On Vehicle Tracking Data-Based Road Network Generation

Sophia Karagiorgou
Institute for the Management of Information Systems
Research Center "Athena"
Artemidos 6, 15125 Maroussi, Greece
karagior@imis.athena-innovation.gr

Dieter Pfoser
Institute for the Management of Information Systems
Research Center "Athena"
Artemidos 6, 15125 Maroussi, Greece
pfoser@imis.athena-innovation.gr

ABSTRACT

Road networks are important datasets for an increasing number of applications. However, the creation and maintenance of such datasets pose interesting research challenges. This work proposes an automatic road network generation algorithm that takes vehicle tracking data in the form of trajectories as input and produces a road network graph. This effort addresses the challenges of evolving map data sets, specifically by focusing on (i) automatic map-attribute generation (weights), (ii) automatic road network generation, and (iii) by providing a quality assessment. An experimental study assesses the quality of the algorithms by generating a part of the road network of Athens, Greece, using trajectories derived from GPS tracking a school bus fleet.

Categories and Subject Descriptors

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

General Terms

Algorithms

Keywords

map generation, FCD, tracking data

1. INTRODUCTION

Road networks and more generally transportation networks are an interesting research subject in that they represent the principal data set for a large range of applications, including GIS, transportation systems, location-based services and Web mapping. Currently, such datasets are provided by two global players, Navteq and TeleAtlas. More recently, community-driven efforts have begun to challenge commercial efforts, at least for the context of non-critical applications (cf. [1]).

This work will address the challenges of evolving map data sets, specifically by working towards *automatic map and attribute generation* from massive amounts of vehicle tracking data. Our objective is to create an algorithm that automatically extracts the road network graph and related attributes such as road categories from tracking data obtained using GPS-based position samples (floating car data - FCD) for large vehicle fleets.

Advances in mobile computing have essentially led to a commodization of online navigation services with a considerable number of users now being able to determine and communicate their location. The advent of Web 2.0 applications that have positioning as their core theme further increases the amount of tracking data that is currently available for data analysis. From this point of view, algorithms that take tracking data as input and produce map data sets would be relevant, especially when considering disaster scenarios in which existing infrastructure is wiped out and ad-hoc networks need to be recorded. Besides deriving road networks, the presented approach can be used to identify implicit movement patterns in any kinds of spatiotemporal tracking data, e.g., animal migration, historic trade routes, etc. Existing approaches to the road network generation problem do exist. As we will see in the next section, both, the GIS and, more recently, the Computational Geometry communities have addressed this problem, each having their specific strengths and limitations.

The presented approach exploits the ubiquitous trajectory data in order to analyze, reconstruct and extract road network geometries enriched by attributes. Our heuristics-based approach relies on "bundling" the trajectories around intersection nodes. Intersection nodes are derived by detecting clusters in changes to movement patterns. Essentially, we identify areas in which different types of turns are detected and designate them as intersection nodes. Linking the trajectories to intersections allows us then to derive links and consequently the entire geometry of the road network. To assess the quality, we sample the generated road network and the actual road network by means of distinct and random shortest-path queries. Comparing the shortest-paths generated in both networks will give us an indication of the quality of the generated road network.

The remainder of this work is organized as follows. Section 2 describes related work and research efforts on road network generation techniques. Section 3 discusses the methods developed regarding trajectories clustering and calculation of intersection nodes, connecting intersection nodes and the generation of a road network. Additionally, we suggest an evaluation approach to provide quality guarantees of the generated road network in relation to the underlying network in Section 4. Section 5 presents an experimental evaluation and Section 6 gives conclusions and directions for future work.

2. RELATED WORK

Automatic map construction and automatic road network refinement for improved navigation services have been tackled using a variety of methods including probabilistic models, computational geometry algorithms such as shape matching or curve similarity, but also image processing methods. What follows is a discussion of such relevant approaches.

Various approaches exist to solve the problem of road network reconstruction. Road networks can be derived from satellite or aerial images by means of image processing techniques [14, 13, 15, 5, 20, 16]. For example, Tavakoli et al. [18] group together edges found by an edge detector into shapes representing buildings and roads. Some of the earliest works that address refining existing map datasets with the use of tracking data is [17]. It exhibits similarities to the present approach in that it applies clustering techniques to infer intersection models. However, no actual road network is derived and no quality estimates are given. Using AI techniques, Bruntrup et al. [7] build a road network by incrementally integrating trace data starting with a single trajectory and placing strict assumptions on the quality of the data. In [12], Guo et al. use statistical analysis of GPS traces to generate road maps. Chen et al. [10] use a Gaussian mixture model to infer the lane structure of roads from GPS data. However, their effort does not go beyond the definition of such a model. Both approaches have not been widely experimented on large-scale data, while our approach uses a portion of the city of Athens.

Several works exist that approach the problem using computational geometry methods with quality guarantees. Chen et al. [9] focus on detecting seed elements in the road network and connecting them subsequently. Aanjaneya et al. [2] view road networks as metric graphs and their focus is on computing the combinatorial structure, but they do not compute an explicit embedding of nodes (vertices) and links (edges). Both approaches are based on sub-sampling the trajectory data and then using an unordered set of points to derive the complete road network. Recently, Ahmed and Wenk [3] developed an incremental method that employs the Frèchet distance to match partial trajectories to a graph. While the authors give partial quality guarantees, their approach does not address the basic connectivity problem and how to measure the respective quality of a generated network.

Heuristics-based methods include Edelkamp et al. [11] addressing efficient algorithms for road segmentation, map matching, and lane clustering using GPS traces. Worrall et al. [19] compress GPS traces to infer a digitized road map and present their results only for small datasets. In [8], Lili and Krumm present a method to eliminate noise in GPS traces and Fathi and Krumm [4] provide an approach that detects intersections by using a trained detector. While a road network is finally derived, their approach works best for well aligned road networks.

Our approach differs in that we try to preserve the underlying connectivity of the road network embedded in the vehicle trajectories. Essentially, trajectories are clustered together based on intersection indicators (turn samples) and the final road network is derived by merging the trajectories that link the intersections. As such this approach works for arbitrary "movement networks". In addition and, to the best of our knowledge, there has not been any comparable effort for computing quality measures with respect to connectivity and spatial similarity for generated road networks.

3. ROAD NETWORK GENERATION

The contribution of this work will be to derive a road network by sampling it using vehicles and GPS tracking. By means of the (set of) algorithms that is discussed in the following, the tracking data is essentially reduced to the actual road network geometry. In addition, road categories are derived based on the amount of data that is available for particular road network portions. Our task will be to align the vehicle trajectories, so as to derive the actual road network underlying it. Figure 1 plots such tracking data with the actual road network being (at least visually) evident.

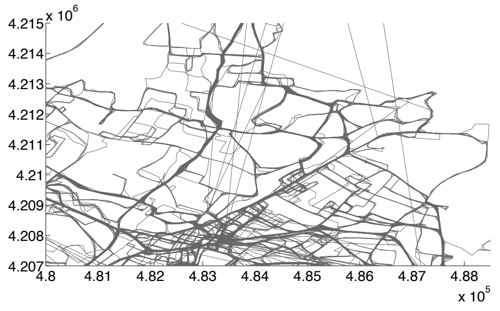


Figure 1: Tracking data example - Athens, school bus data

The algorithm to derive the road network involves three essential steps; (i) *identifying intersections*, i.e., use turns in vehicle trajectories as indicator for intersections, (ii) *connecting intersections*, i.e., create links between intersections by using trajectories, and (iii) *reducing the network graph*, i.e., collapse the links to create a meaningful road network graph. The remainder of this section describes the various steps of the methods in detail.

3.1 Data

A *road network* is made of basic pieces usually referred to as links. A link is an atomic road portion such as a road section between two intersections, or a ramp onto a highway. The link notion is introduced to reflect the connectivity between roads of possibly different importance. A road network is modeled as a directed graph $G = (V, E)$, whose vertices V represent intersections between links and edges E represent links.

The vehicle tracking data is commonly referred to as Floating car data (FCD), i.e., data generated by one vehicle as a sample to assess to overall traffic conditions ("cork swimming in the river"). Typically these data comprise basic vehicle telemetry such as speed, direction and most importantly the position of the vehicle in the form of vehicle tracking data. Having large amounts of vehicles collecting such data for a given spatial area such as a city (e.g., taxis, public transport, utility vehicles, private vehicles), it will create an accurate picture of the traffic conditions in time and space.

The resulting data comprises *vehicle trajectories*, which can be modeled as a list of space-time points $T = p_0, \ldots, p_n$ with $p_i = \langle x_i, y_i, t_i \rangle$ and $x_i, y_i \in R, ti \in R^+$ for $i = 0, 1, \ldots, n$ and $t_0 < t_1 < t_2 < \ldots < t_n$.

The recorded vehicle trajectories are affected by a measurement error due to the limited GPS accuracy and a sampling error due to the sampling rate (typically $30s$), having a great impact on the quality of the data set [6]. As in the case of map-matching, both errors need to be addressed by the respective algorithm, in our case, the map construction algorithm. However, in our experimentation, we introduced some heuristics that allow us to filter out outliers in the data (cf. Section 4).

3.2 Turns and Intersections

Given a vehicle trajectory, we use turns to detect intersection nodes of the road network. Specific indicators for turns are changes in the vehicle's movement in terms of speed and direction.

3.2.1 Indicators

Deriving from a common-sense understanding of vehicular movement, when turning, a vehicle (i) reduces its speed and (ii) changes its direction. Our approach uses a speed threshold in combination with a change in direction. Figure 2 gives an example of a trajectory with two position samples and the respective direction vectors. In the specific experiments of Section 4, a speed threshold of $40km/h$ and a direction threshold of $15°$ were used. These values were determined in series of experiments to yield the best results for the specific network case and tracking data. Figure 4 gives the pseudo-code of the *Intersection Detection* algorithm. Here, the direction and speed difference are considered in Lines 8 and 9, respectively.

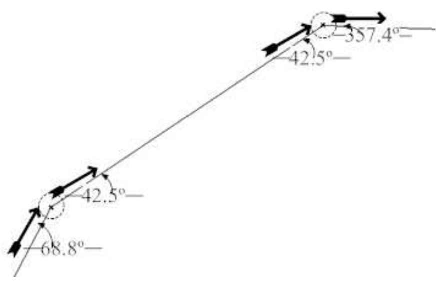


Figure 2: Angular difference

The Intersection Detection algorithm scans all trajectories in a position-by-position and an edge-by-edge manner taking into consideration the above conditions (Lines 6-10). We record all positions that satisfy the turn conditions (Line 11) and label them *turn samples.*

3.2.2 Clustering Turns

Categorizing turns by means of a *turn model* will enable us to cluster turn samples stemming from different trajectories and deriving intersections. The turn model describes all the possible movement patterns by using direction (0 is east, degrees increase counter-clockwise) in relation to an intersection. Using this approach, we essentially sample turns with respect their direction. While in this work we consider typical 4-way intersections, the approach can be extended to cover more complex intersections.

The turn samples are classified by using *eight types of turns*. The turn types captures all the possible combinations of incoming and outgoing links at a candidate intersection. Odd numbers are used for outgoing turns and even numbers for incoming turns resulting in four turn pairs as shown in Figure 3.

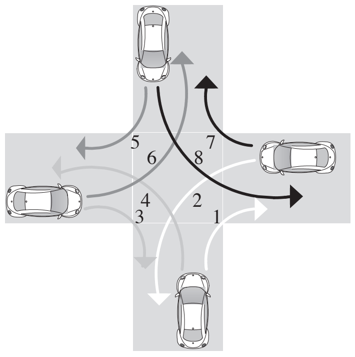


Figure 3: Turn model

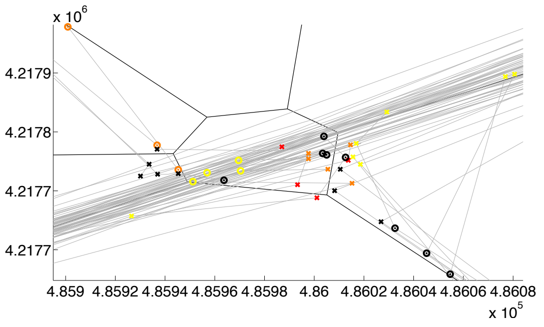
Using the turn model, the turn samples are grouped according to (i) spatial proximity and (ii) turn similarity.

Within each turn category now, we use an agglomerative hierarchical clustering method and a distance threshold of $50m$ (cf. Figure 4, Line 12) to identify *turn clusters*, i.e., turn samples clustered together based on location (candidate intersection location) and turn type. Again, this distance threshold was established through experimentation and yields only for the the specific network and tracking data case the best result.

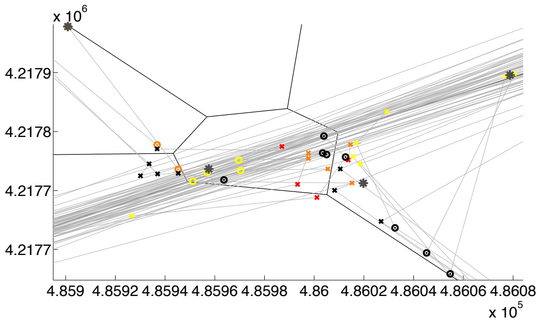
```
Intersections($T$)
     ▷ Clustering turns to compute intersections
 1   $P \leftarrow \emptyset$ ▷ Position sample in trajectory
 2   $P_S \leftarrow \emptyset$ ▷ Turn samples
 3   $P_C \leftarrow \emptyset$ ▷ Turn clusters
 4   $I \leftarrow \emptyset$ ▷ Intersection nodes
 5   $Angle$, $Speed$, $Dist$ ▷ Parameter thresholds
     ▷ Process all position samples in all trajectories
 6   while ($T[i] \neq$ null)
 7       $P \leftarrow T[i]$ ▷ Positions samples of a single trajectory
 8       $a_p \leftarrow$ AngularDiff($P[i-1], P[i], P[i+1]$)
 9       $v_p \leftarrow \frac{\delta x(P[i-1], P[i])}{\delta t(P[i-1], P[i])}$ ▷ Mean speed
10       if ($a_p \in Angle$ and $v_p \in Speed$)
11           $P_S$ .insert($P[i]$, TurnType($P[i]$))
     ▷ Cluster turn samples into turn clusters
12   $P_C \leftarrow$ ClusterTurns($P_S, Dist$)
     ▷ Cluster turn clusters into intersection nodes
13   $I \leftarrow$ ClusterIntersections($P_C, Dist$)
```

Figure 4: Finding intersections

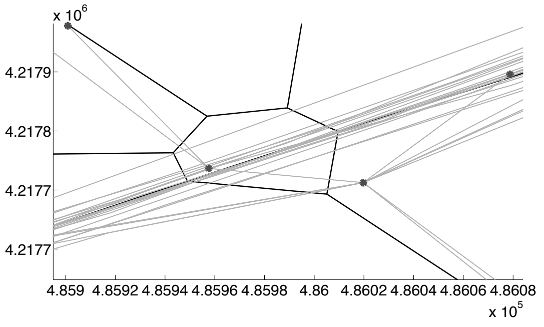
Figure 5(a) shows the calculated result for three roads that are met at an intersection. X and O markers are used for "odd" and "even" turn types, respectively. Using color, we further distinguish turn types. Yellow is used for types 1 and 2, orange for 3 and 4, red for 5 and 6, and black for 7 and 8.



(a) Position samples → turn clusters



(b) Turn clusters → intersection nodes



(c) Connecting intersection nodes

Figure 5: Computing intersection nodes

3.2.3 Intersections

Having identified turns, the question that remains to be answered is how we actually derive intersections. Again, performing agglomerative hierarchical clustering in connection with a distance threshold, (in our case $25m$), we translate the turn clusters established in the previous step, to intersection nodes (Figure 4, Line 13).

For each so generated intersection node, we also record two properties. A *weight* for the node is derived as the sum of the weights recorded for all constituting turn clusters, i.e., the total number of turns the intersection node was derived from. In addition, the *permitted maneuvers* for each node are recorded, i.e., given an intersection, what are the possible turns as "seen" by the GPS tracking data.

The distance threshold of $25m$ was established through experimental evaluation, i.e., it is lower than the threshold used for establishing turn clusters since the clusters' position is already located near a turn. Experimentation showed that a greater threshold would produce fewer intersections, as would a smaller threshold produce too many intersection nodes.

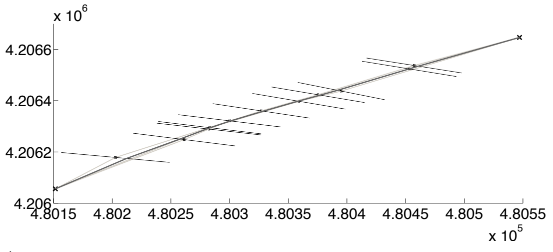
Essentially, we establish turn clusters based on distance and turn type, to then group them into intersection nodes.

Figure 5(b) shows intersection nodes using grey $*$ markers.

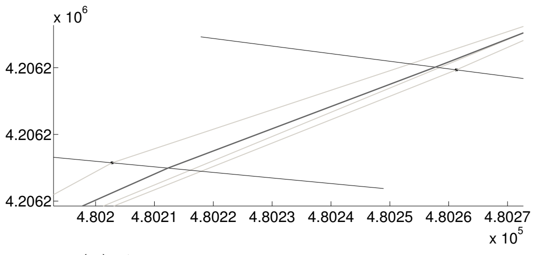
3.3 Connecting Intersection Nodes

At this stage in the network generation process, we succeeded in deriving isolated intersection nodes. In the following, we connect them, i.e., create links, by using the trajectory data. A fringe benefit of the intersection nodes computation based on turns is the connection of trajectory portions to these nodes, i.e., for all trajectories we know which samples helped constituting intersection nodes. To derive links we exploit this knowledge. We record for each intersection node the outgoing and/or incoming trajectory portions connecting this node to other nodes by essentially scanning all trajectories, whether they contain sequences of intersection nodes. The result of this step is the creation of a road network that connects nodes (intersections) with (trajectory portions) links.

The algorithm is simple, in that it essentially examines all trajectories based on whether they contain turn samples (with turn samples constituting intersections nodes) and "marking" the respective trajectory portions. In our data structure handling the trajectory data, all position samples that are also turn samples have been marked as such. Hence, performing a linear scan of all trajectories reveals the respective portions of the trajectories that connect turn samples, and, hence, intersection nodes (Figure 7, Lines 4-7).



(a) Computing link samples between two intersections



(b) Average positions of link samples

Figure 6: Computing intersection nodes

Essentially, two intersection nodes in question will typically be connected by a number of trajectories, i.e., vehicles that have passed more than once from an intersection to the other. In terms of network geometry, at this stage of the overall process, we introduce redundant links between intersection nodes as we simply want to identify which trajectory relates to which intersection node. Merging these links will be the next step. We refer to trajectory portions connecting intersections at this stage as *link samples*. To establish link samples, we merge the spatial portion of trajectories using a sweep-line algorithm (Figure 7, Lines 8-13). Given a set of trajectories, at each position sample we compute an average position based on the normal distance of the position sample to all other trajectories. Figure 6(a) shows a set of positions that comprise trajectory portions and the resulting link (grey) that was derived for connecting the two intersection nodes (black crosses). Horizontal lines indicate positions samples at which the average position is computed. Figure 6(b) shows a close up of the resulting link (grey) for the first two position samples shown at the left of Figure 6(a). In addition, for each link sample (i) a *weight* is derived representing the number of the trajectories comprising a link sample and (ii) a *width* is computed as the maximum spatial extent of the trajectories (thickness of a link sample is derived by the bundled trajectories). This link sample width will be used in the next step when compacting the road network as a size parameter of the buffer region that is used in the process.

Figure 5(c) shows how intersection nodes are connected by various trajectories. It also shows that trajectories that "pass through", i.e., do not turn at the intersection, have so far not been considered (but will be in the next section). In general, the number of generated intersection nodes depends highly on the parameter setting of the algorithm, i.e., choosing lower or higher threshold values will generate more or fewer intersection nodes, respectively. As also discussed in Section 5, the parameters need to be tuned to the network and the data in question!

```
LINKS($T$)
      $\triangleright$ Connecting intersection nodes using trajectories
 1    $I \leftarrow \emptyset$ $\triangleright$ Intersection nodes
 2    $IS \leftarrow \emptyset$ $\triangleright$ Intersection sequence
 3    $LS \leftarrow \emptyset$ $\triangleright$ Link samples
      $\triangleright$ Identify intersection sequences from trajectories
 4    for each $t \in T$
 5        for each $p \in t$
 6            if $p \in I$ $\triangleright$ was mapped to an intersection node
                  $\triangleright$ record prev., current intersection node & pts
 7                $IS \leftarrow \{i^-, i, p^-, p, t\}$
      $\triangleright$ Collect and merge link samples
 8    for each $\{i^-, i\} \in IS$
          $\triangleright$ add all trajectory portions for this intersection pair
 9        for each $t \in IS$
10            $LS \leftarrow \{t, p^-, \ldots, p\}$
          $\triangleright$ cluster link samples
11        $Width \leftarrow$ WIDTH($LS$)
12        $Weight \leftarrow$ WEIGHT($LS$)
13        $LS \leftarrow$ SWEEPMERGE($LS$)
```

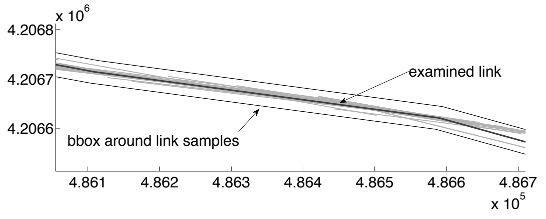
Figure 7: Connecting intersections

3.4 Compacting Links

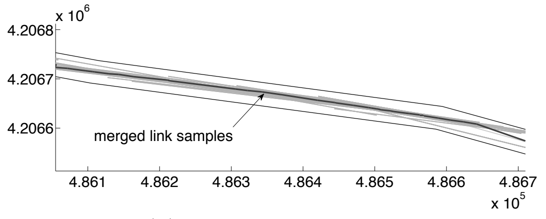
The state of the generated road network at this point is that we have intersection nodes connected by links derived from trajectories that exhibit turns at these intersections. This also means that a large portion of the data, trajectories "passing through" at intersections (bulk of the data shown in, e.g., Figure 5(c)), has not been considered yet with respect to all link samples. In a nutshell, the algorithm identifies trajectory portions that are close to existing links by means of a buffer region and merges their geometry onto the existing link geometry. In this step, we neither introduce new intersections nor do we add new links. We only adjust the geometry of existing links.

The three steps of the algorithm include (i) sorting existing link samples, (ii) using a buffer region around link samples to determine relevant trajectory portions, and (iii) adjusting the geometry of links based on the trajectories' geometry.

A first step is to sort all links according to their length (Figure 9, Line 1) so as to process longer links first as they are more significant for link construction. I.e., the longer a link, the more selective will be the match for a longer trajectory portion to fit in a buffer region. In trying to identify portions of link samples that match other link samples expressed by spatial proximity and direction similarity, the algorithm uses a buffer region around the *examined link sample* and retrieves all intersecting portions of other links (Figure 9, Line 6). The size of the buffer region is determined by the *width* of the respective link sample as described in Section 3.3. In addition to containment in a buffer region, a threshold is used to assess direction similarity. In our experimentation, we found that 45° are an adequate measure. Figure 8(a) shows in black the buffer region of an examined link. The examined link is shown in grey and respective portions of other candidate links are shown in light grey.



(a) Buffer region



(b) Merging link samples

Figure 8: Road Network Generation

As a pre-cursor to merging link samples, we record for each examined link its similar links and the portions that exhibit similarity. The latter is important in order to manage partially similar link samples. As the similar link samples can be located at the beginning, the end, or the middle, the remaining portions are preserved by splitting the respective links (Figure 9, Line 11).

Merging link samples follows an approach similar to the one of Section 3.3, when connections of intersection nodes were established. The method is applied to every portion of the examined link that exhibits partial similarity to other links (Figure 9, Lines 9 & 13). New links are created by interpolating link samples and introducing intersection nodes. In addition, new links preserve a weight that is the sum of the weights of the merged links.

Link samples are updated several times during this stage. While the examined links are reconstructed, new link samples are created and the existing are removed, i.e., additions, deletions and updates to the connectivity of the road network.

```
CompactLinks($L$)
     $\triangleright$ Compacting links to generate road network
 1   $LS \leftarrow$ sort($LS$, $length$) $\triangleright$ Sorting link samples by length
 2   $CLS \leftarrow \emptyset$ $\triangleright$ Candidate link samples
 3   $Width$ $\triangleright$ width of link samples
 4   $Angle$ $\triangleright$ direction threshold
 5   for each $l \in LS$
 6       $CLS \leftarrow$ Find(bbox($l$, $Width(l)$, $Angle$))
 7       for each $cl \in CLS$
 8           if Contains($l$, $cl$)
 9               SweepMerge($l$, $cl$)
10           else $\triangleright$ partial overlap
11               $cl_{in}, cl_{out} \leftarrow$ Split($l$, $cl$)
12               $CLS$.add($cl_{out}$)
13               SweepMerge($l$, $cl_{in}$)
```

Figure 9: Road network extraction algorithm

3.5 Post Processing

While the road extraction algorithm so far has already created a road network graph, the following heuristics-based post-processing step should further improve the quality of the road network. The basic idea in our road extraction process is the use of turns to identify turn clusters, which in turn create intersection nodes. The underlying trajectory data is recorded by means of taking position samples at regular time intervals. In the case of turns, this is especially critical in that a position sample might create turn clusters well in advance or after the actual turn and hence introduce additional intersections. We call this phenomenon *triangular intersections*.

To detect such triangular intersections, we analyze link sample weights in connection with geometric properties. To establish a criterion, we introduce the notion of relative weight ρ between the weights w_i, w_j of two link samples l_1, l_2 defined as $\rho_{i,j} = w_i/w_j$. Our aim is to detect such triangle constellations of links l_1, l_2, l_3 with two sides having respective high relative weights in relation to the third side, i.e., $\rho_{1,2} \gg \rho_{1,3} \wedge \rho_{1,2} \gg \rho_{2,3}$.

Figure 10 gives an example by showing in red link samples with high relative weight and in yellow link samples with low relative weight.

Following a statistical analysis, a link sample may be eliminated provided that both high relative weight ratios are > 0.7 and the low relative weight ratio is < 0.6, i.e., given $\rho_{i,j} > 0.7 \wedge \rho_{i,k} > 0.7 \wedge \rho_{j,k} < 0.6$, l_k can be eliminated.

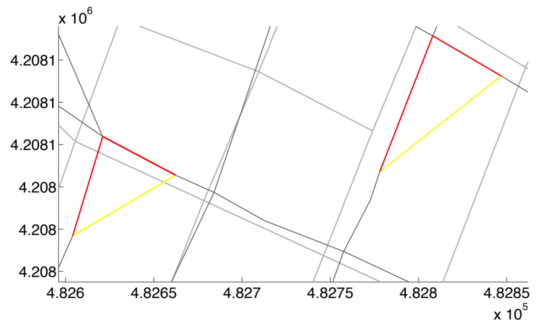


Figure 10: Triangular intersections

Figure 11 shows the distribution of such relative weight ratios (sorted by descending high relative weight ratio) for the 162 triangular intersections detected in the generated road network described in Section 5.

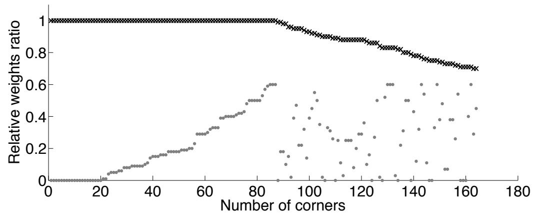


Figure 11: Relative weights comparison

4. EVALUATION

Essential for any automated process is the evaluation of its results. In the case of road map generation, this encompasses the assessment of the quality of the resulting road network. Ideally, we would like to compare the generated with the existing road network graph, i.e., how do the roads and the intersections we found by means of our algorithm line up with the actual road network. Provided that the tracking data does not cover all the road network, such a comparison should only be with respect to the corresponding portion of the network. To this effect, we defined a method that extracts a subset of a road network based on given tracking data samples.

To the best of our knowledge no algorithm/metric exists that compares two road networks, i.e., two metric graphs, we devised our own method that relies on massive shortest-path computation and comparing the resulting routes to reason about the similarity of the two road networks.

Thus, the quality of the generated road network is evaluated by a process that assesses the connectivity of the links and the geometry of the generated result. The evaluation process can be summarized in three steps. The *first step* determines a relevant portion of the actual road network that lines up with the tracking data. In the *second step*, we randomly produce distinct pairs of origin and destination nodes in the generated and the actual, partial road network and compute their respective shortest paths. Finally, in *step three*, we apply Distinct Frèchet distance and Average Vertical distance as quality measures to assess the similarity of the shortest-paths and, thus, reason about the similarity between the derived and the actual road network.

4.1 Road Network Extraction

The trajectory data covers a certain spatial area and we will use this extent to derive the covered portion of the road network. I.e., the reduced network will only comprise links of areas also covered by the tracking data. To find this partial network, we augment the geometry of the underlying road network with buffer regions around the network edges. Experimentation showed that a distance threshold of $50m$ is not too small to exclude road portions and not too large to include all the road network. Now, to derive the partial network, we use the trajectories as a query set with the condition that either they cross or they are spatially contained in the buffer regions representing the links of the actual road network.

In the example of Figure 12, the dashed black lines represent the buffer region, the grey lines the partial road network and the yellow lines an instance of the trajectory data. As this process is by no means perfect, in this example, several links have been falsely identified (redundant links).

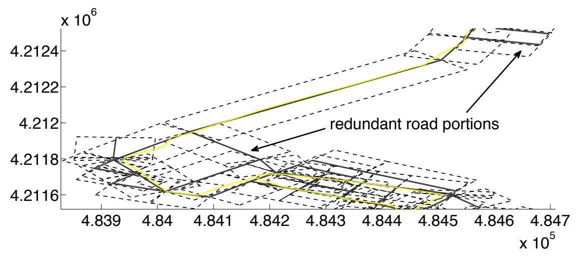


Figure 12: The underlying road network

4.2 Shortest Paths and Network Similarity

To assess the quality of the generated road network, we will use a large number of shortest-path queries in the process. We compute shortest-paths for randomly selected pairs of origin and destination nodes in (i) the generated and (ii) the actual road network. Using Dijkstra's algorithm, besides the actual path, we also record the total cost of the path in terms of distance and the number of links.

We compare the similarity of the shortest paths by using (i) the Discrete Frèchet distance and (ii) the Average Vertical distance to the corresponding pairs of shortest paths. The similarity measures are not applied to individual links, but to the entire paths to able to draw conclusions regarding more extensive portions of the road network. To compute these distance measures, readily available MATLAB routines are used.

The results of this evaluation can be found in Section 5.3.

5. EXPERIMENTATION

Having devised an algorithm to derive road networks from collected trajectories, this section will showcase various results with a focus on the quality of the generated road network data.

5.1 Experimental Setup

All algorithms have been developed in MATLAB given its impressive high-level routines for statistics including clustering and visualization, essentially allowing us to focus on core algorithms and data structures. The experimental setup in

Finding intersections	
Angular difference	$15°$
Mean speed	$40km/h$
Max sampling interval	$35s$
Turn clustering threshold	$50m$
Intersection clustering threshold	$25m$
Road network generation	
Direction threshold	$45°$
Evaluation	
Road network extraction threshold	$50m$

Table 1: Parameters overview

Figure 13 shows the methods developed in order to automatically derive and assess the generated road network.

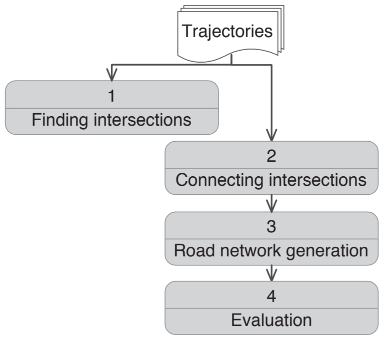


Figure 13: Experimental setup

A series of initial experiments established the proper parameter setting for our road generation algorithm given the specific road network and also tracking data (cf. Table 1). Turn clustering and intersection extraction (cf. Section 3.2) employs four parameters. For turn clustering, (i) the angular difference threshold was set to $15°$. We found that when move straight ahead, vehicles do not change direction by more than $15°$. If they do and in combination with (ii) the mean speed threshold of $40km/h$ that vehicles may experience while turning, then we consider this as an indication for a turn. Both measures have been chosen very conservatively as false positives will eventually be eliminated and they should not be backed up by additional samples. In addition, (iii) a maximum time constraint of $35s$ and a distance threshold of $50m$ were used to cluster position samples. With respect to clustering turn samples into intersection nodes, (iv) a $25m$ distance threshold was established by analyzing the result data. The road network generation algorithm (cf. Section 3.4) uses a buffer region that encloses candidate road network portions. While its size is dynamically established, we initialize it with a maximum width of $20m$.

All parameters were established empirically by running a great number of experiments and assessing the quality of the respective results. While details of these results are omitted in this paper, the above parameters represent the best setting for the specific case (network and tracking data).

5.2 Road Network Generation Results

In what follows, we describe the results of the road network generation algorithm applied to trajectory data covering a portion of the road network of Athens, Greece.

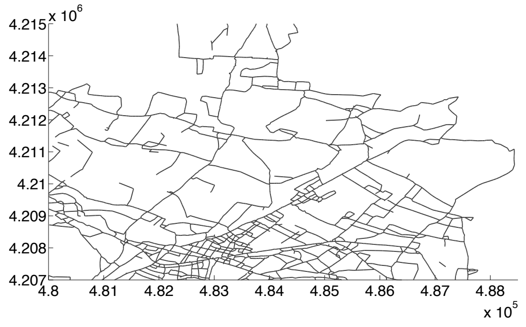
The actual road network portion that we try to generate consists of 22490 links (edges) and 15389 nodes. It covers an area of $13km \times 16km$. The edges have a length of $1813km$. However, the tracking data used for our purpose covers only a portion of this road network. The road network is illustrated in Figure 14(c).

In the experiments, we used vehicle tracking data that was recorded using GPS at a typical sampling interval of $30s$. As the expected sampling rate in our data is $30s$, we consider time intervals of $35s$ in between position samples as a discontinuity in the data and introduce in such a case a new trajectory. Such temporal gaps in the data are typically due to a GPS signal loss or a turned-off GPS receiver. The data comprises 120 vehicle trajectories with a total length of $6070km$ (Figure 1). Following the various stages of the road network extraction algorithm, the algorithm produces the following output. During the first phase, i.e., intersection extraction and connection, 4995 intersection nodes and 5983 link samples are generated. All links combined have a length of $2700km$. The second road network generation phase produces 5124 intersection nodes, 6219 links and a length of $710km$. This result shows that during the second phase of the algorithm, the number nodes remains largely constant but only the length of the links connecting them is significantly reduced since we radically merge links during this phase.

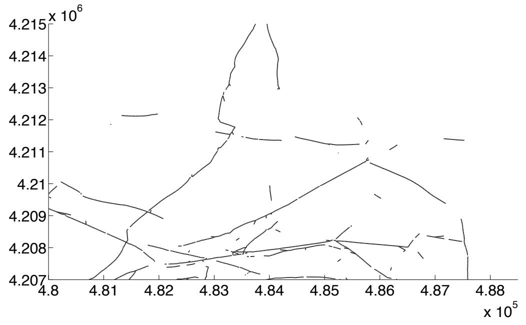
The overall time to compute these results in MATLAB (Windows 7) on an Intel Core2Duo processor running at 2.2GHz was $56min$. This time was achieved without performing any optimization on the data structures used in the implementation.

Figure 14(a) visualizes the generated road network. In addition, Figure 14(b) illustrates how the computed link weight information can be used to identify major roads of the network. In this visualization, links are shown that are traversed at least 10 times. The gaps in the road network are due to uneven distribution of weights during stage two of the algorithm when compacting the road network and the untypical traversal of roads by school buses, i.e., turning off major roads to drop of kids.

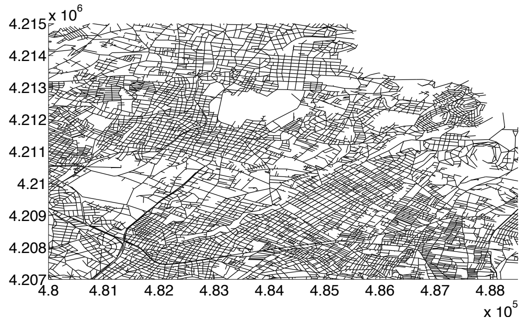
Zooming in on the data, Figure 15 shows the tracking data, the generated road network, and the actual road network of a smaller area. At this scale, it can be clearly seen that the traversed portion of the road network was correctly identified.



(a) Generated road network



(b) Heavily travelled road network



(c) Original road network

Figure 14: Experimental results

5.3 Quality Assessment

As part of the performance study, we evaluate the quality of the generated road network in terms of road network accuracy and connectivity.

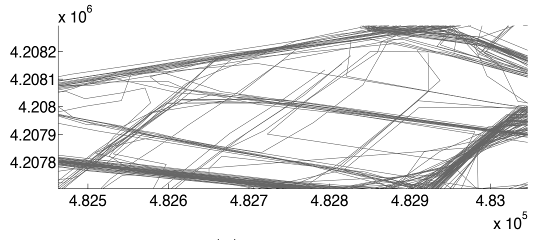
In our case, the quality of the generated result is expressed by the distance measure between computed shortest-paths in the generated and the actual (partial) road network. To this effect, we computed 500 shortest-paths uniformly distributed over the two networks. Figure 16(a) displays the network coverage of the randomly created routes (excerpt).

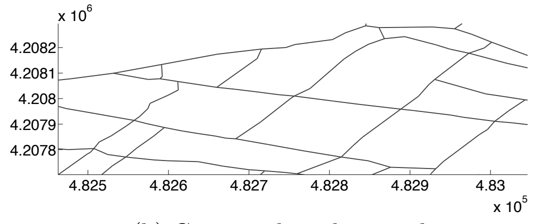
Figure 16(b) gives an example of how similar two randomly created routes are in the actual (light grey) and in the generated (dark grey) road networks.

Besides visual inspection, we needed to come up with an indicator for dissimilarity and a way to assess the quality of the generated network. Dissimilarity can be expressed as an increasing distance between the two paths. Figure 17 shows the Discrete Frèchet distance (in light grey) and Average Vertical distance (in dark grey) of the 500 computed paths. Combining visual inspection with path distance, it was empirically established that a Discrete Frèchet distance greater than $300m$ is an indicator of such dissimilarity. In other words, such distances are an implicit indication for dissimilarity between the generated and the actual road network.

Shortest-path pairs exceeding this distance use portions of the generated road network with connectivity problems that do not exist in the actual road network. Connectivity problems are mostly due to falsely created links. In our experimentation, we found that 5% of the computed routes appear to have such problems, i.e., in 95% of the cases, the computed shortest paths in both networks were almost identical exhibiting only small differences at the origin and the destination nodes accounting for distance measures up to $300m$.

Having a closer look at some of the 5% cases, we can ob-


(a) Raw data


(b) Generated road network

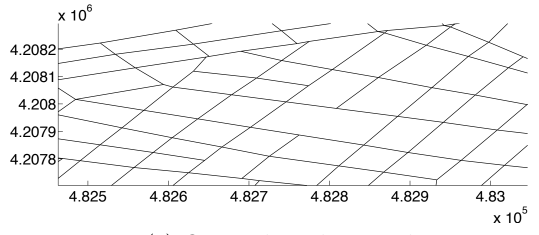
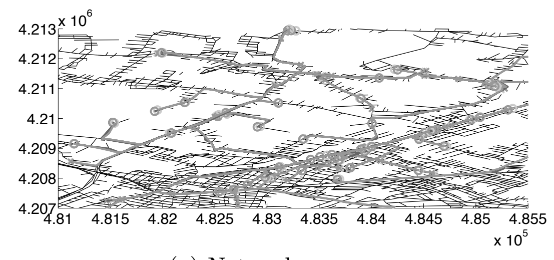

(c) Original road network

Figure 15: Generated road network (close ups)


(a) Network coverage

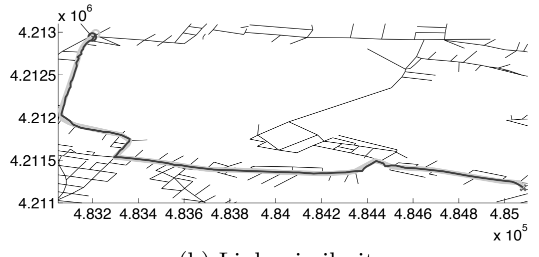

(b) Links similarity

Figure 16: Assessing generated road network

serve two problems for the generated network; (i) spatial accuracy and (ii) connectivity. Figure 18(a) shows an example in which due to the spatial accuracy, an alternative route was computed. In this case, the generated network simple produced a slightly different geometry that made the

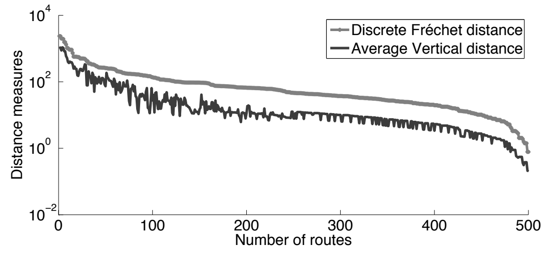
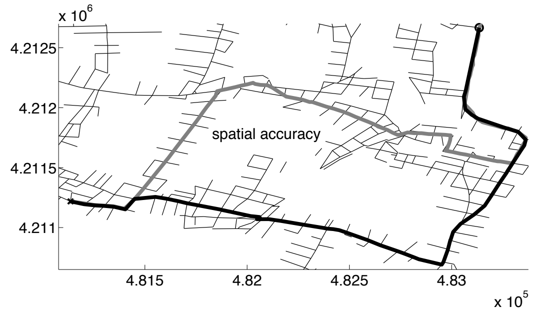
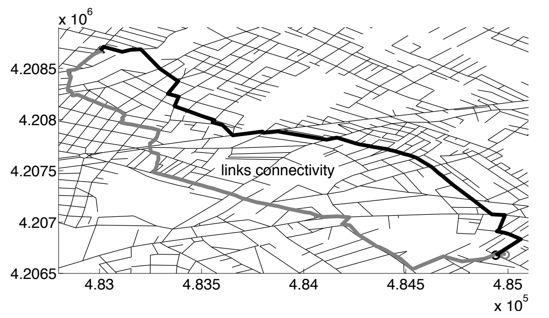


Figure 17: Similarity results

shortest-path algorithm choose a partially different route. A more serious problem is that of connectivity. Figure 18(b) illustrates this problem, which results in different routes due to missing links. In both visualizations, the route of the actual road network is shown in black, while the route of the generated road network is grey.



(a) Spatial accuracy



(b) Links connectivity

Figure 18: Link failures

5.4 Summary

Our experiments have shown that the proposed algorithm taking vehicle trajectories as input produces navigable road networks that when compared to the actual road network dataset exhibit great similarity. Assessing the generated road network using sets of shortest-path queries, we found that 95% of paths computed using the generated road network exhibit great similarity to the comparable paths in the actual road network. 5% show dissimilarity due to different connectivity and other spatial aspects such as link length. Visual inspection shows that the generated road network closely resembles the actual road network if sufficient tracking data that provides redundant coverage of the road network is available.

6. CONCLUSIONS

This work describes a novel approach to road network generation from GPS tracking data. In an nutshell, the algorithm exploits changes in movement patterns by using turns as a means to identify intersection nodes. Intersection nodes are effectively used to bundle vehicle trajectories and links between intersections are derived by merging them. In addition, we developed a method to assess the quality of the generated road network based on comparing computed shortest-paths by means of distance measures. Overall, our algorithm produces road networks that closely resemble the actual road network provided sufficient tracking data is available. Redundancy in coverage increases the quality of the road network.

Our directions for future work include the testing of the method using additional tracking data sources with different sampling rates as well as geographic locations, as the algorithm has several parameters that need to be tuned to a specific setting. Here, we will focus first on creative commons data such as Openstreetmap [1] and other sources as they become available. Assessing the quality of the generated road network is important and our aim is to improve our method towards directly considering topological network properties. If possible, such a method should lead to a tool for benchmarking network generation algorithms in general. One motivation for automatic road network generation from tracking data are continuously evolving road networks and the related challenging maintenance. Here our contribution will be to investigate algorithms that work for incomplete, uncertain and continuously evolving collections of data, i.e., refining an existing network with newly collected data. During experimentation we observed that the properties of the tracking data such as sampling rate and vehicle speed highly affect the produced result. We will investigate techniques to segment the tracking data, to use automatically determined parameters and to generate different aspects of the road network (hierarchies) independently.

Acknowledgments

The research leading to these results has received funding from the European Union Seventh Framework Programme - Marie Curie Actions, Initial Training Network GEOCROWD (http://www.geocrowd.eu) under grant agreement No. FP7-PEOPLE-2010-ITN-264994.

7. REFERENCES

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

[2] M. Aanjaneya, F. Chazal, D. Chen, M. Glisse, L. J. Guibas, and D. Morozov. Metric graph reconstruction from noisy data. In *Proc. 27th SoCG*, pages 37–46, 2011.

[3] M. Ahmed and C. Wenk. Constructing street networks from gps trajectories. In *Proc. 20th ESA conf.*, 2012.

[4] J. K. Alireza Fathi. Inferring the road network from gps data. In *Geographic Information Science*, volume 6292, pages 56 – 69, 2010.

[5] A. Baumgartner, S. Hinz, and C. Wiedemann. Efficient methods and interfaces for road tracking. In *International Archives of Photogrammetry and Remote Sensing*, page B:28, 2002.

[6] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk. On map-matching vehicle tracking data. In *Proc. 31st VLDB conf.*, pages 853–864, 2005.

[7] R. Bruntrup, S. Edelkamp, S. Jabbar, and B. Scholz. Incremental map generation with gps traces. In *Proc. IEEE ITS conf.*, pages 574–579, 2005.

[8] L. Cao and J. Krumm. From gps traces to a routable road map. In *Proc. 17th ACM GIS conf.*, pages 3–12, 2009.

[9] D. Chen, L. J. Guibas, J. Hershberger, and J. Sun. Road network reconstruction for organizing paths. In *Proc. 21st ACM-SIAM Symp. on Discrete Algorithms*, pages 1309–1320, 2010.

[10] Y. Chen and J. Krumm. Probabilistic modeling of traffic lanes from gps traces. In *Proc. 18th ACM GIS conf.*, pages 81–88, 2010.

[11] S. Edelkamp and S. Schrödl. Computer science in perspective. chapter Route planning and map inference with global positioning traces, pages 128–151. Springer-Verlag New York, Inc., New York, NY, USA, 2003.

[12] T. Guo, K. Iwamura, and M. Koga. Towards high accuracy road maps generation from massive gps traces data. In *Proc. IEEE IGARSS conf.*, pages 667–670, 2007.

[13] J. Hu, A. Razdan, J. Femiani, M. Cui, and P. Wonka. Road network extraction and intersection detection from aerial images by tracking road footprints. *IEEE T. Geoscience and Remote Sensing*, 45(12-2):4144–4157, 2007.

[14] J.-G. Lee, J. Han, and K.-Y. Whang. Trajectory clustering: a partition-and-group framework. In *Proc. SIGMOD conf.*, pages 593–604, 2007.

[15] Y. Liu. An automation system: generation of digital map data from pictorial map resources. *Pattern Recognition*, 35(9):1973–1987, 2002.

[16] S. Morris and K. Barnard. Finding trails. In *Proc. IEEE CVPR conf.*, 2008.

[17] S. Schroedl, K. Wagstaff, S. Rogers, P. Langley, and C. Wilson. Mining gps traces for map refinement. *Data Min. Knowl. Discov.*, 9:59–87, July 2004.

[18] M. Tavakoli and A. Rosenfeld. Building and road extraction from aerial photographs. *IEEE Transactions on Systems, Man and Cybernetics*, 12(1):84–91, 1982.

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

[20] H. Zhao, J. Kumagai, M. Nakagawa, and R. Shibasaki. Semi automatic road extraction from high resolution satellite image. In *Proc. Photogrammetric Computer Vision ISPRS Commission III Symp.*, pages 406–411, 2002.