A Layered Approach for More Robust Generation of Road Network Maps from Vehicle Tracking Data

SOPHIA KARAGIORGOU, Institute for the Management of Information Systems, R.C. Athena
DIETER PFOSER, Department of Geography and GeoInformation Science, George Mason University
DIMITRIOS SKOUTAS, Institute for the Management of Information Systems, R.C. Athena

Nowadays, large amounts of tracking data are generated via GPS enabled devices and other advanced tracking technologies. These constitute a rich source for inferring the structure of transportation networks. In this work, we present a novel methodology for revealing a road network map from vehicle trajectories. Specifically, we propose an enhanced and robust map construction algorithm which is based on segmenting the original tracking data according to different types of movement and then constructing the topology of the road network hierarchically. The segmentation produces separate road network layers, which are then fused into a single network. This provides a more efficient way to addresses the challenges imposed by noisy and low sampling rate trajectories. It also allows for a mechanism to accommodate automatic map maintenance on updates. Thus, the proposed approach overcomes the limitations of existing methods and introduces a map construction algorithm which is robust against heterogeneous and sparse data and capable to incorporate changes and improvements. An experimental evaluation extensively assesses the quality of the proposed methodology by constructing large parts of the road networks of four major cities, namely Athens, Berlin, Vienna and Chicago, using as input GPS tracking data of utility vehicles and taxi fleets. Our results show significant improvements concerning the spatial accuracy and the quality of the constructed road network over the current state of the art.

General Terms: Geoprocessing and Map Production Algorithms

Additional Key Words and Phrases: Map construction, trajectories, road networks

ACM Reference Format:
Sophia Karagiorgou, Dieter Pfoser, and Dimitrios Skoutas, 2017. A Layered Approach for More Robust Generation of Road Network Maps from Vehicle Tracking Data. *ACM TSAS* 3, 1, Article XX (April 2017), 21 pages.
DOI: http://dx.doi.org/10.1145/0000000.0000000

1. INTRODUCTION

During the last years, the widespread adoption and use of GPS enabled devices, in conjunction with the increasingly popular phenomenon of crowdsourcing, has opened up new opportunities for tracking the movement of various types of entities, including vehicles, humans and animals. This has enabled a wide spectrum of novel applications and services. Among them is the process of using the traces of moving objects to produce a map of a transportation network.

This problem has two broad categories of application scenarios. The first scenario involves cases where the entities move along specific trails which are not already mapped. Such examples include the movements of hikers or animals. It also involves

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.

DOI: http://dx.doi.org/10.1145/0000000.0000000

(a) Vehicle trajectories (b) Road network

Fig. 1. Vehicle trajectories and the OpenStreetMap road network for the corresponding part of Berlin.

cases where a map exists but is not publicly available and may be too costly to acquire. The goal in these cases is to track the movements of the entities and use the extracted trajectories to infer the map of a movement network. The second scenario involves cases where the map of a network exists but needs to be maintained and updated, or enhanced with additional properties. For example, maps of road networks are traditionally created through the use of aerial imagery [Tavakoli and Rosenfeld 1982], a method which is not suitable for keeping up with road changes or determining dynamic aspects, such as traffic controls, turn restrictions, blockages due to natural phenomena or accidents.

In this paper, we focus on the scenario of inferring a road network from sparse trajectories coming from vehicle tracking data. The inherent inaccuracies and errors of the collected tracking data (quality of GPS signal and transmission errors) make the map construction problem very challenging. Such an example is illustrated in Figure 1, which plots the collected trajectories from vehicle tracking data in Berlin (1a) and the actual map of the corresponding road network (1b) from OpenStreetMap. Clearly, inferring the latter from the former is not a trivial task.

Although recently several road network inference methods have been proposed, they typically rely on uniformly distributed, frequently sampled and low-noise GPS traces, which limits their applicability and effectiveness in many real-world scenarios. In a previous work [Karagiorgou and Pfoser 2012], we presented an automatic road network construction algorithm that takes vehicle tracking data in the form of trajectories as input and produces a road network graph. The method detects changes in the direction of movement to infer intersection nodes, and then "bundles" the trajectories around them to create the network edges. Even though it is relatively robust with respect to noisy GPS traces and different sampling rates, it relies on several parameters which cannot be flexibly adapted when facing a large network where different parts may have different characteristics, i.e. road categories, speed profiles, etc. However, the latter is a standard characteristic of large urban road networks, which exhibit a hierarchical nature and differing densities, comprising streets that range from wide highways and avenues to local and narrow roads.

To address the challenges of map construction from noisy, low-frequency sampled tracking data, in the case of large urban road networks with parts of different characteristics, we perform a layered construction of the network map. The presented approach exploits vehicle trajectories to analyze, segment and reconstruct the underlying movement network in a layered form. This layered approach allows to segment the input dataset into groups of trajectories based on their characteristics, and then to process each group accordingly. Moreover, it makes it possible to deal with changes and incorporate updates in an incremental fashion. Through an experimental and com-

parative evaluation, we show that this method is robust and provides more accurate results when dealing with noisy and heterogeneous vehicle tracking data and road networks.

In particular, the main contributions of our work are as follows:

(i) We introduce a map construction algorithm that segments the original vehicle tracking data into different groups of trajectories based on their characteristics and then constructs the road network in a layered fashion. This cumulative approach enables to combine new road segments or road closures due to incidents and updates.
(ii) During the above process, we introduce a proximity-based expansion algorithm around turn samples based on turn similarity, which allows to create intersection nodes based on the available data by using sets of trajectories that belong to the same speed category.
(iii) We present a detailed experimental and comparative evaluation of our method, using several real-world vehicle tracking datasets, which shows that the proposed method outperforms the current state of the art. The algorithms we compare constitute representatives of different map construction algorithm classes.

The remainder of this paper is organized as follows. Section 2 reviews related work on road network construction techniques. Section 3 presents our algorithms for trajectories segmentation and re-association to build a hierarchical network map by incorporating updates. In Section 4, we evaluate the quality of the layered map construction method, comparing our results to other state of the art approaches. Finally, Section 5 concludes the paper and outlines future research directions.

2. RELATED WORK

Various approaches have been proposed for using GPS vehicle tracking data to construct road network maps or to update, refine and enhance existing ones with additional properties. In the following, we present a review of the literature by using a categorization of the methods according to the type of the algorithms used.

Several methods rely on *k-means clustering* of raw GPS data using distance and direction as criteria to introduce cluster seeds at fixed locations along a vehicle trajectory. Edelkamp and Schroedl [Edelkamp and Schroedl 2003] use various heuristics for road segmentation, map matching, and lane clustering from GPS traces. Schroedl et al. [Schroedl et al. 2004] use *k-means clustering* in order to refine an existing road network map rather than construct the entire road network starting from a blank terrain. Guo et al. [Guo et al. 2007] use statistical analysis of GPS traces to generate road maps. However, the data model assumption with respect to the distribution of GPS data follows a 2D Gaussian symmetry which is unrealistic, especially in error-prone environments. Worrall and Nebot [Worrall and Nebot 2007] emphasize on GPS traces compression to infer a digitized road map of a small scale and their experiments are limited to small datasets. Similarly, Jang et al. [Jang et al. 2010] propose a system for map creation from as few as ten traces and their results are presented in a very small scale, without providing any description regarding the data characteristics, such as the sampling rate and the GPS error. Finally, Agamennoni et al. [Agamennoni et al. 2011] present a machine learning method to consistently build a representation of the road network mostly in dynamic environments such as open-pit mines.

Other methods transform GPS traces to density-based discretized images and are based on *Kernel Density Estimation* (KDE). Most of these algorithms function well either when the data are frequently sampled (e.g., once per second) [Chen and Cheng 2008] or when there is a lot of data redundancy [Shi et al. 2009]. Biagioni and Eriksson [Biagioni and Eriksson 2012] use a dataset which is being sampled very frequently

(from 2 to 6 seconds), while Davies et al. [Davies et al. 2006] use GPS samples which are obtained every 1 second. Steiner and Leonhardt [Steiner and Leonhardt 2011] present an approach which uses vehicle tracking data of lower frequency, but still with intervals not exceeding 15 seconds. The limitation of KDE based algorithms is that they are quite sensitive with respect to noisy data and outliers.

Several methods address the map construction problem from a theoretical perspective based on *computational geometry* techniques, providing also quality guarantees. Chen et al. [Chen et al. 2010] focus on detecting seed elements in the road network and then connecting them accordingly. However, they impose strict assumptions with respect to the GPS sample coverage, the error bounds and the road geometry. Aanjaneya et al. [Aanjaneya et al. 2011] 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. Ahmed and Wenk [Ahmed and Wenk 2012] developed an incremental method that employs the Fréchet distance to match partial trajectories to a graph. While they give quality guarantees, their approach does not address the basic connectivity problem and how to measure the respective quality of a connected network graph. Ge et al. [Ge et al. 2011] cluster the input points using local neighborhood properties to extract the structure from unorganized high-dimensional data such as point clouds or proximity graphs. A common problem with most approaches is that they rely on high-quality GPS traces, i.e., high sampling rate and low positional errors of up to 5 meters, and provide theoretical quality guarantees for the constructed output map, under certain assumptions on the underlying street map and the input tracks. Instead, in this approach, we use trajectory data of low sampling rate (from 15 to 90 seconds). The quality of the constructed map improves with the amount of available data in terms of redundancy rather than the data quality itself.

Another category, to which the present work most closely relates, involves *trace clustering* approaches. These methods either adopt map matching [Quddusa et al. 2007] or heuristics by aggregating GPS traces into an incrementally built road network [Niehofer et al. 2009]. Vehicle heading and distance measures are also used to perform additions and deletions onto the incremental construction of the map. Rogers et al. [Rogers et al. 1999] use trace clustering merely to refine an existing road network rather than extracting it from scratch. Cao and Krumm [Cao and Krumm 2009] eliminate noise in GPS traces, while Fathi and Krumm [Fathi and Krumm 2010a], [Fathi and Krumm 2010b] provide an approach that discovers intersections by using a prototypical detector trained on ground truth data from an existing map. This approach works best for well-aligned road networks and with frequently sampled data of up to 5 seconds. Bruntrup et al. [Brüntrup et al. 2005] and Liu et al. [Liu et al. 2012] efficiently build a road network, but require accurate data and high sampling rates (i.e. 1 second). Zhang et al. [Zhang et al. 2010] use a method similar to GPS trace clustering to continuously refine existing road maps.

In general, although the problems of map construction, update and enhancement are complementary, typically each individual work focuses on a single one of them. For example, a recent work by Wang et al. [Wang et al. 2013] applies trace clustering techniques to introduce a new KDE based road fitting algorithm. The authors achieve an important contribution in terms of map data entries on the OpenStreetMap collection, but their application mainly focuses on updating a map rather than constructing it. Similarly, Shan et al. [Shan et al. 2015] extend over [Wang et al. 2013] by proposing an automatic map update system which focuses on the identification of missing road segments and is robust w.r.t. low sampling rates (on average of 120 seconds). Wang et al. [Wang et al. 2015] efficiently tackle the hard time performance of current

approaches, deal with tracking data of low sampling rate but they mainly focus on inferring a map attributed with topological characteristics.

Compared to the aforementioned approaches, the proposed method differs in that it preserves the underlying connectivity of the road network embedded in the vehicle trajectories meaning that we deliver a road map which constitutes connected components of a graph. It also extracts useful knowledge w.r.t. the underlying road network characteristics. These include the road segment categories exploiting the speed profiles, the bi-directional attributes and properties regarding the enabled manoeuvre near turns which can be further taken into account in a shortest path computation. Trajectories are hierarchically clustered together based on intersection indicators (turn samples) and speed profiles. This work introduces an *improved intersection detection algorithm*, as well as novel concepts, namely the *segmentation of the input trajectories according to speed profiles*, the *construction of separate layers of the network*, and their *conflation into a final, fused road network graph*. A fringe benefit of the segmentation and conflation approach is that in this way, we can support network updates, i.e. incorporating new segments into an existing road network.

3. INFERENCE AND FUSION OF NETWORK LAYERS

3.1. Overview

In this section, we present our proposed method for map construction, called TRACECONFLATION. The main underlying idea is to more effectively address the heterogeneous parts of large urban road networks that comprise streets with different characteristics, ranging from wide, high-speed highways to narrow, low-speed residential roads. To that end, the road network is constructed in a layered fashion, based on segmenting the input trajectory data according to different speed categories.

More specifically, the process involves three steps:

(i) *Segmentation of trajectories*: the input dataset of trajectories is split into subsets of (sub-)trajectories according to their characteristics;
(ii) *Construction of network layers*: each subset is processed separately to identify nodes and edges of the network, resulting in a partial network layer;
(iii) *Conflation of network layers*: the separate, partial layers are assembled to produce the complete map of the road network.

The input to the map construction process comprises vehicle tracking data (a.k.a. floating car data – FCD), which form a set of trajectories. Such data can be collected by a fleet of different types of vehicles, e.g., taxis, public transport, utility vehicles, private vehicles, and they are often used in applications such as traffic monitoring and fleet management.

Given these raw traces, we use linear interpolation between consecutive positions to derive the trajectory of a vehicle. A trajectory is modeled as a list of spatio-temporal points $T = \{p_0, \ldots, p_n\}$ with $p_i = \langle x_i, y_i, t_i \rangle$ and $x_i, y_i \in R, t_i \in R^+$ for $i = 0, 1, \ldots, n$ and $t_0 < t_1 < t_2 < \ldots < t_n$. These trajectories are susceptible to noise, as they are affected by measurement errors, due to the limited GPS accuracy, as well as variations in the sampling rate. These greatly impact the quality of the dataset and, subsequently, the quality of the results extracted by it [Brakatsoulas et al. 2005].

The output of the map construction process is a road network, modeled as a directed graph $G = (V, E)$, where the vertices V correspond to intersection nodes or other breakpoints in streets, and the edges E correspond to links, i.e. road segments, between vertices.

In the following, we describe the steps of the TRACECONFLATION method in detail.

ALGORITHM 1: Segmentation of Trajectories

Input: A set of trajectories $\mathcal{T}$; a set of speed categories $\mathcal{C}$
Output: A set of categorized trajectory segments according to the speed categories

```
1  begin
2      /* Trajectories segmentation according to speed profiles */
3      T_segm ← ∅
4      foreach (T ∈ T) do
5          foreach (L_j ∈ T) do
6              v̄(L_j) ← MEDIAN(v(L_{j−w}), ..., v(L_{j+w}))
7              C ← C ∈ C where v̄(L_j) ∈ C
8              T_segm ← T_segm ∪ {(L_j, C)}
9          end
10     end
11     return T_segm
12 end
```

3.2. Segmentation of Trajectories

As already mentioned above, a road network may comprise heterogeneous parts. Moreover, the raw GPS traces used as input for the map construction process are often noisy and sparse, due to GPS errors, missing values, different sampling rates, different speeds, etc. Thus, treating all the input data equally, inevitably introduces inaccuracies in the extracted results.

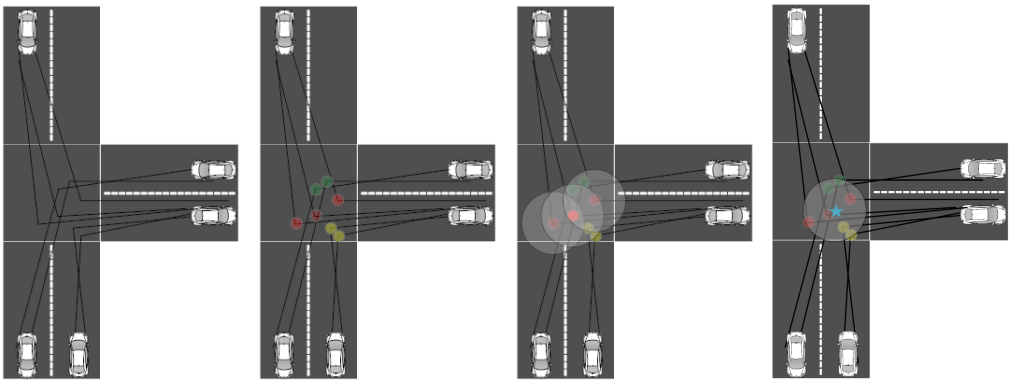
To deal with this problem, we analyze the trajectories in the input data and split them into subsets with different characteristics, in particular according to the speed of the moving object. This allows us to treat each subset separately, e.g., by setting the parameters of our algorithm accordingly. The aim is to derive different but overlapping portions of the network with higher accuracy, which then can be merged to produce the complete network. Hence, this process leads to a layered construction of the network.

We consider different speed categories, e.g., "slow", "medium", "fast", and we then split and classify trajectories accordingly. Typically an object may have moved with different speeds across different parts of the trajectory. In this case, the trajectory is first split into sub-trajectories, with each one being assigned to the corresponding category. A naive process for achieving this is the following. First, a speed value is assigned to each line segment of the trajectory. This value is computed by dividing the length of the segment by the duration of the time interval of its start and end points. Then, each segment is assigned to the corresponding speed category. However, due to the nature of a vehicle's movement, this often leads to a high degree of fragmentation, rendering the dataset unusable. For instance, when a vehicle moves, it may often slow down, due to an intersection, a traffic light or other kinds of obstacles. To avoid excessive fragmentation of trajectories due to such abrupt and short changes in speed, 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. Then, splitting and categorization of sub-trajectories is done according to these "smoothened" speed values.

The process is outlined in ALGORITHM 1. For each line segment L_j of each trajectory T, its median speed is computed over a sliding window of width $2 \cdot w$ (Line 6). The segment is then assigned to the corresponding class according to the minimum and maximum speed constituting each category (Lines 7 - 8).

3.3. Construction of Network Layers

The next step is to use the split and categorized trajectory segments to infer each corresponding layer of the road network. This is done by using an improved version of the road network construction algorithm we had previously introduced in [Karagiorgou



(a) Trajectories (b) Turns (c) Neighborhoods (d) Intersection

Fig. 2. Detection of intersection nodes.

and Pfoser 2012], called TRACEBUNDLE. In the following, we first present briefly the main aspects of the TRACEBUNDLE algorithm, and then we describe the improvements introduced here.

3.3.1. The TRACEBUNDLE *Algorithm.* This is a trace clustering map construction algorithm. It employs heuristics to identify intersection nodes, and then "bundles" trajectories around them. The basic part of TRACEBUNDLE is how it infers intersection nodes. This relies on detecting changes in the vehicle's movement, and then applying a clustering method. Such changes represent turns and are identified as changes in direction and speed. Each detected turn is represented by a single point on the map. Subsequently, *turn clusters* are formed by clustering these turns based on (i) spatial proximity and (ii) turn type (i.e., direction). The centroid location of each computed turn cluster determines an *intersection node* in the constructed road network. Finally, links are derived by associating the initial trajectories to the inferred intersection nodes, and compacting them, i.e., "bundling" them together. Thus, the entire geometry of the road network is derived.

3.3.2. Improved Node Detection. As explained above, the TRACEBUNDLE algorithm identifies intersection nodes by clustering turn samples. The clustering is based on two criteria, *proximity* and *angle difference*, which are controlled by corresponding parameters. Since TRACEBUNDLE does not distinguish between different network layers, and hence does not discriminate between different types of streets in the road network, the same values for these parameters are used across the whole network. Consequently, this clustering configuration may fit well to some parts of the network but less so in others, resulting in clusters of poor quality. For instance, for some streets, the values of these parameters turn out to be stricter than required, resulting in generating multiple nodes where in fact only a single intersection occurs; on the other hand, in other occasions, the same values turn out to be more relaxed than needed, resulting in "bundling" together links belonging to different intersections into a single intersection node.

To overcome this problem, in TRACECONFLATION we introduce a more flexible and robust method for node detection. We refer to it as proximity-based expansion algorithm around turn samples based on turn similarity. The intuition behind this algorithm is illustrated in Figure 2.

ALGORITHM 2: Finding Intersections

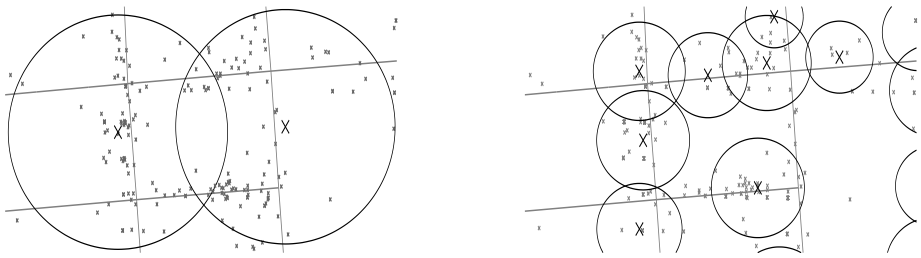
Input: A set of trajectory segments $\mathcal{T}$ belonging to the same speed category; global distance threshold δ_{max}
Output: A set of intersection nodes $\mathcal{N}$

```
1  begin
2      /* Detect turns */
3      N ← ∅
4      P ← ∅ ▷ turn samples
5      foreach (T ∈ T) do
6          foreach (P_i ∈ T) do
7              p_diff ← AngularDiff(P[i − 1], P[i], P[i + 1])
8              p_in ← Angle(P[i − 1], P[i]) ▷ incoming angle
9              p_out ← Angle(P[i], P[i + 1]) ▷ outgoing angle
10             P ← P ∪ {(P_i, p_in, p_out)}
11         end
12     end
13     /* Compute neighborhoods */
14     foreach (p ∈ P) do
15         p* ← FarthestNeighbor(p, δ_max)
16         r_p ← d(p, p*)
17     end
18     P ← Sort(P) ▷ sort by neighborhood radius in descending order
19     /* Group turns */
20     while (P ≠ ∅) do
21         p ← Pop(P)
22         G ← {p' ∈ P : d(p, p') ≤ r_p}
23         N ← N ∪ Centroid(G)
24         P ← P \ G
25     end
26     return N
27 end
```

The input to the algorithm is a set of sub-trajectories belonging to the same speed category (Figure 2(a)). First, all position samples are evaluated to detect all turn occurrences, according to the criteria of changes in direction and speed. Figure 2(b) depicts these detected turn samples. Moreover, each turn sample is assigned to a corresponding type according to the direction of its incoming and outgoing trajectory segments; in the figure this is illustrated by representing turn samples of the same type with the same color. Note that this information is also useful in order to infer turn restrictions in the network.

The next step is to group together turn samples in order to identify network nodes. This is done as explained below. First, for each turn sample p, we retrieve its farthest neighbor p^* within a neighborhood of radius δ_{max}. The latter is a relaxed, global threshold; in our experiments, we set this parameter to 25 meters. Then, we use the distance between p and its farthest neighbor p^* to define a new, adapted neighborhood for p with radius $r_p = d(p, p^*)$ (Figure 2(c)). Afterwards, we iterate again over the turn samples in descending order of this computed radius for each one. During this process, for each visited turn sample p, we create a group containing all other turn samples within distance r_p from it. The centroid of this group constitutes the location of an intersection node (Figure 2(d)). During the iteration, we skip turn samples that have already being included in a previously created group.

The steps of the algorithm are listed in ALGORITHM 2. Specifically, turn samples are first detected from the original traces (Lines 5 - 12). Then, for each turn sample, its neighborhood is calculated (Lines 14 - 17). Subsequently, the turns are sorted according to the radius of these detected neighborhoods in descending order (Line 18). Finally,



(a) Intersections in TRACEBUNDLE (b) Intersections in TRACECONFLATION

Fig. 3. Comparison of detected intersections.

turn samples are grouped together according to these formed neighborhoods, and the centroid of each group is selected for creating an intersection node (Lines 20 - 25).

ALGORITHM 2 dynamically discovers useful turning patterns (Lines 7 - 10) from the moving objects which have traveled the neighborhood (Lines 15 - 16) in a similar fashion. Thus, it is not necessary to tune several parameters in order to group together the turn samples. On the other hand, TRACEBUNDLE needs a series of experiments to establish the proper parameter setting (cf. Table I). More importantly, this setting is determined upon experimentation thus it is static and global and can not cope with heterogeneous and diverse tracking data characteristics. Intersection extraction in TRACEBUNDLE employs five parameters. The angular difference threshold holds for the detection of direction change, the mean speed assisting the indication of a turn expresses the maximum speed that vehicles may have while turning, the sampling rate indicates the maximum time interval between two consecutive positions and the turn clusters and intersection nodes threshold characterize the maximum distance for grouping together the collected samples. The important feature introduced by TRACECONFLATION is that it adopts a data-driven methodology which enables to discover from the available data useful knowledge about intersections. It achieves so by collecting and comparing within the range of δ_{max} samples which experience similar turning patterns. The noteworthy characteristic is that in Section 4, presented in the following, all the road networks have been inferred by using the same δ_{max} which is determined while the algorithm is being executed directly according to the available data, although the datasets used have heterogeneous and diverse characteristics (cf. Table II).

Figure 3 illustrates the outcome of our approach. It contrasts an output from the TRACEBUNDLE algorithm with the current approach employed by TRACECONFLATION in an excerpt taken by the same neighborhood in Berlin. Figure 3(a) shows how the clustering using global and static parameters in TRACEBUNDLE erroneously places nodes between actual intersections, missing also two of them, and resulting in a more coarse-grained representation in this case. Figure 3(b) shows the current approach with nodes being placed more accurately by taking advantage of the discovered knowledge within the neighborhood. Note that, although some of the discovered nodes do not correspond to actual intersections, still the resulting network structure more closely resembles the original one.

3.4. Conflation of Network Layers

The final part of the process comprises the fusion of the constructed network layers for the different speed categories to produce the overall road network. We build the

Table I. Parameters synopsis.

Intersection Inference		
	TRACEBUNDLE	TRACECONFLATION
Angular difference	15°	-
Mean speed	40km/h	-
Sampling rate	35s	-
Turn clusters	50m	-
Intersection nodes	25m	25m

road network incrementally starting from higher speed layers and progressing to lower speed layers. The intuition is that higher speed layers correspond to highways and avenues which can be reproduced with higher spatial accuracy. In these parts of the network, the vehicles exhibit more regular movement patterns and the GPS signal experiences fewer distortion and errors.

Fusing two network layers 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.

Figure 4 presents an example of this conflation process. Figure 4(a) shows the three road network layers that were generated after segmenting the entire trajectory dataset of Figure 1(a). Gray lines link the various connection points between the constructed networks. The final result is shown in Figure 4(b).

The algorithm for this process is outlined in ALGORITHM 3. Starting with the fast and the medium network, we identify matching nodes in terms of spatial proximity. After experimentation, we have set this parameter to 10 meters in our experiments. The next step involves introducing new intersections onto existing links (Lines 5 - 13). For instance, this is the case when the lower speed network includes additional intersection nodes along a street segment which is represented as a single link in the higher speed network. Using a buffer region around links of higher layers, we identify intersection nodes of lower layers that are close to existing links. These new intersection nodes are then mapped onto the existing link and effectively split it (Line 10). Finally, new nodes and links for the uncommon portions of the layered network are added (Lines 14 - 19). For example, a lower network link may be missing from the higher network. Here, links of lower layers are introduced by connecting them to previously introduced intersection nodes. Any intersection node that has not been introduced yet, since not connected to the higher network will be added as well. This accounts also for the case of adding complete (local) road network portions.

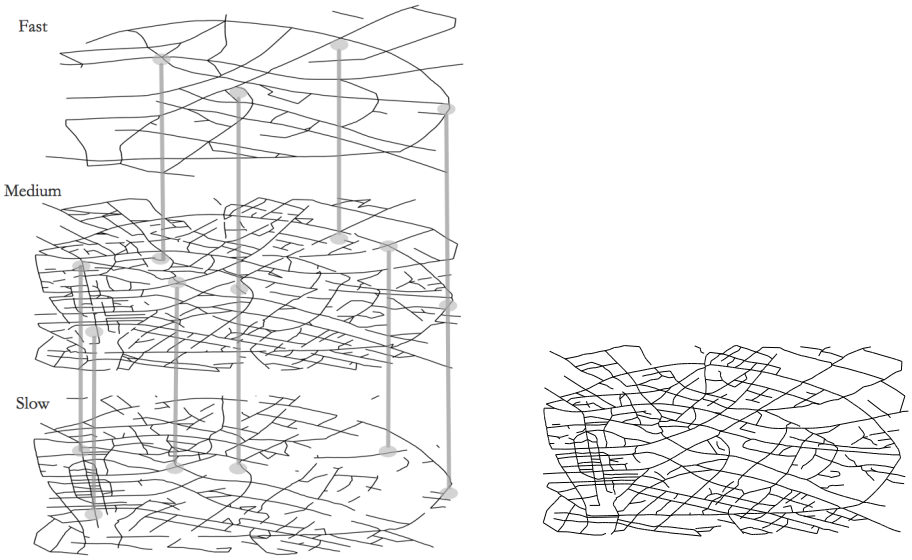
4. EXPERIMENTAL EVALUATION

In this section, we present the experimental evaluation of the TRACECONFLATION algorithm. First, we describe the datasets used in the experiments. Then, to provide some first insights, we visualize a few indicative results. Subsequently, we present a quantitative evaluation, introducing the evaluation measures used to compare the different algorithms and presenting the results.

4.1. Datasets

We have conducted experiments using real-world datasets comprising vehicle tracking data from four different cities, namely Athens, Berlin, Vienna and Chicago. These datasets cover a variety of cases with different characteristics and quality, since they involve different types of vehicles, varying sampling rates, and different network sizes.

In all cases, we consider as ground truth the corresponding road network excerpt obtained from OpenStreetMap. More specifically, we use a subgraph of the whole road



(a) Fast-medium-slow layer (b) Complete constructed network

Fig. 4. Fusion of different network layers - Berlin.

ALGORITHM 3: Conflation of Network Layers

Input: Two road network layers $G_H = (N_H, E_H)$ and $G_L = (N_L, E_L)$
Output: A fused road network graph $G_F = (N_F, E_F)$

```
1  begin
2      /* Different network layers conflation */
3      G_F ← G_H
4      N_HL ← MatchNodes(N_H, N_L) ▷ node alignment
5      foreach (N ∈ N_L \ N_HL) do
6          E ← {E ∈ E_H : Contains(Buffer(E), N)}
7          if (E ≠ ∅) then
8              N_F ← N_F ∪ {N}
9              E_F ← E_F \ E
10             E' ← Split(E, N) ▷ split original edge in two
11             E_F ← E_F ∪ E'
12         end
13     end
14     foreach (N ∈ N_L \ N_F) do
15         N_F ← N_F ∪ {N} ▷ add remaining nodes
16     end
17     foreach (E ∈ E_L \ E_F) do
18         E_F ← E_F ∪ {E} ▷ add remaining edges
19     end
20     return G_F
21 end
```

network, discarding areas in which no vehicles have moved and therefore no GPS traces have been recorded. Apparently, no method can possibly reconstruct those parts of the network. To perform this filtering, we used buffer regions of 20 meters around road network edges intersecting with our input GPS trajectories. The resulting network is used as ground truth in the experiments.

Table II. Datasets used in the experiments.

Dataset	Input Trajectories				Road network			
	Count	*Total length (km)*	*Average sampling rate (sec)*	*Average speed (km/h)*	*Num. of Nodes*	*Num. of Edges*	*Total length (km)*	*Area size (km^2)*
Athens	511	6,781	30.14	20	6,954	7,118	105	12×14
Berlin	26,831	41,116	41.98	35	5,507	6,281	89	6×6
Vienna	12,773	16,106	38.59	34	7,523	9,123	97	5.5×6
Chicago	889	2,869	3.61	33	3,309	3,814	32	7×4.5

Next, we present in more detail the characteristics of the involved datasets. A summary of the datasets' characteristics is provided in Table II.

Athens. The Athens dataset comprises GPS traces obtained from a fleet of 120 school buses. It contains 511 trajectories with a total length of 6,781 km. The average sampling rate is 30.14 sec, while the average speed is 20 km/h. The corresponding road network excerpt consists of 6,954 nodes and 7,118 edges, covering an area of 12 km $\times$ 14 km and having a total length of 105 km.

Berlin. The Berlin dataset includes GPS traces from 15,051 taxis. It contains 26,831 vehicle trajectories with a total length of 41,116 km. The average sampling rate is 41.98 sec, while the average speed is 35 km/h. The corresponding road network excerpt contains 5,507 nodes and 6,281 edges, covering an area of 6 km $\times$ 6 km and having a total length of 89 km.

Vienna. The Vienna dataset contains GPS traces from 7,434 taxis. It comprises 12,773 vehicle trajectories with a total length of 16,106 km. The average sampling rate is 38.59 sec, while the average speed is 34 km/h. The corresponding road network excerpt contains 7,523 nodes and 9,123 edges, covering an area of 5.5 km $\times$ 6 km and having a total length of 97 km.

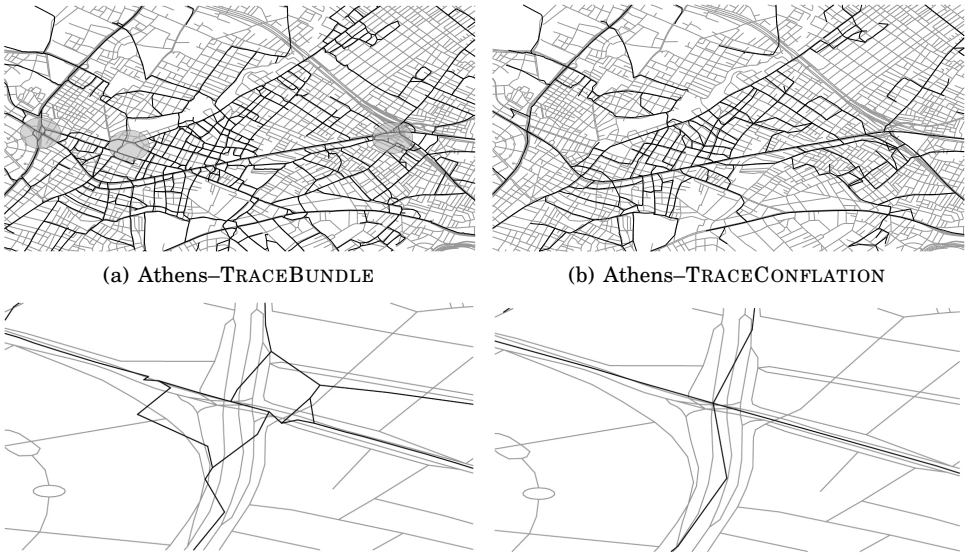
Chicago. The Chicago dataset includes GPS traces from 889 shuttle buses. It comprises 889 vehicle trajectories with a total length of 2,869 km. The average sampling rate is 3.61 sec, while the average speed is 33 km/h. The corresponding road network excerpt contains 3,309 nodes and 3,814 edges, covering an area of 7 km $\times$ 4.5 km and having a total length of 32 km.

4.2. Visual Comparison of Indicative Results

A simple and fast approach to assess the map construction results and gain insights about the performance of the algorithms is to visually compare the created road networks. To that end, we present some indicative results for Athens, Berlin, Vienna and Chicago in Figures 5 through 8.

In each figure, we present the following information. First, in images (a) and (b), we visualize the results (black lines) of TRACEBUNDLE and TRACECONFLATION, respectively, for the whole network, overlaid on top of the corresponding OpenStreetMap network (light gray lines). Moreover, we highlight (gray circles) certain areas where the results of TRACEBUNDLE were of low quality but have been reconstructed more accurately by TRACECONFLATION. For one of these, we zoom in to present a more detailed view in images (c) and (d).

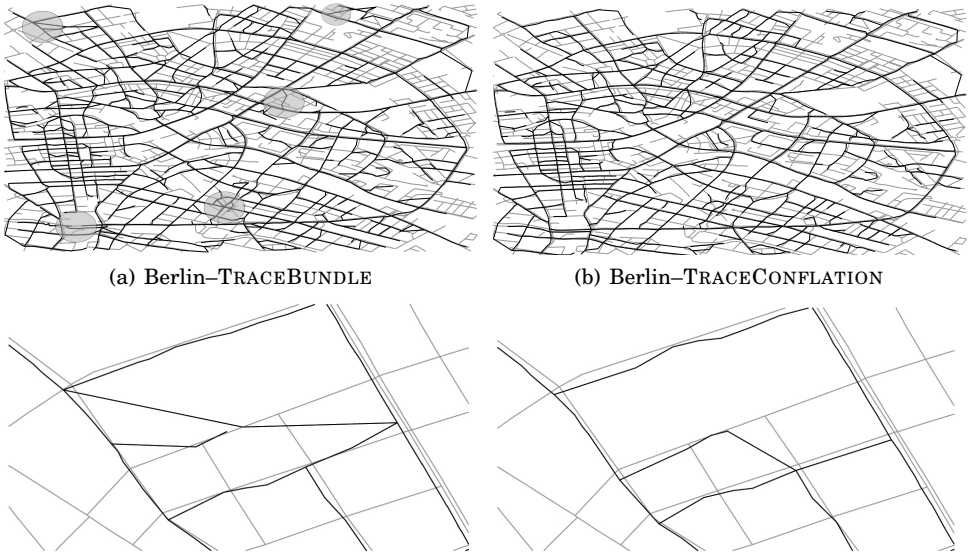
Overall, several cases can be observed where the road network constructed by TRACECONFLATION reflects more closely the topology of the ground truth road network obtained from OpenStreetMap.



(a) Athens–TRACEBUNDLE (b) Athens–TRACECONFLATION

(c) Athens closeup–TRACEBUNDLE (d) Athens closeup–TRACECONFLATION

Fig. 5. Constructed road network in Athens.



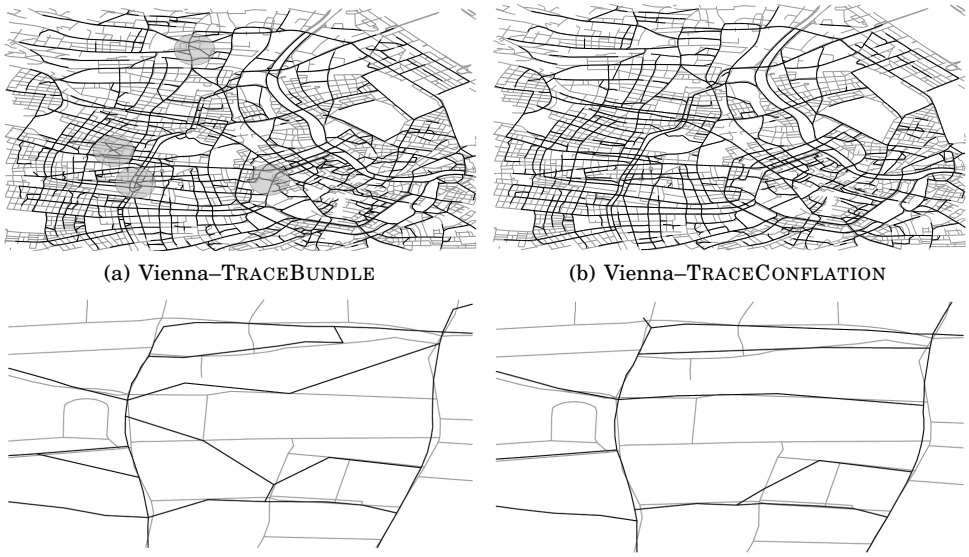
(a) Berlin–TRACEBUNDLE (b) Berlin–TRACECONFLATION

(c) Berlin closeup–TRACEBUNDLE (d) Berlin closeup–TRACECONFLATION

Fig. 6. Constructed road network in Berlin.

4.3. Quantitative Results

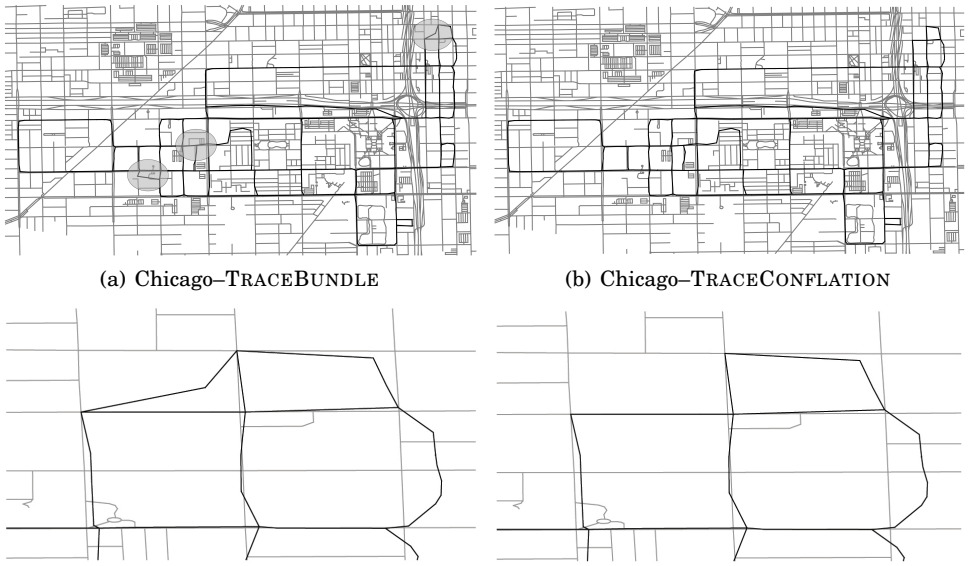
Although the visual inspection allows to derive some useful insights and quick observations regarding the performance of the algorithms, a more objective and quantifiable evaluation is needed to assess their performance. For this purpose, we proceed next to



(a) Vienna–TRACEBUNDLE (b) Vienna–TRACECONFLATION

(c) Vienna closeup–TRACEBUNDLE (d) Vienna closeup–TRACECONFLATION

Fig. 7. Constructed road network in Vienna.



(a) Chicago–TRACEBUNDLE (b) Chicago–TRACECONFLATION

(c) Chicago closeup–TRACEBUNDLE (d) Chicago closeup–TRACECONFLATION

Fig. 8. Constructed road network in Chicago.

conduct a quantitative comparison of the algorithms. We first introduce the evaluation measures used for the comparison and then we present the results.

4.3.1. Evaluation Measures. To provide a quantitative evaluation of the algorithms, we introduce two different evaluation criteria that measure the quality and accuracy of the result from two different perspectives, as detailed below.

Comparison based on network topology. The first criterion is to measure how well the topology (i.e., the nodes and edges) of the automatically constructed network matches that of the ground truth network. For this purpose, we use the well-known measures of *recall* and *precision*, appropriately adapted for our task. Intuitively, recall indicates how many of the nodes/edges of the ground truth network have been successfully discovered by the algorithm, whereas precision indicates how many of the discovered nodes/edges match with an actual node/edge in the ground truth network.

To formally define these measures in our case, we need first to define the criterion for a pair of nodes/edges from the discovered and the actual network to match. Let $G = (V, E)$ and $G^* = (V^*, E^*)$ denote, respectively, the discovered and the ground truth network. Then, given a pair of nodes $v \in V$ and $v' \in V^*$, we define the following matching function:

$$match(v, v') = \begin{cases} true, & \textbf{\textit{if}}\ dist(v, v') \leq \epsilon \\ false, & \textbf{\textit{otherwise.}} \end{cases}$$

where $dist(v, v')$ is the Euclidean distance between the locations of the nodes and ϵ is an error tolerance parameter.

Further, we can use the matching function on nodes to define a matching function for edges. Given a pair of edges $e = (v_i, v_j) \in E$ and $e' = (v'_i, v'_j) \in E^*$, we define the matching function as follows:

$$match(e, e') = \begin{cases} true, & \textbf{\textit{if}}\ match(v_i, v'_i)\ \textbf{\textit{AND}}\ match(v_j, v'_j) \\ false, & \textbf{\textit{otherwise.}} \end{cases}$$

Based on the above, we define the following sets of nodes:

$$V_R = \{v' \in V^* : \exists v \in V\ \ \textbf{\textit{such that}}\ \ match(v, v')\}$$

and

$$V_P = \{v \in V : \exists v' \in V^*\ \ \textbf{\textit{such that}}\ \ match(v, v')\}$$

Now, we can define the measures of recall and precision for the set of nodes of the two compared networks as follows:

$$recall(V, V^*) = \frac{|V_R|}{|V^*|}\ \ \textbf{\textit{and}}\ \ precision(V, V^*) = \frac{|V_P|}{|V|}$$

The two measures are defined analogously for the case of edges.

Finally, recall and precision can be combined to derive a single score, known as F-measure, which is computed as the harmonic mean:

$$F = 2 \cdot \frac{recall \cdot precision}{recall + precision}$$

Comparison based on network paths. The second evaluation measure used in the experiments is a method that was introduced in [Karagiorgou and Pfoser 2012] and

relies on the comparison of randomly selected and distinct shortest paths between corresponding pairs of nodes in the compared networks. This measure is motivated from the fact that perhaps the most common and important use of such maps is to support navigation services. Therefore, given a source and target node, a measure that indicates the quality and accuracy of the automatically constructed road network with respect to the actual one is to compare the shortest path obtained from the former with that obtained from the latter.

Assume a pair of source and target nodes v_s and v_t in the constructed road network graph G which correspond to (i.e., match with) the pair of nodes v'_s and v'_t in the ground truth network G^*. Moreover, let p_{st} and p'_{st} be the shortest paths between those pairs of nodes in the networks G and G^*, respectively. Then, a distance measure $dist(p_{st}, p'_{st})$ between the two paths can be used as a measure to evaluate the accuracy of G with respect to G^* for the scenario of navigating from v'_s to v'_t.

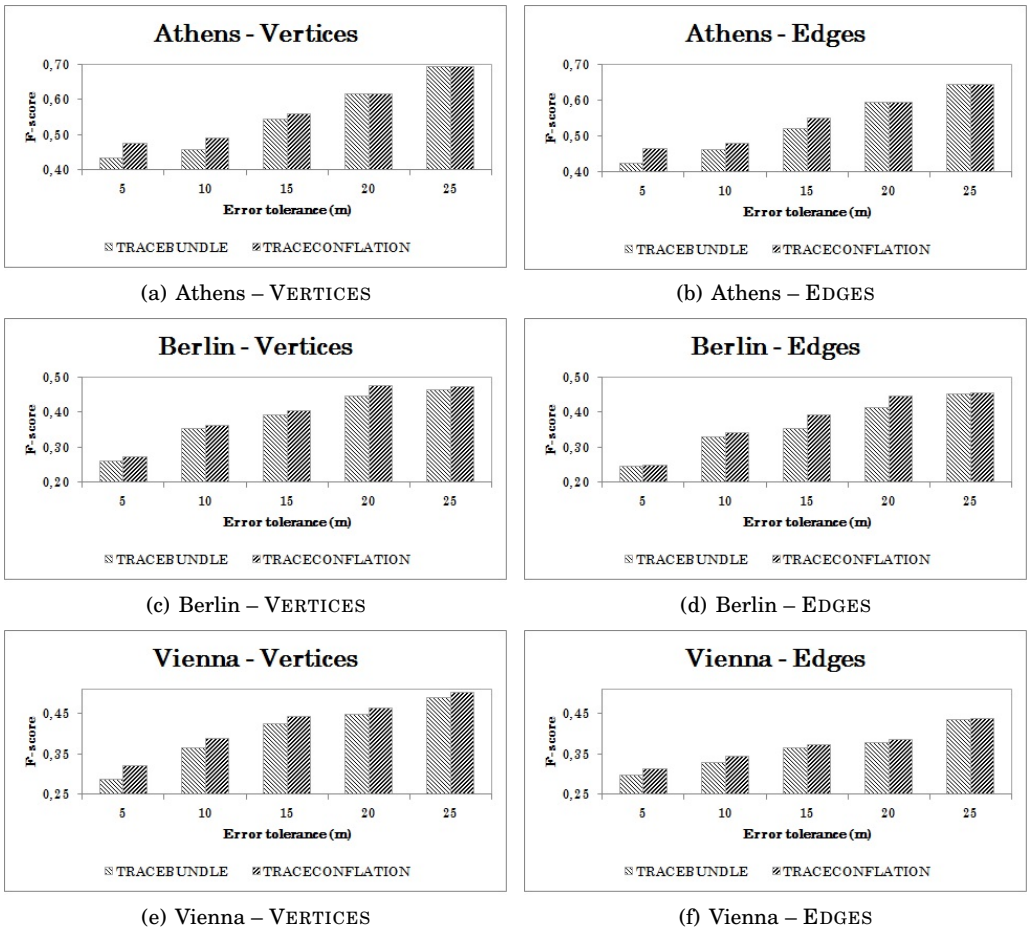
In our evaluation, to measure path distance we use the Discrete Fréchet distance, which is an approximation of the Fréchet metric for polygonal curves [Eiter and Mannila 1994]. In addition, to ensure that the results are not sensitive to this particular distance function, we also compute the average vertical distance of the respective line segments across the two paths.

The rationale for using this evaluation criterion is that measuring the similarity over entire paths instead of individual edges allows to draw conclusions regarding more extensive portions of the road network and, especially, to take into account the connectivity of the generated network. The lower the distance of the shortest paths computed over the automatically constructed network is to those computed over the ground truth network, the more suitable the derived network would be for use in a navigation application.

4.3.2. Comparing TRACECONFLATION *to* TRACEBUNDLE. The primary objective of our experimental evaluation is to evaluate the benefits of our proposed layered approach for the construction of a road network map. Hence, we compare the proposed TRACECONFLATION algorithm with our previous method, TRACEBUNDLE [Karagiorgou and Pfoser 2012], which treats all the input data uniformly. The evaluation is performed on the road networks of Athens, Berlin and Vienna, using both evaluation criteria described above.

We apply both algorithms on each of the three datasets to derive a road network from the original vehicle tracking data. Then, we compare the resulting networks to their respective ground truth networks. Figure 9 presents the F-score achieved by each of the two algorithms in each dataset. As can be observed, TRACECONFLATION becomes more effective than TRACEBUNDLE in the cases where the error tolerance is smaller, i.e., the evaluation is stricter, meaning that it succeeds to construct a network of better spatial accuracy. As the error tolerance becomes more relaxed, both algorithms exhibit similar behavior. Recall that for the case of edges, the matching of two edges requires that both their endpoints should match, in addition of being connected with a link. Thus, since this condition is stricter, it is natural that the scores achieved for recall and precision of edges are slightly lower than those for nodes.

Then, we compare the accuracy of the derived networks according to the criterion based on comparing shortest paths computed in each one. Specifically, we select randomly a set of 500 pairs of source and target nodes in the ground truth network, founding also their corresponding ones in the discovered networks. For each pair, we compute the shortest path in the respective network, and we measure the Discrete Fréchet distance and the average vertical distance between the path in the discovered network and that in the ground truth network. The results are shown in Table III, where the minimum, maximum and average values of the distance over all 500 paths



(a) Athens – VERTICES (b) Athens – EDGES

(c) Berlin – VERTICES (d) Berlin – EDGES

(e) Vienna – VERTICES (f) Vienna – EDGES

Fig. 9. F-Score on the topological components of the road network.

are reported. In addition, we list the minimum, maximum and average length of the computed shortest paths. As it can be seen, these lengths vary from one to several kilometers, thus including different types of routes, both within and across city neighborhoods.

The results for Athens depict that TRACECONFLATION constructs a network of better spatial accuracy compared to TRACEBUNDLE. The cases of Berlin and Vienna show an improvement of TRACECONFLATION with respect to path similarity and, thus, constructed network. This can be interpreted by the fact that in both cases TRACECONFLATION better handles the heterogeneous nature of the tracking data coming from these cities. Instead, the global values for these parameters set by TRACEBUNDLE limit its performance. The results are similar when considering the average vertical distance measure.

4.3.3. Comparison with other methods. In previous work [Ahmed et al. 2015], a comparison of the TRACEBUNDLE algorithm to other existing map construction algorithms has been presented using the Chicago dataset. The algorithms included in the comparison represent the state of the art over the past several years and constitute rep-

Table III. Comparison based on similarity of computed shortest paths.

	Discrete Fréchet distance (m)			Average Vertical distance (m)			Shortest path length (km)		
	min	max	avg	min	max	avg	min	max	avg
Athens									
TRACEBUNDLE	19	432	125	9	225	98	1.01	11.62	6.84
TRACECONFLATION	18	404	118	6	176	95	0.99	11.58	6.76
Berlin									
TRACEBUNDLE	18	428	183	8	209	106	1.32	5.67	3.27
TRACECONFLATION	14	421	179	6	208	104	1.29	5.58	3.19
Vienna									
TRACEBUNDLE	15	416	208	4	212	108	1.28	4.31	2.67
TRACECONFLATION	12	410	203	2	198	97	1.15	4.19	2.56

Table IV. Comparing all methods using similarity of computed shortest paths.

	Discrete Fréchet distance (m)			Average Vertical distance (m)			Shortest path length (km)		
	min	max	avg	min	max	avg	min	max	avg
Chicago									
TRACEBUNDLE	4	103	41	2	50	21	0.90	6.05	3.82
TRACECONFLATION	4	98	38	2	45	19	0.90	5.98	3.79
Ahmed and Wenk 2012	13	208	97	6	92	43	1.21	6.95	4.45
Biagioni and Eriksson 2012	4	98	40	2	49	20	0.89	6.03	3.76
Cao and Krumm 2009	7	131	67	4	76	41	1.02	6.87	3.94
Davies et al. 2006	5	97	41	3	51	23	0.93	6.08	3.88
Edelkamp and Schroedl 2003	12	211	98	5	89	41	1.19	6.88	4.32
Ge et al. 2011	19	241	127	8	94	49	1.58	6.98	4.69

resentatives of different map construction algorithm categories. The criterion used in that evaluation was the measure based on the similarity of computed shortest paths in the different networks. Here, we extend this evaluation to include the proposed TRACECONFLATION algorithm, and we also apply the evaluation measures of recall and precision described above.

The results for the F-score are shown in Figure 10. It is worth noting that even though the Chicago dataset constitutes a less challenging dataset, given its smaller size and –most importantly– its very high sampling rate, the benefits of TRACECONFLATION over the rest of the methods can also be observed here.

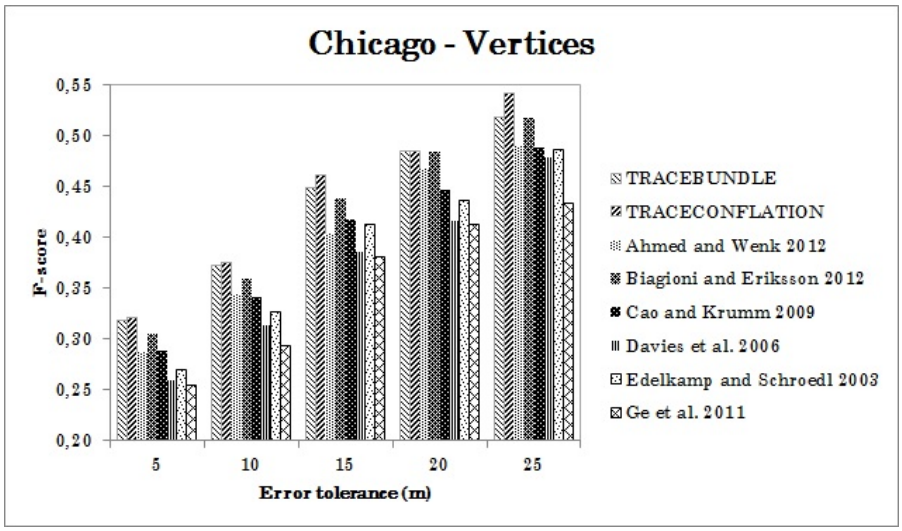
Finally, the results for the comparison based on shortest paths similarity are listed in Table IV. For most algorithms the resulting paths have a small distance to the compared shortest path in the ground truth map. However, in some other algorithms, due to significant differences in the constructed map, different shortest paths have been computed that have a larger distance to the shortest path in the ground truth map.

4.4. Summary

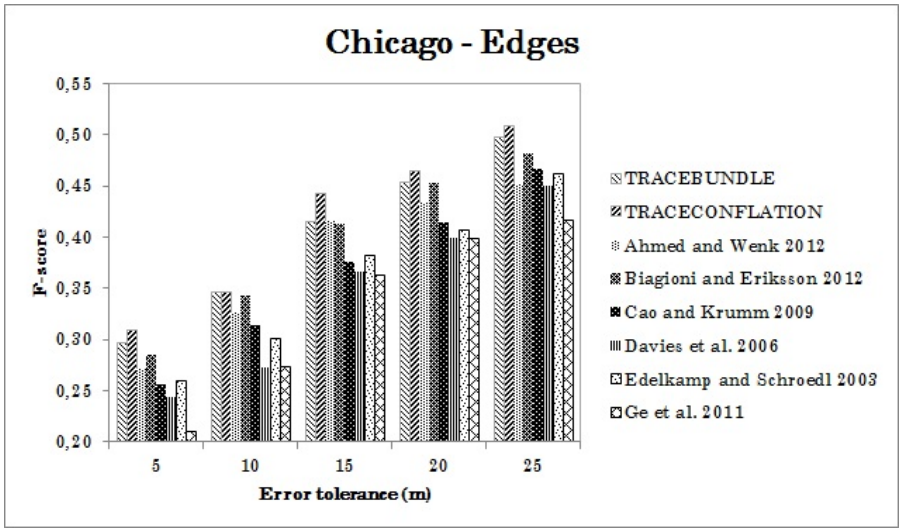
In this experimental evaluation, we have used two evaluation criteria, one based on the F-measure and one based on shortest path similarity, to compare the performance of our proposed TRACECONFLATION algorithm to its predecessor, TRACEBUNDLE, as well as to other algorithms representing the state of the art over the past several years being the representatives of different map construction algorithm classes.

The selection of the right algorithm highly depends on the quality and the characteristics (i.e. heterogeneity, sparsity, etc.) of the input data and for what purpose the map will be utilized. For instance, for the *Chicago* dataset the KDE-based algorithm by Davies et al. [Davies et al. 2006] generates a very good-quality map in terms of spatial distance to the ground truth map, but if the user is interested in maps with good coverage this algorithm will not be the best choice as it ignores tracks in sparse areas as outliers/noise.

The experiments have shown that TRACECONFLATION, by segmenting vehicle trajectories based on speed profiles and performing a layered construction of the network, produces navigable road networks of higher accuracy compared to those derived by the



(a) Chicago – VERTICES



(b) Chicago – EDGES

Fig. 10. Comparing all methods using F-Score.

other algorithms. Also, by taking into account the speed profiles which is an inherent characteristic of road networks, TRACECONFLATION can handle heterogeneous data with different characteristics. Visual inspection shows that the road networks produced by TRACECONFLATION resemble more closely the actual road networks.

5. CONCLUSIONS

In this paper, we have described a new approach to the map construction problem that is based on segmenting the trajectory datasets based on speed profiles, constructing different layers of the map separately, and then fusing them to derive the complete road network. The introduced TRACECONFLATION algorithm employs also an

improved method over its predecessor, TRACEBUNDLE, for detecting node intersections in a more robust and flexible manner. Performing an experimental evaluation using four different trajectory datasets, TRACECONFLATION has been shown to produce road networks of improved accuracy, which more closely resemble the structure and topology of the underlying ground truth networks.

In the future, we plan to experiment more with our proposed algorithm in the context of crowdsourcing platforms such as OpenStreetMap. As it is becoming increasingly easier to gain access to tracking data sources, a map construction algorithm could improve the process of enhancing and enriching crowdsourced map datasets. To this effect, we are investigating automatic methods to infer useful spatial and semantic knowledge from diverse data sources coming from social media applications and mobile phones.

REFERENCES

Mridul Aanjaneya, Frederic Chazal, Daniel Chen, Marc Glisse, Leonidas J. Guibas, and Dmitriy Morozov. 2011. Metric Graph Reconstruction from Noisy Data. In *Proceedings of the 27th ACM Symposium on Computational Geometry*. 37–46.

Gabriel Agamennoni, Juan Nieto, and Eduardo M. Nebot. 2011. Robust Inference of Principal Road Paths for Intelligent Transportation Systems. *IEEE Transactions on Intelligent Transportation Systems* 12, 1 (2011), 298–308.

Mahmuda Ahmed, Sophia Karagiorgou, Dieter Pfoser, and Carola Wenk. 2015. A Comparison and Evaluation of Map Construction Algorithms Using Vehicle Tracking Data. *GeoInformatica* 19, 3 (2015), 601–632.

Mahmuda Ahmed and Carola Wenk. 2012. Constructing Street Networks from GPS Trajectories. In *Proceedings of the 20th Annual European Symposium on Algorithms*. 60–71.

James Biagioni and Jakob Eriksson. 2012. Map Inference in the Face of Noise and Disparity. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*. 79–88.

Sotiris Brakatsoulas, Dieter Pfoser, Randall Salas, and Carola Wenk. 2005. On Map-Matching Vehicle Tracking Data. In *Proceedings of the 31st International Conference on Very Large Data Bases*. 853–864.

René Brüntrup, Stefan Edelkamp, Shahid Jabbar, and Björn Scholz. 2005. Incremental Map Generation with GPS Traces. In *Proceedings of the 2005 IEEE Intelligent Transportation Systems*. 574–579.

Lili Cao and John Krumm. 2009. From GPS Traces to a Routable Road Map. In *Proceedings of the 17th International Conference on Advances in Geographic Information Systems*. 3–12.

Chen Chen and Yinhang Cheng. 2008. Roads Digital Map Generation with Multi-Track GPS Data. In *Proceedings of the 2008 International Workshop on Geoscience and Remote Sensing*. 508–511.

Daniel Chen, Leonidas J. Guibas, John Hershberger, and Jian Sun. 2010. Road Network Reconstruction for Organizing Paths. In *Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms*. 1309–1320.

Jonathan J. Davies, Alastair R. Beresford, and Andy Hopper. 2006. Scalable, Distributed, Real-Time Map Generation. *IEEE Pervasive Computing* 5, 4 (2006), 47–54.

Stefan Edelkamp and Stefan Schroedl. 2003. Route Planning and Map Inference with Global Positioning Traces. *Computer Science in Perspective* (2003), 128–151.

Thomas Eiter and Heikki Mannila. 1994. *Computing discrete Fréchet distance*. Technical Report CD-TR 94/64. Christian Doppler Laboratory for Expert Systems, TU Vienna, Austria.

Alireza Fathi and John Krumm. 2010a. Detecting Road Intersections from GPS Traces. In *Proceedings of the 6th International Conference on Geographic Information Science*. 56–69.

Alireza Fathi and John Krumm. 2010b. Inferring the Road Network from GPS Data. In *Proceedings of the 6th International Conference on Geographic Information Science*. 56–69.

Xiaoyin Ge, Issam Safa, Mikhail Belkin, and Yusu Wang. 2011. Data Skeletonization via Reeb Graphs. In *Proceedings of the 25th Annual Conference on Neural Information Processing Systems*. 837–845.

Tao Guo, Kazuaki Iwamura, and Masashi Koga. 2007. Towards High Accuracy Road Maps Generation from Massive GPS Traces Data. In *Proceedings of the 2007 IEEE International Geoscience and Remote Sensing Symposium*. 667–670.

Sera Jang, Taehwan Kim, and Eunseok Lee. 2010. Map Generation System with Lightweight GPS Trace Data. In *Proceedings of the 12th International Conference on Advanced Communication Technology*. 1489–1493.

Sophia Karagiorgou and Dieter Pfoser. 2012. On Vehicle Tracking Data-Based Road Network Generation. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*. 89–98.

Xuemei Liu, James Biagioni, Jakob Eriksson, Yin Wang, George Forman, and Yanmin Zhu. 2012. Mining Large-Scale, Sparse GPS Traces for Map Inference: Comparison of Approaches. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 669–677.

Brian Niehofer, Ralf Burda, Christian Wietfeld, Franziskus Bauer, and Oliver Lueert. 2009. GPS Community Map Generation for Enhanced Routing Methods Based on Trace-Collection by Mobile Phones. In *Proceedings of the 1st International Conference on Advances in Satellite and Space Communications*. 156–161.

Mohammed A. Quddusa, Washington Y. Ochiengb, and Robert B. Nolandb. 2007. Current Map-Matching Algorithms for Transport Applications: State-of-the-art and Future Research Directions. *Transportation Research Part C: Emerging Technologies* 15, 5 (2007), 312–328.

Seth Rogers, Pat Langley, and Christopher Wilson. 1999. Mining GPS Data to Augment Road Models. In *Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 104–113.

Stefan Schroedl, Kiri Wagstaff, Seth Rogers, Pat Langley, and Christopher Wilson. 2004. Mining GPS Traces for Map Refinement. *Data Mining and Knowledge Discovery* 9, 1 (2004), 59–87.

Zhangqing Shan, Hao Wu, Weiwei Sun, and Baihua Zheng. 2015. COBWEB: A Robust Map Update System Using GPS Trajectories. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*. 927–937.

Wenhuan Shi, Shuhan Shen, and Yuncai Liu. 2009. Automatic Generation of Road Network Map from Massive GPS Vehicle Trajectories. In *Proceedings of the 12th International IEEE Conference on Intelligent Transportation Systems*. 1–6.

Albert Steiner and Axel Leonhardt. 2011. Map Generation Algorithm Using Low Frequency Vehicle Position Data. In *Proceedings of the 90th Annual Meeting of the Transportation Research Board*. 1–17.

Mohamad Tavakoli and Azriel Rosenfeld. 1982. Building and Road Extraction from Aerial Photographs. *IEEE Transactions on Systems, Man and Cybernetics* 12, 1 (1982), 84–91.

Suyi Wang, Yusu Wang, and Yanjie Li. 2015. Efficient Map Reconstruction and Augmentation via Topological Methods. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 1–10.

Yin Wang, Xuemei Liu, Hong Wei, George Forman, Chao Chen, and Yanmin Zhu. 2013. CrowdAtlas: Self Updating Maps for Cloud and Personal Use. In *Proceedings of the 11th Annual International Conference on Mobile Systems, Applications, and Services*. 27–40.

Stewart Worrall and Eduardo Nebot. 2007. Automated Process for Generating Digitised Maps through GPS Data Compression. In *Proceedings of the 2007 Australasian Conference on Robotics and Automation*.

Lijuan Zhang, Frank Thiemann, and Monika Sester. 2010. Integration of GPS Traces with Road Map. In *Proceedings of the 2nd International Workshop on Computational Transportation Science*. 17–22.

Received October 2015; revised June 2016; accepted April 2017