Segmentation-Based Road Network Construction

Sophia Karagiorgou
School of Rural and Surveying Engineering
National Technical University of Athens
sokaragi@mail.ntua.gr

Dieter Pfoser
Department of Geography and GeoInformation Science
George Mason University
dpfoser@gmu.edu

Dimitrios Skoutas
Institute for the Management of Information Systems
R.C. ATHENA
dskoutas@imis.athena-innovation.gr

ABSTRACT

This work proposes a novel method that converts movement trajectories into a hierarchical transportation network. It utilizes an improved map construction algorithm on segmented input data based on types of movement. The produced hierarchical road network layers are then combined into a single network. This segmentation addresses the challenges imposed by noisy, low sampling rate trajectories and provides for a mechanism to accommodate automatic map maintenance on updates. An experimental evaluation is conducted using trajectories derived from GPS tracking taxi fleets and utility vehicles in Berlin, Vienna and Athens.

Categories and Subject Descriptors

H.2.8 [**DATABASE MANAGEMENT**]: Database Applications—*Data mining*

General Terms

Algorithms, Experimentation, Performance

Keywords

map construction, trajectories, road networks

1. INTRODUCTION

The widespread adoption of GPS enabled devices has enabled novel applications, such as automatically inferring the map of a transportation network by analyzing the traces of moving objects. The inherent inaccuracies and errors of the collected tracking data (GPS error, transmission errors, etc.) make the map construction problem very challenging. An example is given in Figure 6(a), which plots a set of vehicle trajectories from Berlin. Figure 6(b) shows the corresponding road network.

Existing map construction methods typically rely on uniformly distributed, frequently sampled, low-noise GPS traces, which limits their applicability and effectiveness in many real-world scenarios. In previous work [6], we have presented a method that relies on detecting changes in the direction of movement to infer intersection nodes, and then "bundling" the trajectories around them to create the network edges. Although that approach is more robust w.r.t. noisy GPS traces and different sampling rates, it still requires the tuning of several parameters to adapt to different network characteristics.

In this paper, we address the challenges of map generation from noisy, low-sampled tracking data, by analyzing, segmenting and reconstructing the underlying movement network in a layered form. We also introduce a proximity-based expansion algorithm around turn samples based on turn similarity. This layered approach allows us to segment the input dataset into groups of trajectories based on their characteristics and then process each group separately. Moreover, in this way we can also deal with changes and incorporate updates in an incremental fashion. Through an experimental evaluation, we show that this method, when compared to existing approaches, produces more accurate results when dealing with noisy and heterogeneous datasets with low and non-uniform sampling rates.

SIGSPATIAL'13, Nov 05-08 2013, Orlando, FL, USA
ACM 978-1-4503-2521-9/13/11.
http://dx.doi.org/10.1145/2525314.2525460

2. RELATED WORK

As examples of related map construction works we can cite the following. Several methods rely on *k-means clustering* of raw GPS data using distance measures and the heading to introduce cluster seeds at fixed distances along a vehicle trajectory (e.g., [4]). Other approaches are based on *Kernel Density Estimation* (KDE) and transform GPS traces to discretized images. They function well for frequently sampled data [3] and when there is a lot of data redundancy [2], but are sensitive with respect to noise. Recently, Wang et al. [7] addressed the problem of map updates using a KDE-based approach. Other approaches rely on a *computational geometry techniques* such as distance measures for map construction, e.g., [1]. These algorithms pose rather strict assumptions on GPS data coverage, or, they give partial quality guarantees. The final category involves *trace clustering* approaches. These adopt heuristics-based methods by aggregating GPS traces into an incrementally built road network (e.g., [5]). Similarly, in [6], the authors try to preserve the underlying connectivity of the road network embedded in the vehicle trajectories. Related to this, the contributions of this work are (i) the segmentation of the network into layers based on speed profiles, and (ii) the construction of a single road network by conflating network layers.

TRAJECTORYSEGMENTATION(T)

```
   ▷ Trajectories segmentation according to speed profiles
1  for (T_i ∈ T)
2      for (L_j ∈ T_i)
3          v̄(L_j) ← MEDIAN(v(L_{j−w}, ..., (L_{j+w}))
4          if v̄(L_j) ∈ C
5              if v̄(L_i) ∈ [C_min, C_max]
6                  C ← L_i
```

Figure 1: Segmentation of Trajectories

3. INFERENCE AND FUSION OF NETWORK LAYERS

This section introduces the new segmentation-based map construction algorithm called TRACECONFLATION. The input to the process comprises a set of vehicle trajectories. A trajectory is modeled as a list of spatiotemporal points $T = \{p_0, \ldots, p_n\}$ with $p_i = \langle x_i, y_i, t_i \rangle$ and $x_i, y_i \in R, t_i \in R^+$. The output of the process is a road network modeled as a directed graph $G = (V, E)$, where the vertices V correspond to intersection nodes and the edges E correspond to links. The process comprises three main steps described below.

3.1 Segmentation of Trajectories

First, the input trajectories are split into subsets of (sub-) trajectories according to their characteristics. This allows to treat each subset separately, e.g., by refining the parameters of the map inference algorithm accordingly, to derive different (but probably overlapping) portions of the network with higher accuracy.

We split and classify trajectories to different speed categories, e.g., "slow", "medium", "fast". A speed value is assigned to each line segment of the trajectory by dividing the length of the segment by the length of the time interval of its start and end points. To avoid excessive splitting due to changes of short duration (e.g., when a vehicle slows down at an intersection or a traffic light), we apply a sliding window across the trajectory, replacing the speed value of each segment by the median value computed over a series of consecutive line segments around it. The segmentation algorithm is outlined in Figure 1. For each line segment L_j of each trajectory T_i, its median speed is computed over a sliding window of width $2 \cdot w$ and the segment is then assigned to the corresponding speed category.

3.2 Construction of Network Layers

Next, a layer of the road network is inferred for each speed category. This is based on the TRACEBUNDLE algorithm previously introduced in [6]. TRACEBUNDLE identifies turn samples by detecting *changes in movement*, i.e., changes in direction and speed, and clusters them to derive intersection nodes. Clustering is based on proximity and angle difference by using static parameters. However, since different types of roads and intersections exist in a road network, such a setting often results in erroneous clusters, e.g., generating multiple nodes for a single intersection or generating a single node for multiple nearby intersections. Here, we further improve this algorithm with a more robust node inference process, in particular a *proximity-based expansion algorithm around turn samples based on turn similarity.*

INTERSECTIONS(T)

```
    ▷ Clustering turns to compute intersections
1   P ← ∅ ▷ Position samples set
2   P_S ← ∅ ▷ Turn samples set
3   C_T ← ∅ ▷ Turn clusters set
4   C_I ← ∅ ▷ Intersection nodes set
5   α_max ▷ angle difference threshold
6   d_max ▷ proximity threshold
    ▷ Position Samples → Turn Samples
7   For all (T[i] ≠ NULL)
8       P ← T[i] ▷ Positions samples of a single trajectory
9       α_d ← ANGULARDIFF(P[i − 1], P[i], P[i + 1])
10      if (α_d ∈ Angle)
11          α_in ← ANGLE(P[i − 1], P[i]) ▷ incoming angle
12          α_out ← ANGLE(P[i], P[i + 1]) ▷ outgoing angle
13          P_S .INSERT(P[i], α_in, α_out)
    ▷ Turn Samples → Turn Clusters
14  For all (P_S[i] ∉ C_T) ▷ not yet considered
15      NN_P ← FINDNN(P_S[i], d_max)
16      C_T ← COMPUTETURNCLUSTER(P_S[i], NN_P)
    ▷ Turn Clusters → Intersection Nodes
17  For all (C_T[i] ∉ C_I)
18      NN_C ← FINDCONTAINED(C_T[i])
19      C_I ← COMPUTEINTERSECTIONS(C_T[i], NN_C)
```

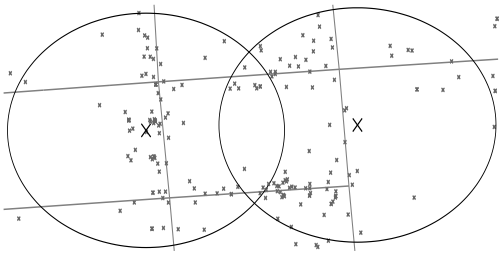
Figure 2: Intersections inference

The algorithm is outlined in Figure 2. First, turn samples are classified according to the change of direction between the incoming and outgoing edges. Next, samples that show a similar motion, in terms of absolute direction and spatial proximity, are grouped together into turn clusters. The turn clusters are constructed bottom up by finding for each turn sample its set of nearest-neighbor samples. Turn clusters stemming from different movement directions (left turn vs. right turn) but relating spatially to the same intersection are then grouped together to produce a single intersection node. This improved method results in intersection nodes being placed more accurately (see example in Figure 3).

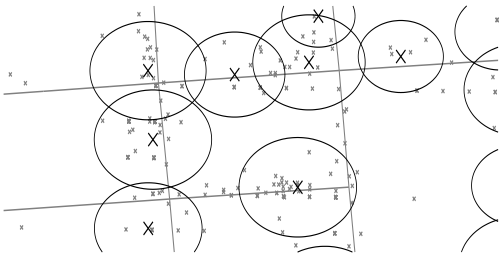
3.3 Conflation of Network Layers

The final step is the fusion of the generated layers for the different speed categories. This is done incrementally starting from higher speed layers and progressing to lower speed layers. The intuition for this is that higher speed layers correspond to avenues and highways and can be reproduced with higher accuracy. Fusion comprises: (i) finding intersection node correspondences among the different network layers, (ii) introducing new intersection nodes onto the existing links of a higher layer and (iii) introducing new links of lower layers for the uncommon portions of the road network.

The algorithm is outlined in Figure 4. Corresponding nodes across layers are identified by spatial proximity. Next, using a buffer region around intersection nodes of lower layers (e.g., medium network), we identify intersection nodes that are close to existing links of higher layers (e.g. fast network). These new intersection nodes are then mapped onto the existing link and effectively split it. Finally, new links for uncommon portions of the layered network are added by connecting them to previously introduced intersection



(a) Intersection nodes in TraceBundle



(b) Intersection nodes in TraceConflation

Figure 3: TraceBundle vs TraceConflation nodes

Tracking Data	Vehicles	Trajectories	Sampling rate (sec)	Trajectory length (km)	Speed (km/h)
Berlin	15051	26831	41.98	41116	35.23
Vienna	7434	12773	38.59	16106	33.68
Athens	120	511	30.14	6781	20.16

OSM Network	Nodes	Links	Length (km)	Area (km^2)
Berlin	5894	6839	360	36
Vienna	8081	9969	495	33
Athens	32212	39699	2000	168

Table 1: Statistics for datasets used

nodes. Figure 5 illustrates an example of this conflation process.

4. EXPERIMENTAL EVALUATION

We have conducted an experimental evaluation comparing TraceConflation to TraceBundle [6] on three tracking datasets for Berlin, Vienna and Athens, respectively. In each case, the corresponding road network obtained from OpenStreetMap was used as ground-truth. The statistics of the datasets are provided in Table 1.

A quick and easy way to get an overview of the quality of the inferred road network is by *visual inspection*, i.e., by overlaying it on the reference network and looking for similarities and differences. Due to space limitations, in Figure 6 we illustrate only the results for the Berlin dataset. For better illustration, we have marked some areas on the map where improvements of TraceConflation (Figure 6(d)) over TraceBundle can be observed (Figure 6(c)).

A more systematic and *quantitative evaluation* can be performed using the method introduced in [6]. Given the constructed and ground-truth networks, a common set of 500 pairs of nodes (origin, destination) is selected in both. Then,

ConflateNetworks(H, L)

```
E_H ← Edges(H)
N_H ← Nodes(H)
E_L ← Edges(L)
N_L ← Nodes(L)
N_HL ▷ intersection pairs
▷ Node alignment
For all N_L[i]
    N_HL ← (N_L[i], 1-NN(N_L[i], N_H))
▷ Node insertion to higher layer
For all (N_L[i] ∉ N_HL)
    E_i = On(E_H, N_L[i])
    if E_i ≠ null
        N_H.add(N_L[i])
        E_H.delete(E_i)
        E_i* ← E_i.split(N_L[i]) ▷ produces two links
        E_H.delete(E_i)
        E_H.add(E_i*)
▷ Link insertion
For all (N_L[i] ∉ N_H)
    N_H.add(N_L[i]) ▷ remaining nodes
For all (E_L[i] ∉ E_H)
    E_H.add(E_L[i]) ▷ remaining links
```

Figure 4: Conflation of Network Layers

		Discrete Fréchet distance			Average Vertical distance		
		min	max	avg	min	max	avg
Berlin	TraceBundle	18	428	183	8	209	106
	TraceConflation	14	398	137	6	201	98
Vienna	TraceBundle	15	410	111	4	212	94
	TraceConflation	12	382	103	2	198	81
Athens	TraceBundle	19	432	125	9	225	98
	TraceConflation	15	401	104	6	176	86

Table 2: Shortest-path comparison summary

the shortest paths between those pairs are computed in both networks. Performing a number of random shortest-path experiments, the geometric difference/similarity between the computed shortest paths can be used as a means to assess the quality of the constructed network. In particular, we calculate two similarity measures for each pair of shortest paths: (i) the Discrete Fréchet distance and (ii) the Average Vertical distance. This method produces an aggregated, quantitative comparison over whole portions of the road network. The evaluation shows a significant improvement in path similarity and, consequently, the constructed network: 93.8% of the paths showed increased similarity. Similar results were obtained for the Vienna and Athens datasets. Table 2 provides aggregated results for all three datasets.

5. CONCLUSIONS

This work describes a novel approach to the map construction problem based on segmenting the trajectory dataset using speed profiles, constructing separate map layers, and then conflating them into a single road network. The results of our experimental evaluation on three large-scale trajectory datasets from vehicles moving in Berlin, Vienna and Athens have shown significant improvement of the result quality when compared to existing approaches.

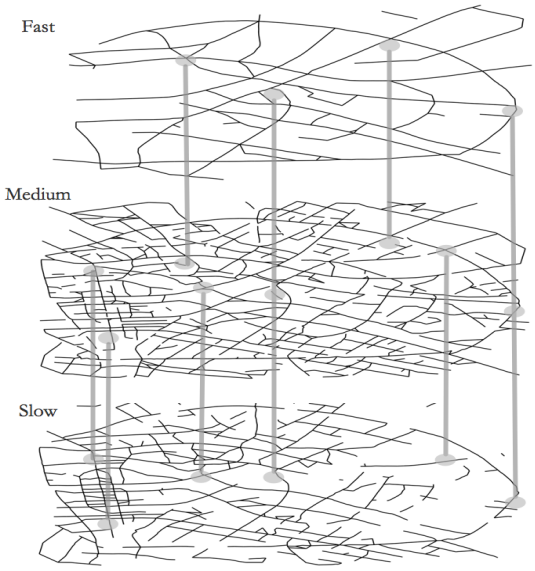


Figure 5: Stitching different network layers

Acknowledgments

This work was supported by the EU FP7 Marie Curie Initial Training Network GEOCROWD (FP7-PEOPLE-2010-ITN-264994).

6. REFERENCES

[1] M. Ahmed and C. Wenk. Constructing street networks from gps trajectories. In *Proc. 20th Annual European Symposium on Algorithms*, 2012.

[2] J. Biagioni and J. Eriksson. Map inference in the face of noise and disparity. In *Proc. 20th ACM SIGSPATIAL GIS conf.*, pages 79–88, 2012.

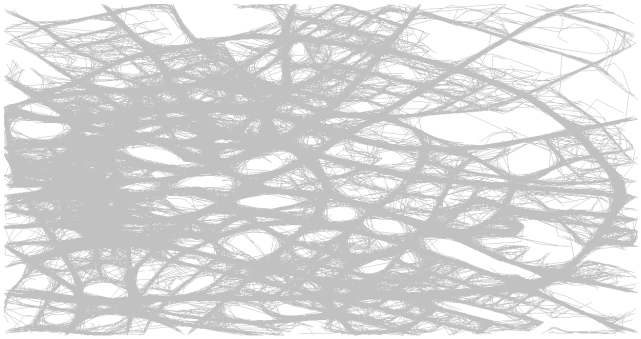
[3] C. Chen and Y. Cheng. Roads digital map generation with multi-track gps data. In *Proc. of the 2008 Int'l Workshop on Geoscience and Remote Sensing*, pages 508–511, 2008.

[4] S. Edelkamp and S. Schroedl. *Route planning and map inference with global positioning traces*, pages 128–151. Springer-Verlag, New York, 2003.

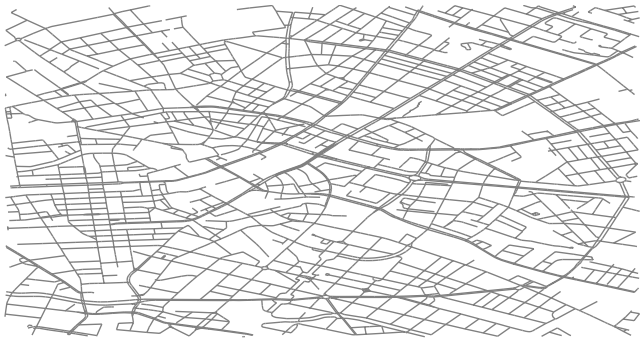
[5] A. Fathi and J. Krumm. Inferring the road network from gps data. In *Geographic Information Science*, volume 6292, pages 56 – 69, 2010.

[6] S. Karagiorgou and D. Pfoser. On vehicle tracking data-based road network generation. In *Proc. 20th ACM SIGSPATIAL GIS conf.*, pages 89–98, 2012.

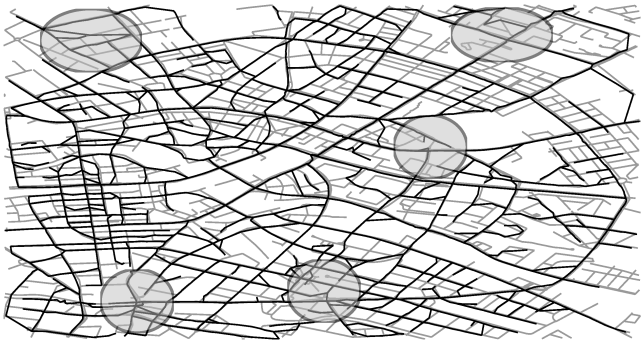
[7] Y. Wang, X. Liu, H. Wei, G. Forman, C. Chen, and Y. Zhu. Crowdatlas: Self updating maps for cloud and personal use. In *Proc. 11th MobiSys conf.*, 2013.



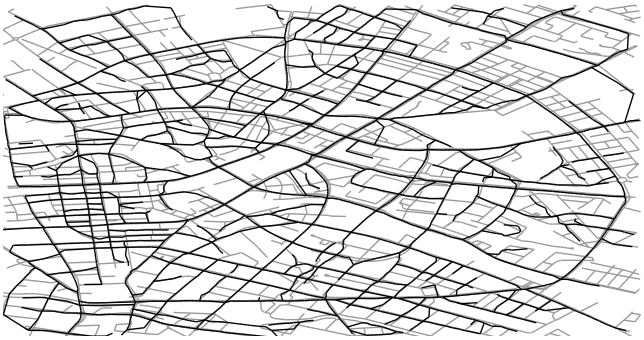
(a) Berlin–Vehicle tracking data



(b) Berlin–Corresponding road network (OSM)



(c) Berlin–TraceBundle



(d) Berlin–TraceConflation

Figure 6: Visual comparison of trajectories, original and generated road networks