative summary while additional details can be found in surveys such as [5, 2, 12].

Most prior work on map construction can be divided into three categories [5]. K-Means based algorithms perform clustering over the GPS points (typically, latitude and longitude, but sometimes also the direction). Once the clustering converges, the centroids are linked to get a routable map. Representative algorithms include [10, 1, 11]. Kernel density estimation (KDE) based algorithms such as [7, 9, 21] transform the input GPS points into a density discretized image that is then used to construct maps through image processing algorithms such as centerline detection. Finally, trace merging based approaches such as [6, 3] start with an empty map and incrementally insert traces into it based on distance and direction. `Kharita` is a hybrid algorithm that combines k-Means clustering followed by trajectory processing similar to trace merging. [4] proposed a hybrid pipeline based on KDE approach along with adaptive thresholds, geometry and topology refinement. Most recently, a number of novel methods have been proposed which borrow some techniques from the existing literature and also develop new intelligent methods to infer the road network from the GPS data [20, 23]. [8] proposed a supervised learning framework that can leverage prior knowledge on real-world road networks. There has been a series of papers that can infer additional road metadata such as intersections, number of lanes, speed limit, road type etc [11, 18].

Algorithms for spanners have been developed for general graphs [19], geometric graphs [17] and in streaming settings [16]. The distance approximating functionality of spanners has been used for robotics motion planning [22], telecommunication network design, serving as distance oracle for proximity problems [17, 19] etc. However, our work is the first to introduce graph spanners in the context of map inference.

8 Conclusion

In this paper, we proposed two efficient algorithms `Kharita` and `Kharita`* for constructing maps from opportunistically collected GPS information. We also provide a unique viewpoint of associating the map inference problem as a multiple network alignment problem. `Kharita` is a two-phase algorithm that clusters GPS points followed by a sparse graph construction using spanners. `Kharita`* is an online algorithm that can create and update the map when the GPS data points arrive in a streaming fashion. While our approach is conceptually simple it significantly outperforms the state-of-the-art due to efficient exploitation of angle and speed information and elegant handling of geographic information (via clustering) and topological structure (via graph spanner).

References

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

[2] 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.

[3] M. Ahmed and C. Wenk. Constructing street networks from gps trajectories. In *ESA*. Springer, 2012.

[4] J. Biagioni and J. Eriksson. Map inference in the face of noise and disparity. In *ACM SIGSPATIAL 2012*.

[5] J. Biagioni and J. Eriksson. Inferring road maps from global positioning system traces: Survey and comparative evaluation. *TRR-JTRB*, (2291), 2012.

[6] L. Cao and J. Krumm. From gps traces to a routable road map. In *Proceedings of the 17th ACM SIGSPATIAL*, pages 3–12, 2009.

[7] C. Chen and Y. Cheng. Roads digital map generation with multi-track gps data. In *International Workshop on Geoscience and Remote Sensing*, volume 1, pages 508–511. IEEE, 2008.

[8] C. Chen, C. Lu, Q. Huang, Q. Yang, D. Gunopulos, and L. J. Guibas. City-scale map creation and updating using GPS collections. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1465–1474, 2016.

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

[10] S. Edelkamp and S. Schrödl. Route planning and map inference with global positioning traces. In *Computer Science in Perspective*. 2003.

[11] S. S. et al. Mining gps traces for map refinement. *Data mining and knowledge Discovery*, 9(1):59–87, 2004.

[12] X. L. et al. Mining large-scale, sparse GPS traces for map inference: comparison of approaches. In *ACM SIGKDD 2012*.

[13] N. Fisher. *Statistical analysis of circular data*. Cambridge University Press, 1995.

[14] E. Liberty, R. Sriharsha, and M. Sviridenko, editors. *An Algorithm for Online K-Means Clustering*. SIAM, 2016.

[15] E. Malmi, S. Chawla, and A. Gionis. Lagrangian relaxations for multiple network alignment. *Data Min. Knowl. Discov.*, 31(5):1331–1358, 2017.

[16] R. Mariescu-Istodor and P. Franti. Cellnet: Inferring road networks from gps trajectories. *Tech report*, 2018.

[17] A. McGregor. Graph stream algorithms: A survey. *SIGMOD Rec.*, 43(1):9–20, May 2014.

[18] G. Narasimhan and M. Smid. *Geometric spanner networks*. Cambridge University Press, 2007.

[19] B. Niehfer, 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 *SPACOMM*. IEEE, 2009.

[20] D. Peleg and A. A. Schäffer. Graph spanners. *Journal of graph theory*, 13(1):99–116, 1989.

[21] W. Shi, S. Shen, and Y. Liu. Automatic generation of road network map from massive gps, vehicle trajectories. In *IEEE ITSC 2009*, 2009.

[22] W. Wang, D. Balkcom, and A. Chakrabarti. A fast online spanner for roadmap construction. *IJIR*, 2015.

[23] K. Zheng and D. Zhu. A novel clustering algorithm of extracting road network from low-frequency floating car data. *Cluster Computing*, pages 1–10, 2018.