Road Network Fusion for Incremental Map Updates

Abstract. In the recent years a number of novel, automatic map-inference techniques have been proposed, which derive road-network from a cohort of GPS traces collected by a fleet of vehicles. In spite of considerable attention, these maps are imperfect in many ways: they create an abundance of spurious connections, have poor coverage, and are visually confusing. Hence, commercial and crowd-sourced mapping services heavily use human annotation to minimize the mapping errors, and their response to changes in the road network is inevitably slow.
In this paper we describe `MapFuse`, a system which fuses the human-annotated map (in our case OpenStreetMap) with any automatically inferred map, thus effectively allowing quick map updates. In addition to new road creation, we study in depth the road closures which have not been examined in the past. By leveraging solid, human-annotated, maps with minor corrections we derive maps which minimize the trajectory matching errors due to both road network change and imperfect map inference of fully-automatic approaches.

Keywords: Map Fusion, Map Inference, Road Closures

1 Introduction

Map Fusion Problem: Generating accurate maps from geospatial data is an active area of research. A number of these works [10, 12, **?**,17] utilize crowd-sourced GPS data such as from smartphones. An alternate strain of work tries to use other sources such as satellite images [20]. Despite considerable interest and effort by the research community, the existing automatic map inference solutions have a number of shortcomings including: limited coverage, visually confusing layout, spurious roads, and imperfect turn restrictions. Hence commercial maps such as Google Maps, Nokia HERE, Apple Maps often use multiple sources of data information to generate initial maps, and then rely heavily on humans (both annotators and volunteers) to detect and correct the possible imperfections. However, the involvement of humans results in a very slow response in updating maps when a change in road network occurs. In many cities in Asia and Africa with with heavy construction, this results in substantial latency. One potential way to solve this issue is to automatically update the map using GPS traces given an existing map. However, most of those approaches are simple adaptations of classical map inference algorithms and suffer from the same disadvantages. In this work, we advocate for a new approach - Map Fusion - which automatically fuses two maps. One of the maps is a high-quality slowly updated map such as OpenStreetMap (OSM) [14] or Google Maps [19], while the other one is an

automatically inferred map with incomplete coverage and imperfect topological structure. Our proposed system, `MapFuse`, synthesizes a new map that overcomes the deficiencies of the two maps discussed above. In the rest of the section, we enunciate this overall approach.

1.1 Shortcomings of Fully Automatic Map Inference

As mentioned above, there has been extensive work (see surveys [10, 5, **?**]) on automatic map creation from GPS traces. However, these algorithms - both academic and commercial - suffer from a number of important shortcomings. We now highlight three of the major ones.

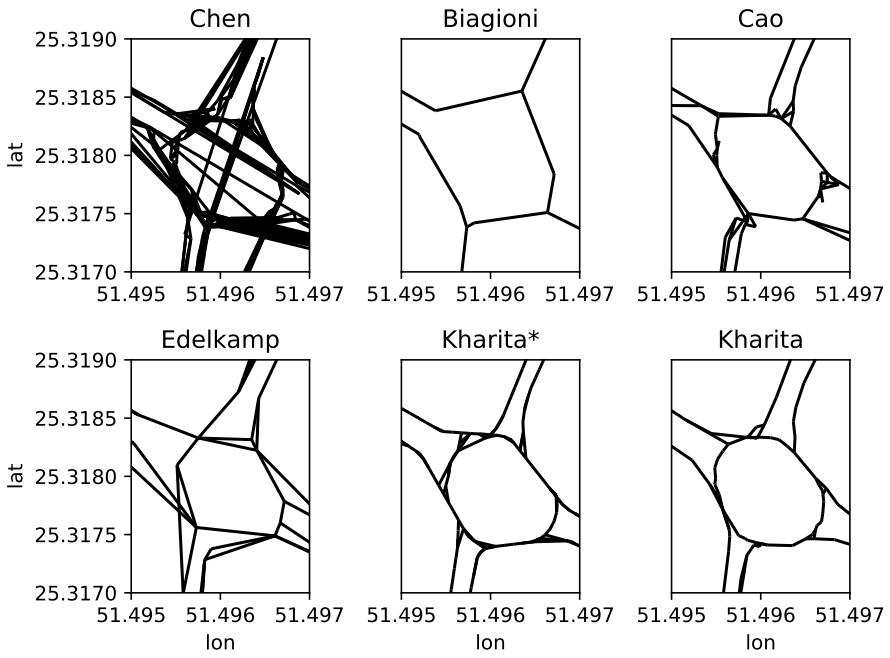

Fig. 1. Automatically inferred maps of 6 existing methods.

– *Poor coverage.* The popularity of roads segments in the road network (measured, say, in number of trajectories which pass by the segment) is very skewed. While a few road segments (e.g., those lying on a highway) carry a massive number of trajectories, a large fraction of roads serves only a handful of cars. Hence, a vehicle fleet which opportunistically collects the GPS data needs to collect a massive amount of spatial samples in order to have a decent coverage of the road network. In the case of the fleet whose data we

analyzed, if we denote by L the total length of all the roads in Doha (L is in the order of 10s of thousands of kilometers) our data which corresponds to the trajectories with overall length of $175 \cdot L$ covers only about 48% of the road segments (see Figure 3). In order to cover close to 100% of the road network with such opportunistic GPS probes, one would need to collect an order or two orders of magnitude more data, which in case of Doha would translate to 10s or 100s of millions of kilometers of driving. Thus, independent of the map-inference method one utilizes, one needs to have an extremely high-volume of opportunistically collected GPS data in order to cover large portions of the road network.

- *Visually confusing outlook.* Most of the existing approaches do not control for the visual appearance of their maps, and hence the resulting maps have rather confusing look and are not visually appealing. In Figure 1 we depict maps of a prominent "TV roundabout" in Doha derived by several well-known map-inference algorithms [10, 12, **?**,17, 24]. Due to different nature of their inference process, they all have some unique features, yet they all have spurious or missing road segments, which can confuse the end-user and the navigation system which may utilize such maps.
- *Low topological accuracy.* Possibly the most serious concern regarding the existing map-inference methods is their low topological accuracy. Namely, due to the GPS noise as well as the inability to efficiently handle such noise, all existing methods often miss the connections between road segments or infer non-existing connections between road segments. Such topological inaccuracies are absolutely non-tolerable, yet existing solutions have topological Biagioni $F1$-score[1] [10] in the range of 0.6 to 0.8 [9, 24]. We believe that a commercially acceptable map would likely need to have Biagioni $F1$-scores in the nearest proximity of 1.

1.2 Shortcomings of Automatic Map Updates

TomTom reports that 15% of roads change each year in some way [26]. The road changes are particularly common in many developing countries in Asia and Africa due to rapid construction of new roads. For example, thousands of kilometers of new expressways are being constructed each year in China and India for the past few years [25]. Automating the map update in a way that minimizes the disruption to the original map is of paramount importance. There has been extensive work on automatically updating an existing map using newly acquired GPS data (see Section 6 for details). However, many of these algorithms are often simple adaptations of existing batch map inference algorithms, and suffer from the same issues mentioned above. In addition, they often start with an automatically generated map which also suffers from the issues mentioned above. Hence the resulting map is often of substandard quality.

[1] Biagioni $F1$-score is a well known metric for measuring the topological accuracy of a map and lies in the range $[0, 1]$ with 0 being absolutely wrong map, and 1 being a perfect map.

1.3 Shortcomings of Hybrid Map Updates

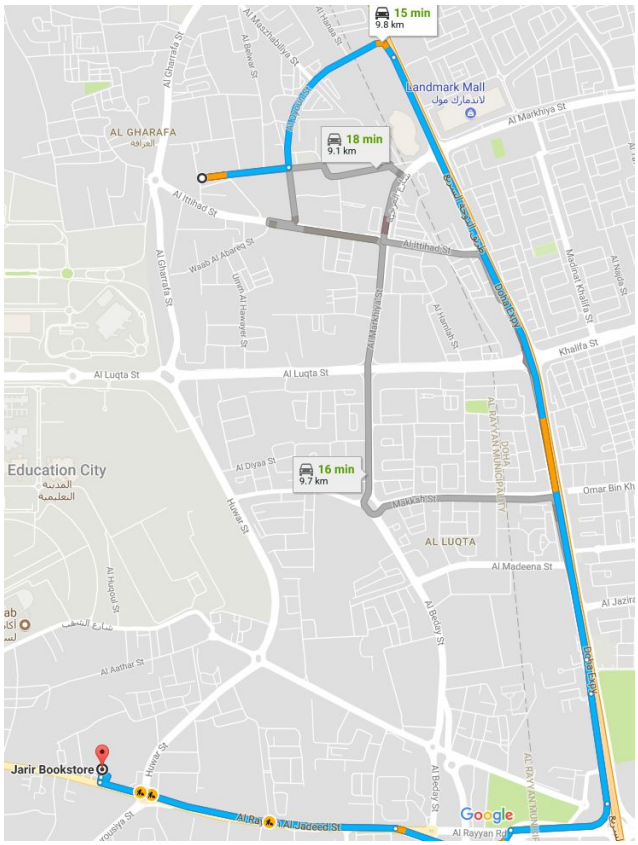

Fig. 2. Google maps route suggestion between two locations in Doha are almost twice longer (in length and duration) than the optimal route.

The above comments might indicate that perhaps a hybrid method involving automatic algorithms along with humans are the way forward. The substandard quality maps from purely automated means are often unacceptable for commercial map systems such as Google Maps, Apple Maps, Bing Maps, Nokia HERE, and Tom Tom. The creation of these maps is in many ways automated, however it requires human attention to examine possible places. For example, Google maps have a large team of so called operators who ensure the validity and consistency of the Google maps [1] and hence any possible change in the road network needs to be approved by one of the operators. Similarly, the largest global crowdsourced mapping effort OpenStreetMap (OSM) updates around 1M nodes per day. These maps have reasonably high accuracy in most cities with static road infrastructure.

However, even this approach has some fundamental limitations. First, due to the human in the loop, they suffer from slow update response when changes happen (see Figure 2). In many cities such as Doha, there are constant and large changes in road networks that are not updated regularly. Second, the automated algorithms often ignore the fact that most urban areas globally already have a fairly accurate map infrastructure. Not utilizing such great resource while constructing the map (as most automatic map inference solutions do) is unfortunate and hurts the overall map inference process. Let us illustrate this effect with a real-world example.

In the city like Doha, with a very dynamic road network,[2] the quality of existing maps is rather poor. For example, when one queries Google Maps for a route suggestion between two points in west Doha (see Figure 2), the suggested routes are almost twice as long (in both time and length) than the optimal one. Even though the optimal route has existed for over a year, the Google Maps has not updated the relevant portion of the road network to reflect the current road layout.

1.4 Proposed Approach

In this paper we propose `MapFuse`, a system for map fusion which automatically merges two maps. Specifically, we seek to fuse (1) s high-quality slowly updated map such as OSM [14] or Google Maps [19] and (2) an automatically inferred one, with incomplete coverage and imperfect topological structure. `MapFuse` produces a map which overcomes the deficiencies of the two maps discussed above.

In contrast with the existing approaches on map updating, which update the existing map (say OSM) by using a set of GPS trajectories via a specific map-inference tool, `MapFuse` is oblivious to the map inference approach one wishes to use to capture the road network segments and the interconnections between them. Hence we can fuse *any* map to the existing underlying map. This is important because existing map inference solutions suffer from a number of issues, and future solutions will most certainly rectify many of those. Fusing such better-inferred maps will most certainly lead to higher quality maps.

Finally, a very relevant aspect of map updating are road closures (both temporary and permanent) which are overlooked by the previous work on map updating, and which focuses only on new road additions [22, 26]. We use the GPS trajectory data to understand the road dynamics and infer the road closures as soon as they happen.

Summary of Contributions: We make the following contributions in our paper.

- We introduce a novel problem of map fusion, that seeks to update a base map with another inferred map, as a geometric graph matching problem and show it can be treated as a minimal vertex cover problem on an appropriately-defined bipartite graph.

[2] Influenced by a rapid construction of the city metro and a number of ongoing infrastructure projects.

– Due to the size of the graphs representing the two maps (which can have hundreds of thousands of nodes) the polynomial solution to the bipartite vertex cover problem is not practical and we propose an efficient heuristic that fuses two maps.
– We suggest a new methodology for inferring closed road segments which utilizes dynamic statistics of the roads as well as a node centrality measure. As an unexpected advantage of our closure detection we identify the errors in the OSM maps (e.g., we can automatically pinpoint several roundabouts which are represented in the OSM as two-way roads, while they are obviously one-way only) which can be harmful to the navigation systems.
– Using a set of GPS trajectories from a fleet of vehicles in Doha we demonstrate that the fused map is more accurate than either of the two maps, and reduces the average/median/99th-percentile trajectory matching error by 30%.

2 Problem Formulation

A common representation of a map in the map-inference literature is a directed graph as following. A map is a geometric graph $G(V, E, L)$, where V is the set of vertices, $E \subseteq V \times V$ is the set of edges connecting pairs of vertices, and $L : V \to \mathbb{R}^2$ is a location function which assigns coordinates (latitude and longitude) to each vertex.

Given two instances of such graphs (maps), G_1 and G_2, our goal is to create a new fused graph $G_f = f(G_1, G_2)$ which preserves some properties of the source graphs. In particular, we wish for the connectivity of the fused graph to subsume the connectivity of the source graphs. However, we also wish to do so with the minimum number of edges, in order to avoid unnecessary and spurious ones.

In order to express the connectivity property, we consider the set of shortest paths π_i within each graph G_i. The fused graph G_f should be so that

$$\forall p \in \pi_i,\ \exists \hat{p} \in \pi_f \ \text{ s.t. } d(p, \hat{p}) \leq \theta,\ i \in \{1, 2\}, \tag{1}$$

where $d(\cdot, \cdot)$ is a suitable distance function between paths which takes into account their geometry, and θ is a user-specified tolerance parameter. In our paper we use the following distance function

$$d(p_0, p_1) = \min_{i=0,1} \max_{u \in p_i} v(u, p_{1-i})$$

where $v(u, p)$ is the minimum distance between a point u and path p measured in meters. Thus a small $d(p_0, p_1)$ indicates that one of the two paths can be matched onto the other.

In addition, we wish to find the "minimum" such graph, i.e., the one that minimizes the sum of the lengths of its shortest paths:

$$\arg\min_{G_f} \sum_{p \in \pi_f} \ell(p).$$

This problem formulation can be reconducted to a minimum vertex cover problem on a suitably-defined bipartite graph $H(\pi_1, \pi_2, F)$. The two sets of vertices in H are all the possible shortest paths in G_1 and in G_2 (π_1 and π_2, respectively). There is an edge (u, v) between two elements u and v if their distance is below the threshold, i.e.

$$(u, v) \in F \iff d(u, v) \leq \theta,\ u \in \pi_1, v \in \pi_2.$$

Finding a minimum vertex cover M on H is equivalent to finding a minimum set of shortest paths such that their union maintains the connectivity property of the two source graphs. Therefore, G_f can be build from the union of these paths $M \subseteq \pi_1 \cup \pi_2$.

Note that due to König's theorem, the minimum vertex cover problem on a bipartite graph is actually tractable in polynomial time (and not NP-hard as in the general case). However, the size of the problem is $\mathcal{O}(n^2)$, and that to materialize H naïvely we need to compute $\mathcal{O}(n^4)$ distances between pairs of shortest paths.

Graphs representing the OSM and inferred maps in a large city such as Doha have more than $n = \mathcal{O}(100K)$ nodes. Hence the polynomial solution we hinted above is impractical. Therefore in the following section we propose a simple and efficient heuristic for tackling map fusion problem.

3 New Roads Detection

A common approach used in the literature [22, 26] to identify or detect new roads is the following. First, run a map matching algorithm between an existing map and a collection of GPS trajectories to identify the sub-set of trajectories that remain unmatched. Second, run some sort of road creation algorithm on the collection of unmatched trajectories to identify the new roads. Finally, link the newly created road segments to the existing map. That is, at the heart of the process, an algorithm is required to create roads from GPS points, which is exactly what all map inference algorithms do. Thus, it is hard to understand the real added value of map updating algorithms compared to what map inference algorithms do. For instance, if we assume that the initial map is very sparse, then it becomes clear that map update algorithms will be creating most of the road network, just like map inference algorithms do. Another way to look at the issue is to consider an initial empty map: in this case the map update and map inference become equivalent problems.

In our work, we take a slightly different approach. We assume that two maps are given to us. One that represents the base map (e.g., OSM) and another one that is generated using GPS traces via one of the many map inference algorithms available. The problem of map updating is then redefined as the merging of these two maps.

The function *FindOutliers* takes as input two maps M_1 (original) and M_2 (inferred), and generate a set of outliers. Outliers are set of nodes in the map M_2 which are at distance at least θ (here we use $\theta = 20m$). Mappings link nodes in

M_2 to M_1, whereas outliers are those nodes in M_2 that have no correspondents in M_1. These nodes are considered as candidates to be part of new road segments not covered in M_1. Our road addition procedure (see Algorithm 1) Works as follows.

Algorithm 1 `MapFuse`

1: **Input:** Base road map M_1, inferred map M_2
2: **Parameters:** collision radius (r, in meters)
3: $outliers = FindOutliers(M_1, M_2)$
4: $DRS = Subgraph(M_2, outliers)$
5: **for** each $o \in outliers$ **do**
6: compute $distance(o, M_1)$
7: **end for**
8: $outliers = Sort(outliers)$ in decreasing order of distance to M_1
9: **for** each $o \in sorted(outliers)$ **do**
10: $sg = BFS(o, DRS)$
11: **for** each node $n \in sg$ **do**
12: **if** $distance(n, M_1) \leq r$ **then**
13: $merge(n, argmin(n, M_1))$
14: **end if**
15: $outliers = outliers - \{n\}$
16: **end for**
17: **end for**
18: **return** M_1

In line 4, the sub-graph of newly detected roads (DRS) in M_2 is generated from the outliers. In lines $5-8$, the outliers are sorted in a decreasing order of their geometric distance to M_1. The intuition here is that the farther a node is from M_1, the more likely that node lays on a new road segment not covered by M_1. Outlier nodes are then processed in their order as follows. For each node o, we run a breadth first search (BFS) in M_2 starting from o until it reaches a leaf node or a node that is within a radius r (e.g., 2 meters) from M_1. Leaf nodes are assumed to be dead ends of newly detected road segments whereas nodes within a radius distance r from M_1 are assumed to belong to M_1. Nodes in the latter case are then merged with their closest nodes in M_1 as per line 13.

It is not difficult to see that the output M_f of above algorithm satisfies the condition from the Eq. (1). All paths from M_1 are indeed in M_f and are obviously matched by paths of M_f, the nodes from M_2 which are more than θ away from M_1 eventually get merged into the M_f and clearly satisfy the matching requirement (1).

4 Closed Roads Detection

Recall that the input to our process is the original map M_1, GPS-level trajectory data and the automatically inferred map M_2. An important characteristic

of the road network are road closures, which are sometimes permanent, but often temporary. Unfortunately, road closures have been overlooked by previous map-inference/map-update literature and in this section we propose two novel techniques for inferring road closures. The first one is 'static', in that it infers the road closures on a fixed input of trajectory data on the roads which have been closed prior to the start of the data collection. The second technique is more dynamic, as it observes the time series of the trajectories passing by a given road segment and by looking for anomalies is such time-series it effectively detects the road closures on the segments which have previously carried some trajectories in the data.

4.1 Cold-start road closure detection

As we hinted above, trajectory data collection inevitably has a starting point which is determined by either the functionality of the probe and the back-end system which stores the data, or by privacy regulations which may require sensitive trajectory data to be deleted after a period of time elapses.

What makes detection of closed road segments (from map M_1) difficult is the fact that there is a very high skew in the frequency of trajectories on different road segments: some segments (e.g., highways) carry a large number of trajectories while others in the capillary roads may not carry even a single trajectory. In Figure 3 we show how many new segments are 'discovered' as more driving data is collected. If we denote by L the total length of the road network, after trajectories with total length of L, only about 10% of the unique road segments are touched by those trajectories. After the total trajectory length gets to $10L$ they touch around 22% unique road segments. With all trajectories in our dataset with total length of $175L$, we get to detect only about 48% of the road network.

Thus, therein lies a dilemma: is a segment from map M_1 which has not carried any trajectory a closed road segment or it simply did not see a trajectory due to its peripheral nature? To answer this dilemma we initially aimed to exploit the OSM meta-data of OSM road segments such as road type, speed limit, number of lanes or one-way tag. However, the OSM meta-data appears to be rather sparse and is unlikely to give us the relevant road importance score which would help answering the above dilemma.

We address the aforementioned question by evaluating the node betweenness centrality (BC)[3] in map M_1. The BC of a node acts as an indicator of the importance of the node in the graph M_1, and not-surprisingly we see a strong dependence between the centrality of a given road segment and the number of trajectories in our data that pass through it. As seen in Figure 4, the trend is that the more trajectories a node has the higher BC and vice versa. In Figure 5 we depict the empiric CDF of node BC for two classes of nodes: those who lie on

[3] We believe using another node-centrality measure would likely give similar results, though we do not evaluate the impact of the choice of centrality measure in this work. However, the use of betweenness is consistent with the problem definition in Section 2.

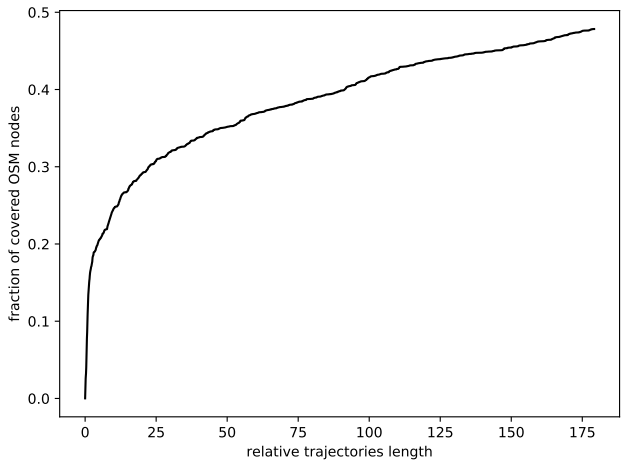

Fig. 3. Fraction of OSM nodes which are covered by at least one trajectory as a function of relative trajectory length defined as the ratio between the total length of all trajectories up to a point in time and total length of the road infrastructure. For close to 100% coverage one would need to have very, very, large trajectory dataset.

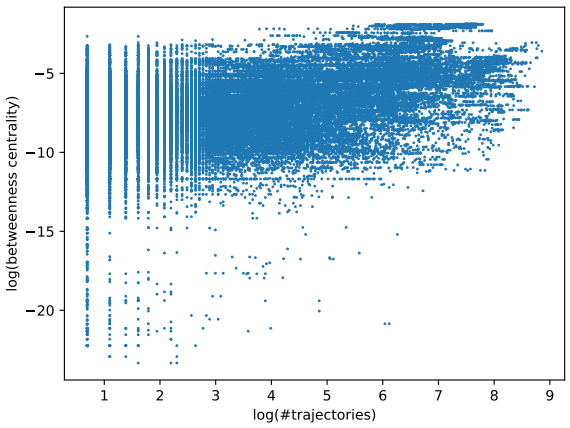

Fig. 4. Scatter plot OSM node betweenness centrality vs. number of trajectories passing through each node (logarithmic scale).

at least one trajectory and those who do not. We observe that BC mean/median among the nodes which lie on at least one trajectory is an order of magnitude larger than among the nodes which are not carry any trajectory.

Based on these observations, we declare the road segment closed if it has no trajectories passing by it and its BC is greater than the threshold γ. We choose $\gamma = 0.01$ to shave off the tail of the BC distribution among the nodes with no trajectories. Such γ identifies a handful of roads which are closed which we confirm by inspecting each one of them. In addition to those closed roads which are a sequence of closed nodes (with $BC > \gamma$) there are several nodes which are candidates for closure but are isolated from the other candidates. In order to declare the road closed we require that at least 100 meters segment (approximately 5-6 nodes) with corresponding nodes to be candidates for closure.

We would also like to point out that the proposed methodology allows us to infer inconsistencies between the OSM data and the traffic reality as captured by the GPS data. Namely, several roundabouts (formed by nodes with $BC > \gamma$) are represented in OSM as two way streets, however the clock-wise direction in those roundabouts is not matched by any trajectory and hence it is correctly identified as closed road (in that direction) which is an unexpected benefit of using the method described above.

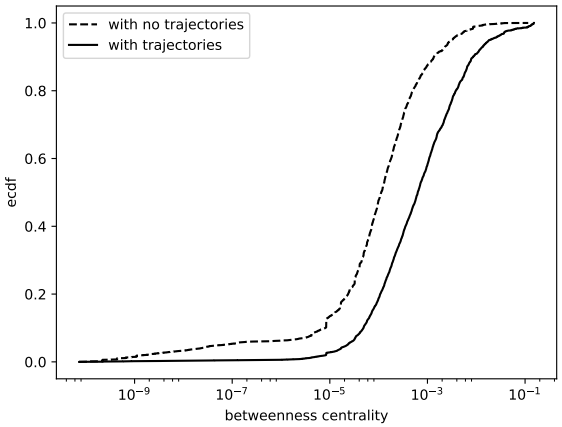

Fig. 5. Empiric CDF of node centrality for two classes of nodes: those who lie on at least one trajectory and those who do not. Node betweenness centrality is generally much smaller among nodes with no trajectories.

4.2 Road Closure as Anomaly Detection

The method described in the previous section detects the road closures which have happened before the data collection started and it is applicable only to

major roads - those with high betweenness centrality. However, for roads which get closed during the data collection we develop an anomaly detection module which monitors the traffic on each road segment and identifies "abnormal" gaps in the traffic stream.

For each node in the map M_1 we track the list of timestamps each time a trajectory is matched to that node. Note that sometimes a trajectory may have multiple records which are mapped to the same node (e.g., if the node is near a traffic light and the vehicle is static it will generate multiple data records which map to the same node in the map) and hence we only record the first match of the trajectory at the node and ignore the others.

As described previously, the road popularity (measured by number of trajectories which pass by it) distribution is rather skewed. In Figure 6 we plot the number of trajectories that are mapped to every OSM node in our dataset and observe that a large fraction of nodes have only a handful of trajectories which pass by it. Consequently, detecting anomalies on such low-frequency roads is rather challenging.

To detect the road closure during the data collection, each node v in the OSM graph maintains $mean_v(t)$: the *average* inter-arrival time among all trajectories which have passed that node until time t. In addition to that it also maintains the time *elapsed* since the last trajectory: $e_v(t)$. Note that for optimization reasons, the time elapsed is also computed when needed such as a case where a route query is triggered.

We declare the node closed at time t if:

$$e_v(t) > \alpha \cdot mean_v(t)$$

where α is a parameter which determines how conservative we are when deciding to declare the road closed. Small values of α may declare roads closed prematurely, while with large α it may take a long time before a closed road is declared as such.

To understand what is the right choice of α in Figure 7 we depict the histogram of the ratio between the maximum and the average trajectory inter-arrival time for all nodes which receive at least 2 trajectories per day, in average. We observe that the distribution of the max-to-average ratio is rather wide, and there is not clear cut-off point. However, most of the distribution is in the range between 1 and 40 with only a few nodes with the ratio greater than 40. Hence we choose $\alpha = 40$. Such choice results in only one closed road-section depicted in Figure 8 during our 2-month long observation. It involves a closed roundabout and respective access roads.

Finally, note that choosing a smaller α is likely to identify temporary road closures. However, since we could not confirm whether or not such nodes correspond to actual road closure or they simply fall in the tail of the distribution we leave the detailed discussion of temporary closures to future work.

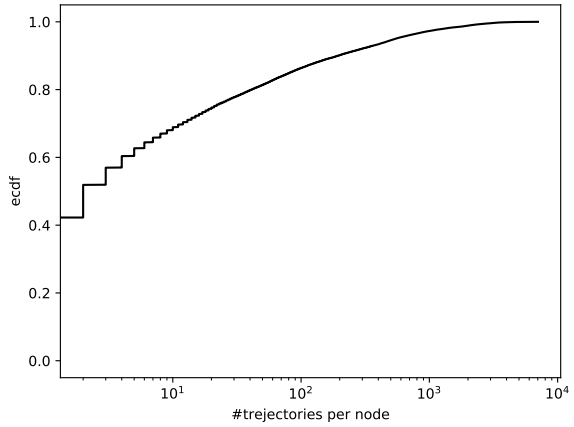

Fig. 6. Empiric CDF of the number of trajectories per node for all nodes in the OSM map. In our dataset only 18% of nodes have more than one trajectory per day in average.

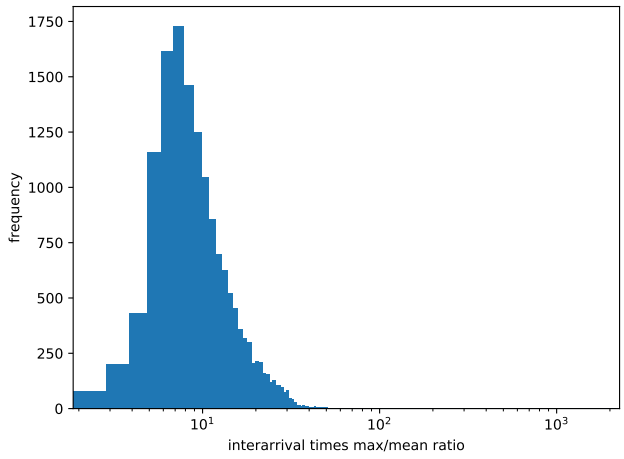

Fig. 7. The distribution of the ratio between maximum and average inter-arrival times for all nodes with at least 2 trajectory per day (in average). Most of the distribution falls in the range 1-40 with outliers corresponding to the nodes depicted in Figure 8

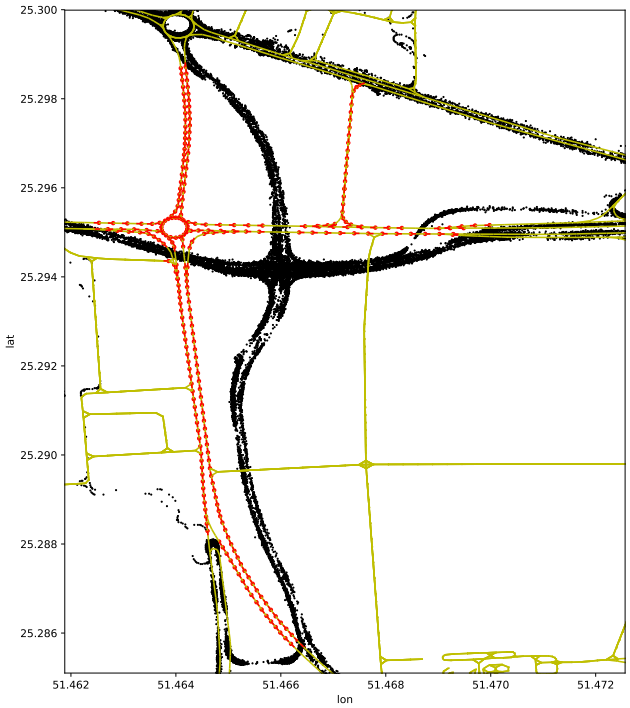

Fig. 8. Detected closed OSM road segments (red). OSM road network (yellow). GPS points after the road closure (black).

5 Evaluation

In this section we will exploit the GPS trajectory data to evaluate the quality of the fused map.

5.1 Data

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 the datasets from Doha (Qatar) with around 400 vehicles, 11 Million GPS points (sampled every $10s$). The dataset includes all GPS data points which fall into 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. Every data record contains: timestamp, latitude, longitude, speed, and heading of the moving direction of the vehicle. Heading is measured in angles against the North axis in degrees reporting values from 0 to 360°.

We preprocessed the data to eliminate those data points with speed $\leq 5kmph$ which are known to have non-trivial noise when reporting location.

5.2 Using trajectory data to evaluate maps

In this section we analyze how well can we match trajectories to the maps. For a map $\mathcal{M}$ and a trajectory $\tau = (p_1, \ldots, p_k)$ we denote by $\delta(\tau, \mathcal{M})$ the maximum distance between the points on the trajectory τ and $\mathcal{M}$:

$$\delta(\tau, \mathcal{M}) = \max_{p_i \in \tau} \min_{(u,v) \in \mathcal{M}} v(p_i, (u, v))$$

where $v(p_i, (u, v))$ is simple distance to line segment in geo-distance, measured in meters.

In our data we split all the trajectories in two subsets: training and test. We use the training set for constructing map $\mathcal{M}_2$ and the test set of trajectories for evaluating the matching distance. Since many trajectories from the same driver coincide, we make sure that trajectories from the same driver do not fall into both training and testing data. To that end, we split the set of drivers into training/test drivers (75%/25% split) and assign all the trajectories from the training/test driver into training/test trajectory dataset, respectively.

For automatic map inference we use `Kharita` [24], but note that using any other automatically inferred map [9, 12, **?**,17] could be used with relatively small (small, since only a handful of roads are being added to the map) impact on the final fused map.

For each trajectory in the test data we evaluate $\delta(\tau, \mathcal{M}_1)$, $\delta(\tau, \mathcal{M}_2)$, and $\delta(\tau, \mathcal{M}_1 \bigoplus \mathcal{M}_2)$, where $\mathcal{M}_1$ is the underlying (OSM) map, $\mathcal{M}_2$ is the automatically inferred map using the training trajectory data and $\mathcal{M}_1 \bigoplus \mathcal{M}_2$ is the merged map.

In Table 1 we report the mean, median and 99th-percentile trajectory matching distance for the three maps. All three metrics (mean, median and 99th-percentile) are minimized for the merged map and are around one third smaller than for automatic map. The improvements in trajectory matching come for two reasons. On one hand, trajectories which follow the new roads non-existing in the OSM map, but discovered by the automatic map, enjoy better matching in the merged map. On the other, the parts of the trajectories which correspond to the roads which are not covered in the training data, are likely to be covered in the OSM map and hence in the merged map.

$\delta(\cdot,\cdot)$	mean	median	99th-%
OSM	40.3m	9.3m	333m
automatic	12.3m	9.1m	70.4m
merged	8.1m	6.0m	53.4m

Table 1. Trajectory matching distance.

6 Related Work

Map Inference: Constructing maps from crowdsourced GPS traces has been extensively studied (see surveys [10, 5, **?**]. K-Means based algorithms cluster the GPS points and link the resulting clusters into a routable map. Representative works include [17, 2, 6]. Kernel density estimation (KDE) based algorithms such as [13, 15, 23] transform the GPS points into a density discretized image that are processed by image processing techniques to obtain maps. Trace merging based approaches start with an empty map and carefully add traces into it. Representative works include [12, 3].
Maintaining Maps: Recall that almost 15% of roads change every year in the US [26]. This number is even higher in many developing countries in Asia and Africa due to rapid construction of new roads. For example, thousands of kilometers of new expressways are being constructed each year in China and India for the past few years [25]. This necessitates research into work that maintain and update maps as and when new GPS data points arrive. Some representative work include [3, 6, 8, 11, 26, 30, 22, 27, 25]. However, most of these approaches do not have good practical performance and are very sensitive to differential sampling rates, disparity in data points, GPS errors etc. Often, these algorithms seek to directly extend one of the three approaches and suffer from bottlenecks arising from algorithmic step that is fundamental to it (such as clustering, density estimation, clarification, map matching) etc.

Additionally, while most of the prior work handle the simple case of new road additions, road closures are rarely addressed. CrowdAtlas [26] is exception that uses a simple heuristic in which each road segment is assigned an appropriate

timeout proportional (3x) to the maximum time observed between the traversal of two successive vehicles in a training window. To cope with the cold start problem, no timeout is set for a segment until it has accumulated at least a week of data and at least five traces. Thus, most residential roads have no timeout established.

Graph Matching: Given two graphs, identifying if one graph is a subgraph of another is known to be NP-Complete [18]. In fact, even identifying the minimal set of 'edits' to transform one graph to another is also NP-Complete [29]. However, it is possible to apply a number of heuristics for the case of road networks to solve this problem effectively. Matching of two road networks has been extensively studied due to its practical importance. The process of integrating different geospatial data to get new cartographic products is called map conflation. See [21] for a review of techniques used. Often, a wide variety of information including spatial features (such as distances, angles, shapes of the map) and topographical information (such as neighborhood) are used. For example, [28] proposed a heuristic probabilistic relaxation procedure to integrate multi-source geospatial data by using similarities between shapes. Recently, [16, 16] studied the problem of integrating authoritative geo-spatial data (such as OpenStreetMap) with crowdsourced GPS information. However, they use auxiliary information such as names and types of POIs that may not always be available.

7 Conclusion

In this paper, we proposed a new map update paradigm: map fusion. Instead using a customized map-inference algorithm when updating a map, we allow any map to be fused to the underlying (say OSM) map. Such fusion allows for quick map updates, with minimal changes to the high-quality underlying map. In addition to the map fusion, we also study in detail the road closure detection and propose two methods which efficiently detect road closure by comparing the statistical expectation of the traffic on a road segment against the actual traffic.

References

1. A. Lookingbill, M.W.M.: Project ground truth: Accurate maps via algorithms and elbow grease
2. Agamennoni, G., Nieto, J.I., Nebot, E.M.: Robust inference of principal road paths for intelligent transportation systems. IEEE Transactions on Intelligent Transportation Systems 12(1), 298–308 (2011)
3. Ahmed, M., Wenk, C.: Constructing street networks from gps trajectories. In: European Symposium on Algorithms. pp. 60–71. Springer (2012)
4. et al., C.C.: City-scale map creation and updating using GPS collections. In: ACM SIGKDD 2016
5. et al., M.A.: A comparison and evaluation of map construction algorithms using vehicle tracking data. GeoInformatica 19(3), 601–632 (2015)

6. et al., S.S.: Mining gps traces for map refinement. Data mining and knowledge Discovery 9(1), 59–87 (2004)
7. et al., X.L.: Mining large-scale, sparse GPS traces for map inference: comparison of approaches. In: ACM SIGKDD 2012
8. van den Berg, R.P.: All traces lead to roma. MS Thesis (2015)
9. Biagioni, J., Eriksson, J.: Map inference in the face of noise and disparity. In: ACM SIGSPATIAL 2012
10. Biagioni, J., Eriksson, J.: Inferring road maps from global positioning system traces: Survey and comparative evaluation. Transportation Research Record: Journal of the Transportation Research Board (2291), 61–71 (2012)
11. Bruntrup, R., Edelkamp, S., Jabbar, S., Scholz, B.: Incremental map generation with gps traces. In: Intelligent Transportation Systems, 2005. Proceedings. 2005 IEEE. pp. 574–579. IEEE (2005)
12. Cao, L., Krumm, J.: From gps traces to a routable road map. In: Proceedings of the 17th ACM SIGSPATIAL. pp. 3–12 (2009)
13. Chen, C., Cheng, Y.: Roads digital map generation with multi-track gps data. In: International Workshop on Geoscience and Remote Sensing. vol. 1, pp. 508–511. IEEE (2008)
14. Contributors, O.: Openstreetmap
15. Davies, J.J., Beresford, A.R., Hopper, A.: Scalable, distributed, real-time map generation. IEEE Pervasive Computing 5(4) (2006)
16. Du, H., Alechina, N., Hart, G., Jackson, M.: A tool for matching crowd-sourced and authoritative geospatial data. In: Military Communications and Information Systems (ICMCIS), 2015 International Conference on. pp. 1–8. IEEE (2015)
17. Edelkamp, S., Schrödl, S.: Route planning and map inference with global positioning traces. In: Computer Science in Perspective (2003)
18. Garey, M.R., Johnson, D.S.: Computers and intractability, vol. 29. wh freeman New York (2002)
19. Google: Google maps
20. Mnih, V., Hinton, G.E.: Learning to detect roads in high-resolution aerial images. In: European Conference on Computer Vision. pp. 210–223. Springer (2010)
21. Ruiz, J.J., Ariza, F.J., Ureña, M.A., Blázquez, E.B.: Digital map conflation: a review of the process and a proposal for classification. International Journal of Geographical Information Science 25(9), 1439–1466 (2011)
22. Shan, Z., Wu, H., Sun, W., Zheng, B.: Cobweb: a robust map update system using gps trajectories. In: Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing. pp. 927–937. ACM (2015)
23. Shi, W., Shen, S., Liu, Y.: Automatic generation of road network map from massive gps, vehicle trajectories. In: IEEE ITSC 2009 (2009)
24. Stanojevic, R., Abbar, S., Thirumuruganathan, S., Chawla, S., Filali, F., Aleimat, A.: Kharita: Robust map inference using graph spanners. arXiv preprint arXiv:1702.06025 (2017)
25. Wang, T., Mao, J., Jin, C.: Hymu: A hybrid map updating framework. In: International Conference on Database Systems for Advanced Applications. pp. 19–33. Springer (2017)
26. Wang, Y., Liu, X., Wei, H., Forman, G., Chen, C., Zhu, Y.: Crowdatlas: Self-updating maps for cloud and personal use. In: Proceeding of the 11th annual international conference on Mobile systems, applications, and services. pp. 27–40. ACM (2013)

27. Wu, H., Tu, C., Sun, W., Zheng, B., Su, H., Wang, W.: Glue: a parameter-tuning-free map updating system. In: Proceedings of the 24th ACM International on Conference on Information and Knowledge Management. pp. 683–692. ACM (2015)
28. Yang, B., Zhang, Y., Luan, X.: A probabilistic relaxation approach for matching road networks. International Journal of Geographical Information Science 27(2), 319–338 (2013)
29. Zeng, Z., Tung, A.K., Wang, J., Feng, J., Zhou, L.: Comparing stars: On approximating graph edit distance. Proceedings of the VLDB Endowment 2(1), 25–36 (2009)
30. Zhang, L., Thiemann, F., Sester, M.: Integration of gps traces with road map. In: Proceedings of the second international workshop on computational transportation science. pp. 17–22. ACM (2010)