A comparison and evaluation of map construction algorithms using vehicle tracking data

**Mahmuda Ahmed · Sophia Karagiorgou ·
Dieter Pfoser · Carola Wenk**

Abstract Map construction methods automatically produce and/or update street map datasets using vehicle tracking data. Enabled by the ubiquitous generation of geo-referenced tracking data, there has been a recent surge in map construction algorithms coming from different computer science domains. A cross-comparison of the various algorithms is still very rare, since (i) algorithms and constructed maps are generally not publicly available and (ii) there is no standard approach to assess the result quality, given the lack of benchmark data and quantitative evaluation methods. This work represents a first comprehensive attempt to benchmark such map construction algorithms. We provide an evaluation and comparison of seven algorithms using four datasets and four different evaluation measures. In addition to this comprehensive comparison, we make our datasets, source code of map construction algorithms and evaluation measures publicly available on http://mapconstruction.org. This site has been established as a repository for map construction data and algorithms and we invite other researchers to contribute by uploading code and benchmark data supporting their contributions to map construction algorithms.

M. Ahmed
University of Texas at San Antonio, San Antonio, TX, USA
e-mail: mahmudaahmed@gmail.com

S. Karagiorgou
National Technical University of Athens, Athens, Greece
e-mail: sokaragi@mail.ntua.gr

D. Pfoser (✉)
George Mason University, Fairfax, VA, USA
e-mail: dpfoser@gmu.edu

C. Wenk
Tulane University, New Orleans, LA, USA
e-mail: cwenk@tulane.edu

1 Introduction

Street maps and transportation networks are of fundamental importance in a wealth of applications. In the past, the production of street maps required expensive field surveying and labor-intensive postprocessing. Proprietary data vendors such as NAVTEQ (now Nokia), TeleAtlas (now TomTom) and Google therefore dominated the market. Over the last several years, Volunteered Geographic Information (VGI) [24] efforts such as OpenStreetMap (OSM) [26, 34] have complemented commercial map datasets. They provide map coverage especially in areas which are of less commercial interest. VGI efforts however still require dedicated users to author maps using specialized software tools. Lately, on the other hand, the commoditization of GPS technology and integration in mobile phones coupled with the advent of low-cost fleet management and positioning software has triggered the generation of vast amounts of tracking data. As a size indicator one can consider the contribution of tracking data in OpenStreetMap, which is steadily increasing in size and currently amounts to 2.6 trillion points [33]. Besides the use of such data in traffic assessment and forecasting [18], i.e., map-matching vehicle trajectories to road networks to obtain travel times [9], there has been a recent surge of actual *map construction algorithms* that derive not only travel time attributes but actual road network geometries from tracking data, e.g., [1, 2, 4, 7, 8, 10–13, 16, 17, 20, 23, 25, 27, 28, 30, 39–41, 43, 45]. Among those only a few algorithms give theoretical quality guarantees [1, 4, 13]. An example of a constructed map is given in Fig. 1, which shows (a) the vehicle trajectories collected for Berlin in grey color and (b) the respective constructed map, shown in black color, using the algorithm of [28] with an OpenStreetMap background map, shown in grey color.

A major challenge in the research community is to compare the performance and to evaluate the quality of the various map construction algorithms. Visual inspection remains the most common evaluation approach throughout the literature and only a few recent papers incorporate quantitative distance measures [3, 7, 8, 28, 30]. However, the cross-comparison of different algorithms remains rare, since algorithms and constructed maps are generally not publicly available. Also, there is a lack of benchmark data, and the quantitative evaluation with suitable distance measures is in its infancy. A cultural shift has recently been triggered by Biagioni and Eriksson [7]: In addition to providing an extensive survey of eleven map construction algorithms, they have performed a quantitative evaluation of three representative map construction algorithms. They have made their implementations of these algorithms, as well as their dataset, publicly available. The present work complements and significantly expands these benchmarking efforts to provide an evaluation and comparison
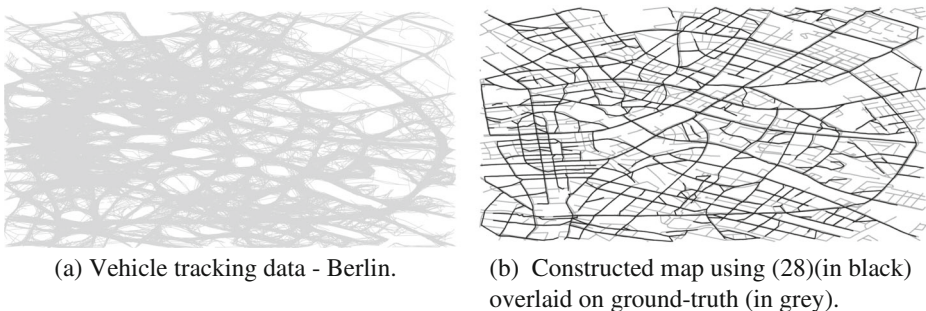

(a) Vehicle tracking data - Berlin.   (b) Constructed map using (28)(in black) overlaid on ground-truth (in grey).

Fig. 1 Vehicle tracking data vs constructed map overlayed on ground-truth

of more map construction algorithms on more diverse datasets using various quality measures suitable for different applications. Such an effort can only be sustained in a culture of sharing that makes data, methods and source code publicly available.

In this work, we evaluate and compare seven map construction algorithms using four benchmark tracking datasets and four different distance measures. The *algorithms* we compare represent the state-of-the-art over the past several years and constitute representatives of different map construction algorithm classes. The algorithms include the recent algorithms by Ahmed and Wenk [4], by Ge et al. [23], and by Karagiorgou and Pfoser [28], in addition to the algorithms by Cao and Krumm [11], Davies et al. [16], Edelkamp and Schrödl [17], and Biagioni and Eriksson [8]. Among those, the algorithms by [11], [16] and [17] were previously compared by Biagioni and Eriksson [7]. We have used the publicly available implementations of the algorithms by [11], [16, 17] and by [8], and the authors of [23] ran their algorithm for us. We have also made our own implementations of the algorithms by [4, 28] publicly available (see below). The *four distance measures used to assess the constructed map quality* comprise two novel distance measures that have not been used for comparative evaluations of map construction before and that work with unmodified and unbiased ground-truth maps: the Directed Hausdorff distance [6] and the path-based distance measure presented by Ahmed et al. [3]. We also use a distance measure based on shortest paths by Karagiorgou and Pfoser [28] and the graph-sampling-based distance measure by Biagioni and Eriksson [8]. The implementation of the latter distance measure [8] has been made available to us by the authors. The *tracking datasets* include the *Chicago* dataset provided by Biagioni and Eriksson [7, 8], and three additional tracking datasets: two from *Athens*, Greece and one from *Berlin*, Germany (see details in Section 4). They are available together with unmodified ground-truth maps obtained from OpenStreetMap. We use different datasets because they cover diverse roads (i.e. highways, secondary roads), different sampling rates, and different scale.

In addition to providing the largest comprehensive comparison of map construction algorithms, we make our three new benchmark datasets, the map construction algorithms and outputs by Ahmed and Wenk [4] and by Karagiorgou and Pfoser [28], as well as the metric code for computing the three distance measures: the Directed Hausdorff distance [6], the path-based distance [3] and shortest path based measure [28] publicly available on the internet at http://mapconstruction.org. We have established this Web site as a repository for map construction data and algorithms, and we invite other researchers to contribute by uploading code and benchmark data supporting their map construction algorithms. We expect that such a central repository will encourage a culture of sharing and will enable the development of improved map construction algorithms.

Our main goal with this work is to provide a common platform to do comparative analysis of map construction algorithms. As different distance measures capture different features of a constructed map, it is hard to combine them into a single score and rank the algorithms based on that. Also, which algorithm is the best highly depends on the quality of the input data and for what purpose the map will be used. For example, for the *Chicago* dataset the KDE-based algorithm by Davies et al. [16] generates a very good-quality map in terms of spatial distance to the ground-truth map (captured using path-based and Directed Hausdorff distance), but if the user is interested in maps with good coverage (captured by shortest path based and graph-sampling based distance measures) this algorithm will not be the best choice as it ignores tracks in sparse areas as outliers/noise. So, we leave it to the user to pick the distance measure that suits her needs best.

The outline of the paper is as follows. Section 2 surveys map construction algorithms by introducing categories for types of algorithms and gives more details on the algorithms

that we will use in our evaluation. Section 3 discusses quality measures that will allow us to assess the quality of the constructed maps. The tracking datasets that we provide for evaluation purposes are briefly discussed in Section 4. The datasets are available for download and also include the respective ground-truth map data. A comprehensive performance study comparing the various algorithms across datasets is given in Section 5. Finally, Section 6 provides conclusions and directions for future work.

2 Map construction algorithms

We assume that the input is given as a set of *tracks*, where each track is a sequence of *measurements*. Each measurement consists of a point (latitude/longitude or (x, y)-coordinates after suitable projection), a time stamp, and optionally additional information such as vehicle heading or speed. The desired output is to construct a *street map*, which we model as an undirected geometric graph in the plane. There are many possible models for street maps, mostly depending on the desired application and granularity. For example, an intersection can be modeled as a single vertex embedded as a point in the plane, or it could be a set of vertices possibly annotated with turn restrictions, or it could be a region. An edge can be modeled as an abstract connection between vertices, as a curve embedded in the plane, as a set of curves to model multiple lanes, and an edge might be directed to model one-way streets. We focus on the most basic model of *a street map as an undirected geometric graph*, where each vertex is embedded as a point in the plane and each edge is a polygonal curve that connects two vertices. The map construction algorithms in the literature generally model the maps as undirected graphs or different variants of directed graphs. But often, an undirected graph is computed as a first step and additional information such as edge directions, number of lanes, turn restrictions, and mean speed are computed in an additional post-processing step, e.g. [8, 11, 16, 17, 39]. We therefore perform the comparison on a common street map representation based on undirected graphs, although some algorithms may produce some additional attributes.

2.1 Related work

There exist several different approaches in the literature for constructing street maps from tracking data. These can be organized into the following categories: Point clustering (this includes *k*-means algorithms and Kernel Density Estimation (KDE) as described in Biagioni and Eriksson [8]), incremental track insertion, and intersection linking.

2.1.1 Point clustering

Algorithms in this general category assume the input to be a set of points, which are then clustered in different ways to obtain street segments that result in a street map. The input point set either comprises the set of all raw input measurements, or a dense sample of all input tracks. Here, the input tracks are assumed to be continuous curves obtained from interpolating (usually piecewise-linearly) between measurements. The point clustering techniques can be reduced to two types of methods for obtaining a street map. One approach type (cf. [17]) initially clusters the points to generate intersections and then computes the connecting segments as centerlines based on the trace points connecting the respective intersection clusters. Other approaches, such as KDE methods, compute the street map in one sweep. The set of points are interpreted as a skeleton image of the road network. The street

map is computed as the set of centerlines derived from this image using, e.g., kernel density estimates.

The first approach type, spearheaded by Edelkamp and Schrödl [17], employs the *k*-means algorithm to cluster the input point set, using distance measures (e.g., Euclidean distance) and possibly also vehicle heading of the measurement, as a condition to introduce seeds at fixed distances along a path. Their map construction algorithm incorporates new algorithms for road segmentation, map-matching, and lane clustering. In [39] this approach was used to refine an existing map rather than building it entirely from scratch. In their short paper [25], Guo et al. make use of statistical analysis of GPS tracks in order to extract central line representation of a street, assuming that the GPS data follows a symmetric 2D Gaussian distribution. This assumption may become unrealistic, especially in error-prone environments. Worrall et al. [43] compute point clusters based on location and heading, and in a second step link these clusters together using non-linear least-squares fitting. They emphasize compression of the input tracks to infer a digitized road map and present their results only for small datasets. They are mostly concerned with topological elements and not with connected way points. Agamennoni et al. [2] present a machine-learning method to consistently build a representation of the map mostly in dynamic environments such as open-pit mines. They focus on estimating a set of principal curves from the input traces to represent the constructed map. Liu et al. [30] first cluster line segments based on proximity and direction, and then use the resulting point clusters and fit polylines to them, to extract road segments. In our comparisons, we use the algorithm by Edelkamp and Schrödl [17].

Another approach employs KDE methods to first transform the input point set to a density-based discretized image. Most of the KDE algorithms function well either when the data is frequently sampled (i.e., once per second) [12], or when there is a lot of data redundancy [8, 16, 40, 41]. A similar approach to [8] is presented in Liu et al. [30]. Generally, KDE algorithms have a hard time overcoming the problem of noisy samples when they accumulate in an area. Recently, Wang et al. [42] addressed the problem of map updates by applying their approach to OpenStreetMap data using a KDE-based approach. In our comparisons, we use the algorithms by Davies et al. [16] and by Biagioni and Eriksson [8], which are both KDE-based but use very different approaches to extract the map from the kernel density estimate.

In the computational geometry community, map construction algorithms have been proposed that cluster the input points using local neighborhood properties by employing Voronoi diagrams, Delaunay triangulations [13, 23], or other neighborhood complexes such as the Vietoris-Rips complex [1]. All these algorithms assume a densely sampled input point set, and provide theoretical quality guarantees for the constructed output map, under certain assumptions on the underlying street map and the input tracks. Aanjaneya et al. [1] view street maps as metric graphs, and they focus on computing the combinatorial structure by computing an almost isometric space with lower complexity, but they do not compute an explicit embedding of vertices and edges. Chen et al. [13] focus on detecting "good" street portions in the map and connect them subsequently. The theoretical quality guarantees, however, assume dense point sample coverage and error bounds, and make assumptions on the road geometry. In our comparisons, we use the algorithm by Ge and Wang [23].

2.1.2 Incremental track insertion

Algorithms in this category construct a street map by incrementally inserting tracks into an initially empty map [32], often making use of map-matching ideas [36]. Distance measures and vehicle headings are also used to perform additions and deletions during the incremental

construction of the map. Intuitively, these methods use the first trace as a base map and refine it incrementally by adding more traces. With each insertion, new map detail is added and existing geometry is updated using interpolation.

One of the first algorithms in this category [38] clusters the tracks merely to refine an existing map and not to compute it from scratch. Cao and Krumm [11] first introduce a clarification step in which they modify the input tracks by applying physical attraction to group similar input tracks together. Then they incrementally insert each track by using local criteria such as distance and direction. Bruntrup et al. [10] propose a spatial-clustering based algorithm that requires high quality tracking data (sampling rate and positional accuracy). The work in [45] discusses a map update algorithm based on spatial similarity. It uses a method similar to GPS trace merging to continuously refine existing road maps. Ahmed and Wenk [4] present an incremental method that employs the Fréchet distance to partially match the tracks to the map. In our comparisons, we use the algorithms by Cao and Krumm [11] and by Ahmed and Wenk [4], which use very different approaches for incremental track insertion.

2.1.3 Intersection linking

The intersection linking approach emphasizes the correct detection of intersection nodes and the linkage of these nodes, both, in terms of connectivity and actual geometry. Intersections are identified based on movement characteristics (speed, direction) or point density. The intersections are then linked by interpolating the geometry of the connecting traces.

Fathi and Krumm [20] provide an approach that detects intersections by using a prototypical detector trained on ground truth data from an existing map. While a map is finally derived, their approach works best for geometries of vertically aligned street maps and it uses frequently sampled data of $1s$ or $5s$. The method by Karagiorgou and Pfoser [28] relies on detecting changes in the direction of movement to infer intersection nodes, and then "bundling" the trajectories around them to create the map edges. It uses less frequently sampled data ($>30s$) and produces street maps for geometries of arbitrary road networks. In our comparisons, we use the algorithm by Karagiorgou and Pfoser [28].

2.2 Compared algorithms

Here we give some more details on the map construction algorithms that we compare in Section 5. The algorithms and respective categories are shown in Table 1.

Table 1 Algorithms categories

Algorithm	Point Clustering	Incremental Track Insertion	Intersection Linking
Ahmed and Wenk [4]		✓	
Biagioni and Eriksson [8]	✓		
Cao and Krumm [11]		✓	
Davies et al. [16]	✓		
Edelkamp and Schrödl [17]	✓		
Ge et al. [23]	✓		
Karagiorgou and Pfoser [28]			✓

(a) Existing graph and tra-
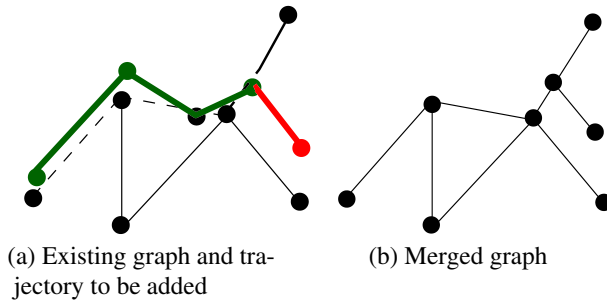jectory to be added

(b) Merged graph

Fig. 2 Incremental track insertion algorithm (images from presentation of[4])

2.2.1 Ahmed and Wenk [4]

The algorithm by Ahmed and Wenk [4] is a simple and practical incremental track insertion algorithm. It models the street map as an undirected graph and uses one parameter ε to model error associated with the GPS tracks and with the street width. The insertion of one track proceeds in three steps. The first step performs a partial map-matching of the track to the partially constructed map in order to identify matched portions and unmatched portions. Figure 2a gives an example of a track with its matched portions shown in dark green (Fréchet distance of the sub-curve to the map is less than ε) and its unmatched portions shown in red (Fréchet distance of the sub-curve to the map is greater than ε). This partial map-matching is based on a variant of the Fréchet distance. In the second step, the unmatched portions of the track are then inserted into the partially constructed map, which requires creating new vertices and creating and splitting edges. In a third step, the already existing edges in the map that are covered by the matched portions of the trajectory, are updated using a minimum-link algorithm to compute a new representative edge (cf. Fig. 2b). This last step is only needed to provide a guaranteed bound on the complexity of the output map; in the implementation of this algorithm that we use in Section 5, this last step has been omitted. Ahmed and Wenk also give theoretical quality guarantees for the output map computed by their algorithm, which include a one-to-one correspondence between well-separated "good" portions of the underlying map and the output map, with a guaranteed Fréchet distance between those portions.

2.2.2 Biagioni and Eriksson [8]

Biagioni and Eriksson [8] describe a point clustering-based algorithm that uses KDE methods. Their algorithm proceeds in using KDE with various thresholds to compute successive versions of a skeleton map. They annotate the map by performing a map-matching pass of
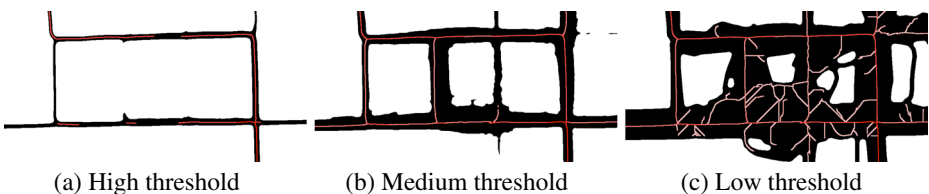

(a) High threshold

(b) Medium threshold

(c) Low threshold

Fig. 3 KDE-based map construction using threshold ranges (images from presentation of [8])

the input tracks with the skeleton map. Figure 3 gives three example stages of the skeleton construction process using high to low KDE thresholds. The skeleton is represented as an undirected graph, and in a final step the edges are replaced with one directed edge per direction, as well as turn lanes at vertices. Just as in [8], for evaluation purposes we use the final undirected skeleton as the output.

2.2.3 Cao and Krumm [11]

This incremental track insertion approach models the streep map as a directed graph with one directed edge per direction. It proceeds in two stages. In the first stage, simulation of physical attraction is used to modify the input tracks to group portions of the tracks that are similar together. This results in a cleaner data set in which track clusters are more pronounced and the two differently directed portions of a road segment are more separated. Then, this much cleaner data is used as the input for a fairly simple incremental track insertion algorithm. This algorithm makes local decisions based on distance and direction to insert an edge or vertex and either merge the vertex into an existing edge, or add a new edge and vertex.

Figure 4 gives a respective map construction example. The three trajectories of Fig. 4a are used to incrementally build the graph in Fig. 4b by (i) either merging nodes to existing nodes if the distances are small and the directions of the traces match (nodes in boxes), or (ii) by creating new nodes and edges otherwise (nodes in circles).

2.2.4 Davies et al. [16]

This is a classical KDE-based map construction algorithm. It first computes for each grid cell the density of tracks that pass through it (cf. the example of Fig. 5a). Then it computes the contour of the resulting bit map (Fig. 5b), and it uses the Voronoi diagram of the contour to compute a center line representation, followed by additional cleanup and assignment of edge directions (Fig. 5c). The final output is a directed graph in which each edge is labeled as directed or bidirected.

2.2.5 Edelkamp and Schrödl [17]

Edelkamp and Schrödl [17] were the first to propose a map construction approach based on the k-means method. Their point clustering algorithm creates road segments based on tracking data, represents the center line of the road using a fitted spline and performs lane
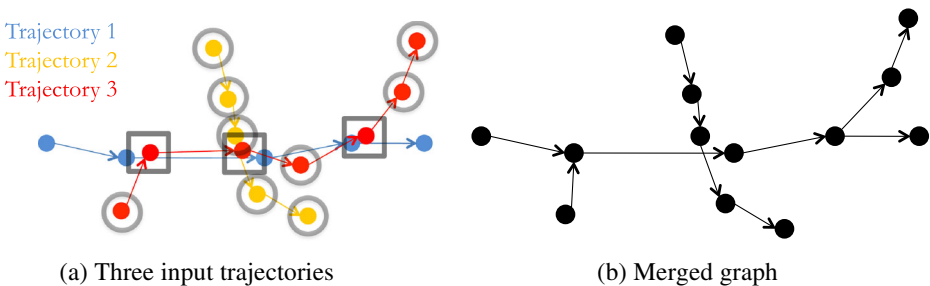

Trajectory 1
Trajectory 2
Trajectory 3

(a) Three input trajectories   (b) Merged graph

Fig. 4 The incremental track insertion algorithm - adapted from [11]

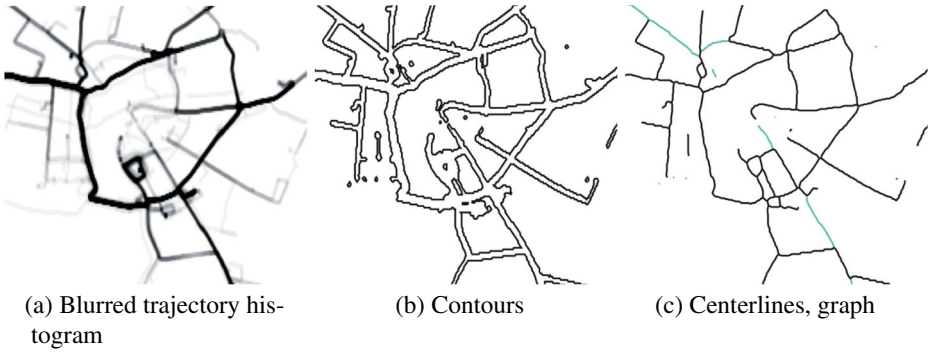| (a) Blurred trajectory histogram | (b) Contours | (c) Centerlines, graph |

Fig. 5 Clustering-based map construction algorithm (images from [16])

finding. The lanes are found by clustering tracks based on their distance from the road center line. The final graph is directed with a directed edge per lane (Fig. 6).

2.2.6 Ge et al. [23]

This algorithm is a point clustering approach that applies topological tools to extract the underlying undirected graph structure. The main idea of this algorithm is to decompose the input data set into sets, each corresponding to a single branch in the underlying graph. The authors assume that the input point set is densely sampled and their algorithm only needs a distance matrix or proximity graph of the point set as input. Then they define a function on the proximity graph, which assigns to every point in the graph its geodesic distance to an arbitrary base point. They employ the Reeb graph to model the connected components of the level set of the inverse of this function. Finally, there is a canonical way to measure importance of features in the Reeb graph, which allows them to easily simplify the resulting graph. Runtime guarantees are provided, as well as partial quality guarantees for correspondence of cycles. They compared street-maps as sets of cycles. If a cycle organization one map does not correspond to a cycle on another map, then obviously a street or a turn is missing on the second map. An embedding for the edges is then obtained by using a principal curve algorithm [29] that fits a curve to the points contributing to the edge. Figure 7 gives an example of a constructed graph based on a point cloud shown as light (yellow) dots.
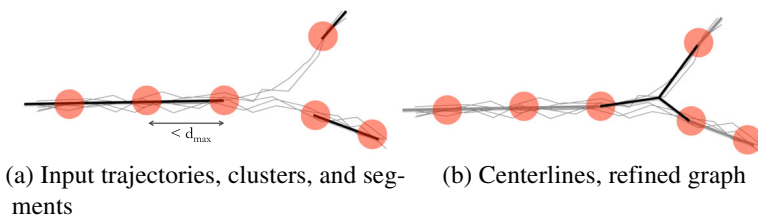

| (a) Input trajectories, clusters, and segments | (b) Centerlines, refined graph |

Fig. 6 Clustering-based map construction algorithm - adapted from [17]

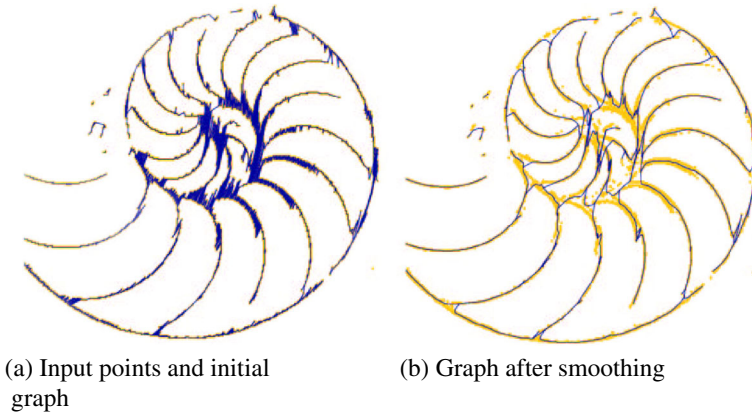(a) Input points and initial
graph

(b) Graph after smoothing

Fig. 7 Reeb graph based map construction (images from [23])

2.2.7 Karagiorgou and Pfoser [28]

This intersection-linking map construction algorithm is a heuristic approach that "bundles" trajectories around intersection nodes. It represents the street map as a directed graph in which each edge is labeled as directed or bidirected. The main contribution of *TraceBundle* algorithm is its methodology to derive intersection nodes. It relies on detecting changes in movement to cluster "similar" nodes. A node at which a change in direction and speed occurs is considered a turn indicator. Turn clusters are produced based on (i) spatial proximity and (ii) turn type. The centroid of a turn cluster then becomes an intersection node. Links between intersection nodes are derived by compacting the trajectories connecting the intersection nodes. Figure 8 visualizes the steps of the algorithm. Figure 8a shows the constructed intersection nodes as black stars. The constituting turn clusters are shown as x and o markers. Figure 8b shows the links between intersections nodes as black lines. The constituting trajectories are shown as dashed lines. The TraceBundle algorithm has three tunable
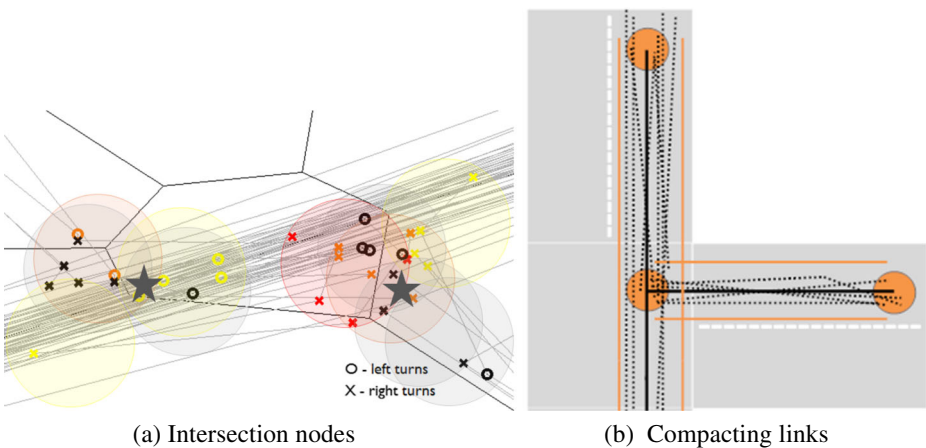

(a) Intersection nodes

(b) Compacting links

Fig. 8 The *TraceBundle* algorithm [28]

parameters, *angular difference*,*speed* and *spatial proximity*. Angular difference is the relative change of the vehicle direction measured in *degrees*. The speed threshold indicating turning vehicles is measured in km/h. This is am empirical maximum threshold to separate high-speed turns from turns at intersections. Spatial proximity distance threshold for clustering turn clusters into intersection nodes (in m).

3 Quality measures for map comparison

There are two key ingredients for evaluating the quality of a constructed map: (i) the availability of an adequate ground-truth map G as part of the benchmark data and (ii) a quality measure used to evaluate the similarity between the constructed map C and the ground-truth map G.

There are essentially two cases of what can be considered as a ground-truth map G. Ideally, G is the underlying map consisting of all streets, and only those streets, that have been traversed by the entities that generated the set of input tracks. If such a G was available, then a suitable quality measure would compare C to all of G and the ideal would be for C to equal G. However, in practice, it is hard to obtain an unbiased ground-truth map that exactly corresponds to the coverage of the tracking data. This non-trivial task has been addressed in the past by pruning the ground-truth either manually, by proximity to the tracking data, or by map-matching the tracking data to the map [7, 8, 28, 30]. By using graph topologies based on human judgment, or from the cropping behaviors of the different pruning algorithms, clearly all these approaches introduce an undesired bias.

Actually, it is much easier to obtain a ground-truth map that covers a superset of all the streets covered by the input tracks, e.g., street maps taken by proprietary vendors or OpenStreetMap. Therefore, if G is a superset, then the quality measure attempts to *partially* match C to G. Of course, another possible scenario is that C contains additional streets that are not present in either variation of G.

3.1 Related work

In the graph theory literature, there are various distance measures for comparing two abstract graphs that do not necessarily have a geometric embedding [15, 22, 37]. Most closely related to street map comparison are the subgraph isomorphism problem and the maximum common isomorphic subgraph problem, both of which are NP-complete. These, however, rely on one-to-one mappings of graphs or subgraphs, and they do not take any geometric embedding into account. Graph edit distance [21, 44] is a way to allow noise by seeking a sequence of edit operations to transform one graph into the other, however it is NP-hard as well. Cheong et al. [14] consider a graph edit distance for geometric graphs (embedded in two different coordinate systems, however), and also show that it is NP-hard to compute.

For comparing street maps, distance measures based on *point sets* and distance measures based on *sets of paths* have been proposed. *Point set-based distance measures* treat each graph as the set of points in the plane that is covered by all its vertices and edges. The idea is then to compute a distance between the two point sets. A straightforward distance measure for point sets are the directed and undirected Hausdorff distances [6]. The main drawback of such an approach is that it does not use the topological structure of the graph. Biagioni and Eriksson [7, 30], use two distance measures that essentially both use a variant of a partial one-to-one matching that is based on sampling both graphs densely. The two distance

measures compare the total number of matched sample points to the total number of sample points in the graph, thus providing a measure of how much of the graph has been matched. They do require though to have as input a ground-truth graph that closely resembles the underlying map and not a superset.

For *path-based distance measures* on the other hand, the underlying idea is to represent the graphs by sets of paths, and then to define a distance measure based on distances between the paths. This captures some of the topological information in the graphs, and paths are of importance for street maps in particular since the latter are often used for routing applications for which similar connectivity is desirable. Mondzech and Sester [31] use shortest paths to compare the suitability of two road networks for pedestrian navigation by considering basic properties such as respective path length. Karagiorgou and Pfoser [28] also use shortest paths, but to actually assess the similarity of road network graphs. Computing random sets of start and end nodes, the computed paths are compared using Discrete Fréchet distance and the Average Vertical distance. Using those sets of distances, a global network similarity measure is derived. In another effort, Ahmed and Wenk [3] cover the networks to be compared with paths of k link-length and map-match the paths to the other graph using the Fréchet distance. They are the first to introduce the concept of *local signature* to identify *how* and *where* two graphs differ.

3.2 Quality measures used for comparison

Here we give some more detail on the quality measures that we use in Section 5 to compare the different road network construction algorithms. Note that in our experiments the ground-truth G is an unmodified street map from OpenStreetMap and thus expected to be a superset of the underlying graph. We use the Directed Hausdorff distance [6], the path-based distance measure presented by Ahmed et al. [3], the distance measure based on shortest paths by Karagiorgou and Pfoser [28] and graph-sampling based distance measure by Biagioni and Eriksson [7]. The first two measures have not been used for comparative evaluations of road network constructions before.

3.2.1 Directed hausdorff distance [6]

The *directed Hausdorff distance* of two sets of points A, B is defined as $\overrightarrow{d}(A, B) = \max_{a \in A} \min_{b \in B} d(a, b)$. Here, $d(a, b)$ is usually the Euclidean distance between two points a and b. Intuitively, the directed Hausdorff distance assigns to every point in a its nearest neighbor $b \in B$ and takes the maximum of all distances between assigned points. In order to compare two graphs, we identify each graph as the set of points that is covered by all its vertices and edges. If the directed Hausdorff distance from graph C to graph G is at most ε, this means that for every point on any edge or vertex of C there is a point on G at distance at most ε. Or equivalently, every point of C is contained in the Minkowski sum of G with a disk of radius ε; the Minkowski sum intuitively "fattens" G by "drawing" each of its edges with a thick circular pen. This distance measure gives a notion about spatial distance for graphs. If C is the constructed graph and G is the ground-truth, the lower the distance from C to G, the closer the graph C to G.

3.2.2 Path-based distance [3]

The path-based map distance considers graphs as sets of paths. The distance between two sets of paths is then computed in the Hausdorff setting, while the Fréchet distance which is

a natural distance measure for curves that takes monotonicity and continuity into account, is used to compute the distance between two paths.

For curves f, g, the Fréchet distance is defined as

$$\delta_F(f, g) = \inf_{\alpha, \beta:[0,1] \to [0,1]} \max_{t \in [0,1]} d(f(\alpha(t)), g(\beta(t))), \tag{1}$$

where α, β range over continuous, surjective and non decreasing reparametrizations.

A common intuition is to explain it as the minimum leash length required such that a man and dog can walk on the two curves from beginning to end in a monotonic way.

Under this scope, let C and G be two planar geometric graphs, and let π_C be a set of paths generated from C, and π_G be a set of paths generated from G. The *path-based distance* is defined as:

$$\overrightarrow{d}_{C,G}(\pi_C, \pi_G) = \max_{p_C \in \pi_C} \min_{p_G \in \pi_G} \delta_F(p_C, p_G) \tag{2}$$

Ideally, π_C and π_G should be the set of all paths in C and G, which however has exponential size. In [3] they showed that $\overrightarrow{d}_{C,G}(\Pi_C, \Pi_G)$ can be approximated using $\overrightarrow{d}_{C,G}(\Pi_C^3, \Pi_G)$ in polynomial time using the map-matching algorithm of [5], under some assumptions on C. Here, Π_C is the set of all paths and Π_C^3 is the set of all link-3 paths of C. A link-k path consists of k "edges", where vertices of degree two in the graph are not counted as vertices. Using this asymmetric distance measure $\overrightarrow{d}_{C,G}(\Pi_C^k, \Pi_G)$, which can be computed in polynomial time for constant k, the following properties have been shown in [3], under some assumptions on C:

1. $k = 1$: For each edge in C, there is a path in G which is within Fréchet distance $\overrightarrow{d}_{C,G}(\Pi_C^1, \Pi_G)$.
2. $k = 2$: For each vertex v in C there is a vertex in G within bounded distance $\overrightarrow{d}_{C,G}(\Pi_C^2, \Pi_G)/\sin\frac{\theta}{2}$, where θ is the minimum incident angle at v between its adjacent edges.
3. $k = 3$: $\overrightarrow{d}_{C,G}(\Pi_C^3, \Pi_G)$ approximates $\overrightarrow{d}_{C,G}(\Pi_C, \Pi_G)$ within a factor of $1/\sin\frac{\theta}{2}$ if the vertices of C are reasonably well separated and have degree $\neq 3$. [1]

Similar to Directed Hausdorff distance, the lower the value of $\overrightarrow{d}_{C,G}(\Pi_C, \Pi_G)$ the more closely the constructed map C resembles the ground-truth map G.

The *local signature* of a vertex $v \in C$ is defined as $\Delta_v = \overrightarrow{d}_{C,G}(\Pi_{Cv}, \Pi_G)$ where Π_{Cv} is a set of paths that contains v. In a similar way, the local signature of an edge $e \in C$ is defined as $\Delta_e = \overrightarrow{d}_{C,G}(\Pi_{Ce}, \Pi_G)$ where Π_{Ce} is a set of paths that contains e. Based on the value of these signatures one can identify which vertices or edges are very similar and which are not.

3.2.3 Shortest path based distance [28]

Karagiorgou et al. [28] propose a measure that essentially samples each graph using random sets of shortest paths. First a ground-truth network is derived using the tracking data as a filter. Augmenting the geometry of the ground-truth network with buffer regions around its

---

[1]The degree assumption is only a technical requirement for the theoretical quality guarantees, and the authors have shown [3] that similar approximation guarantees appear to hold in practice as well.

edges and intersecting it with the tracking data, we obtain a reduced network graph that we use as a ground-truth network G. In, both, the constructed and the ground-truth networks C and G, respectively, we randomly select the same set of origin, destination nodes and compute the respective shortest paths in both networks. The geometric difference/similarity between these shortest paths is used to assess the similarity between C and G, and so the quality of the constructed network. The Discrete Fréchet distance and the Average Vertical distance are used to compare the shortest paths. The rationale for using this approach is that measuring the similarity for sets of paths instead of individual links allows one to better reason about the connectivity of the generated network. The more "similar" the shortest paths in the constructed network are to the ground-truth network, the higher also the quality of the network. The results of this shortest path comparison can be assessed by plotting the distance of all paths against each other, or by deriving parameter of the entire set of paths. We employ both approaches in our experiments as shown in the following.

3.2.4 Graph-sampling based distance [7]

Biagioni and Eriksson [7] introduce a graph-sampling based distance measure in order to evaluate geometry and topology of the constructed road networks represented by graphs. The main idea is as follows: starting from a random street location, explore the topology of the graphs by placing point samples on each graph during a graph traversal outward within a maximum radius. This produces two sets of locations, which are essentially spatial samples of a local graph neighborhood. These two point sets are compared using one-to-one bottleneck matching [19] and counting the unmatched points in each set. Note that the graph traversal can take directions of edges into account if desired; in [7] the authors apply this distance measure to directed graphs, while in [8] the authors apply it to undirected graphs.

The sampling process is repeated for several seed locations. For the bottleneck matching, the sample points on one graph can be considered as "marbles" and on the other graph as "holes". The algorithm considers one-to-one matchings between the point-sets and only allows points to be matched that are at distance at most a given threshold. Intuitively, if a marble lands close to a hole it falls in, marbles that are too far from a hole remain where they land, and holes with no marbles nearby remain empty. If one of the graphs is the ground truth, this difference represents the accuracy of the other graph. Counting the number of unmatched marbles and empty holes quantifies the accuracy of the generated road network with respect to the ground truth according to two metrics. The first metric is the proportion of spurious marbles, $spurious = spurious\_marbles / (spurious\_marbles + matched\_marbles)$ and the second is the proportion of missing locations (empty holes), where $missing = empty\_holes / (empty\_holes + matched\_holes)$.

To produce a combined performance measure from these two values, the well-known F-score is used, which is computed as follows:

$$F\text{-}score = 2 * \frac{precision * recall}{precision + recall} \tag{3}$$

where, $precision = 1 - spurious$ and $recall = 1 - missing$.

The higher the F-score, the closer the match. Sampling the graphs locally is an important aspect of this approach as it provides the ability to capture the connectivity of the graphs at a very detailed level, allowing the topological similarity to be measured. Repeated local sampling at randomly chosen locations yields an accurate view of local geometry and topology throughout the graph.

A modified version is used in [8] where the method ignores parts of the road network where no correspondence could be found between generated and ground-truth networks, for our experiments we used this modified version.

3.3 Comparison of distance measures

All the distance measures described in Section 3.2 capture different properties of graphs. Based on the desired type of similarity, different distance measures could be employed. For example, if one is interested in ensuring similar shortest paths in the two graphs, requiring that independent queries produce similar routes, then the shortest path based measure would be the prefect choice [28, 31] among all. If, however, one wants to know the spatial displacement between the two graphs without necessarily considering any kind of topology or path similarity, then the directed Hausdorff distance [6] would be the distance measure to choose.

On the other hand, the two distance measures described in [8] and [3] maximize the use of topology in comparing graphs. Using the concept of *local signature* described in [3] one can visualize the exact differences in graphs using any of these two measures. Figure 9 shows an example where the graph sampling based distance [8] fails to identify local differences (the dotted graph has a broken connection in the gray square region). As it samples small sub-graphs starting from a root location, it cannot capture this kind of broken connection when another connecting detour between the two parts is available in that small sub-graph. As the path-based distance [3] exploits every adjacency transition around a vertex, it verifies all connectivities.

Among these four measures only the graph sampling based distance [8] ensures one-to-one correspondence. So, if one of the graphs has missing streets or extra edges, that is reflected in the overall score as well as in the local signatures.

4 Datasets

A basic means for assessing map construction algorithms is the underlying dataset comprising vehicle trajectories and ground-truth map datasets. The datasets are in a projected coordinate system (UTM, GGRS87). All the visualizations of the datasets are also available on the http://mapconstruction.org Web site. The statistics of the datasets are provided in Table 2.

Our experiments use several tracking datasets from different cities (Fig. 10). While other publicly available GPS-based vehicle tracking datasets exist, e.g., GeoLife [46] and Open-StreetMap GPX track data [35], the selected range covers the various types of existing

Fig. 9 Graph *G* (*dotted edges*) overlayed on *H* (*gray*). *G* and *H* differs in the shaded squared region. The distance measure in [8] fails to capture the broken connection in *G*, as there is always detour available to reach every edge and sample it
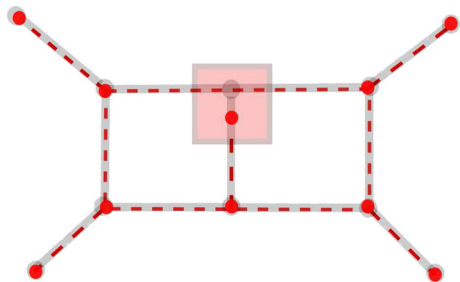
Table 2 Statistics for datasets used

Tracking Data	Trajectories	Sampling rate (s)	Trajectory length (km)	Speed (km/h)	OSM Network	Vertices	Edges	Length (km)	Area (km²)
Athens large	120	30.14	6,781	20.16	Athens large	32,212	39,699	2,000	12 × 14
Athens small	129	34.07	443	19.55	Athens small	2,694	3,436	193	2.6 × 6
Berlin	26,831	41.98	41,116	35.23	Berlin	5,894	6,839	360	6 × 6
Chicago	889	3.61	2869	33.14	Chicago	9,429	11,801	61	7 × 4.5

datasets produced by different types of vehicles, at varying sampling rates and representing different network sizes. In the experiments, it becomes obvious that map construction algorithms produce less accurate street maps for less frequent sampling rates and large scaled networks.

The *Athens large* dataset consists of 511 trajectories with a total length of 6,781 *km* (average: 13.27 *km* and standard deviation: 10.79 *km*) obtained from school buses covering an area of 12 *km* × 14 *km*; the tracks range from 32 to 80 position samples, with a sampling rate of 20 *s* to 30 *s* (average: 30.14 *s* and standard deviation: 24.77 *s*) and an average speed of 20.16 *km/h*. The *Athens small* dataset consists of 129 tracks with a total length of 443 *km* (average: 3.82 *km* and standard deviation: 1.45 *km*) obtained from school buses

(a) *Athens large*

(b) *Athens small*

(c) *Berlin*

(d) *Chicago*
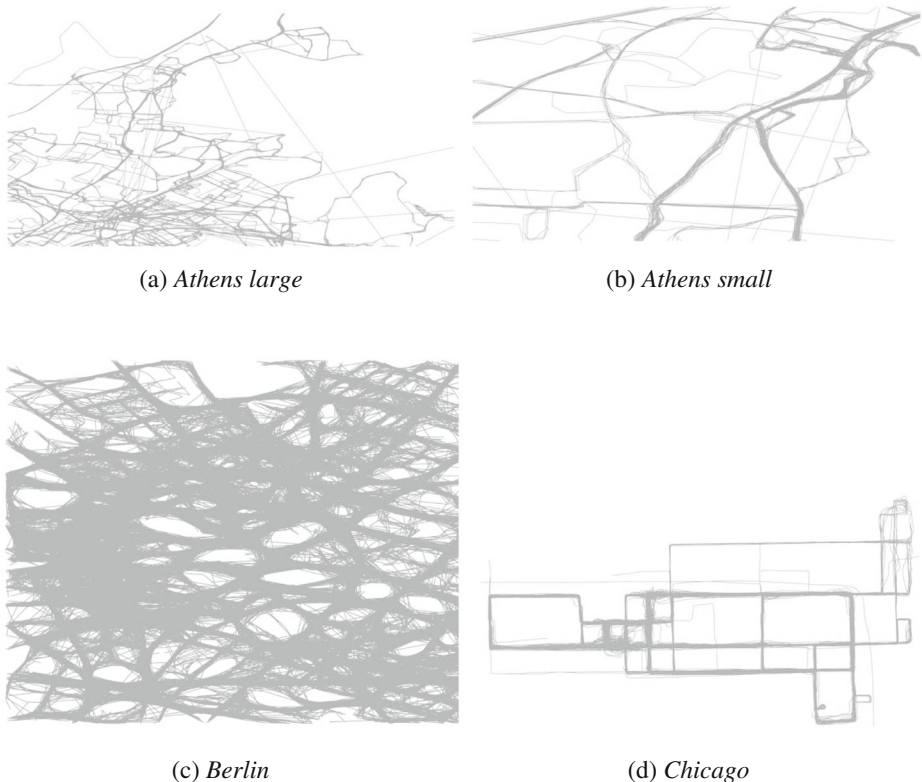
Fig. 10 Tracking data

covering an area of 2.6 $km \times 6\ km$; the tracks range from 13 to 47 position samples, with a sampling rate of 20's to 30 s (average: 34.07 s and standard deviation: 31.92 s) and an average speed of 19.55 km/h. The *Berlin* dataset consists of 26,831 tracks with a total length of 41,116 km (average: 1.53 km and standard deviation: 634.51 m) obtained from a taxi fleet covering an area of 6 $km \times 6\ km$; the tracks comprise from 22 up to 58 position samples, with a sampling rate of 15 s to 127 s (average: 41.98 s and standard deviation: 38.70 s) and an average speed of 35.23 km/h. The *Chicago* dataset [7, 8] consists of 889 tracks with a total length of 2,869 km (average: 3.22 km and standard deviation: 894.28 m) obtained from university shuttle buses covering an area of 7 $km \times 4.5\ km$; the tracks range from 100 to 363 position samples, with a sampling rate of 1 s to 29 s (average: 3.61 s and standard deviation: 3.67 s) and an average speed of 33.14 km/h.

For all cases, we consider as ground-truth map data the corresponding OpenStreetMap excerpt.

5 Experiments

What follows is a description of the map construction experiments that were conducted for the range of algorithms, datasets and evaluation measures, with the scope to *assess the quality of the constructed maps*. In the experiments, we used C, Java, Python, and Matlab implementations of seven map construction algorithms. For evaluation purposes, we used only the underlying undirected graph structures computed by the algorithms, and we dropped directions and any other additional annotations. The experiments for six algorithms have been performed by the authors and the implementations have been made available on the http://mapconstruction.org Web site. The authors of [23] performed the experiments themselves, since we did not have access to their implementation. Given the implementations, (i) their difference in code base, (ii) their scope, i.e., to construct small-scale maps from GPS trajectories, and (iii) their quality, i.e., all are academic prototypes, we did not assess the characteristics of the algorithms themselves by means of a performance study or theoretical analysis. However, to at least give an impression of their running times, for the *Chicago* dataset the running times of the algorithms range from 10 min to 20 h. For the larger *Berlin* dataset, the running times range from 2 h to 4 *days*. Given the quality of the implementations, another problem we encountered was that some algorithms could not cope with the size of the input dataset (trajectories) resulting in runtime crashes attributed to memory consumption and leaks. Hence, not all algorithms could be tested on the large datasets and results for all algorithms are only available for the smaller datasets, i.e., *Athens small* and *Chicago*.

5.1 Constructed Maps

What follows is an initial overview of the experiments in terms of constructed maps and the respective result quality. Figure 11 illustrates the ground-truth map (light gray) and the generated maps (black) for the small *Chicago* dataset. On larger datasets, i.e., *Athens large* and *Berlin*, we ran the algorithms described in Sections 2.2.1, 2.2.6 and 2.2.7. Figure 12 illustrates the ground-truth map (light gray) and the generated maps (black) for the case of the larger *Berlin* dataset.

Each of the algorithms uses different parameter settings. The values of clustering parameters for Ahmed and Wenk [4] are: 180, 90, 170 and 80 meters for *Athens large*, *Athens small*, *Berlin* and *Chicago*, respectively; and for Cao the value is 20 m [11]. The minimum
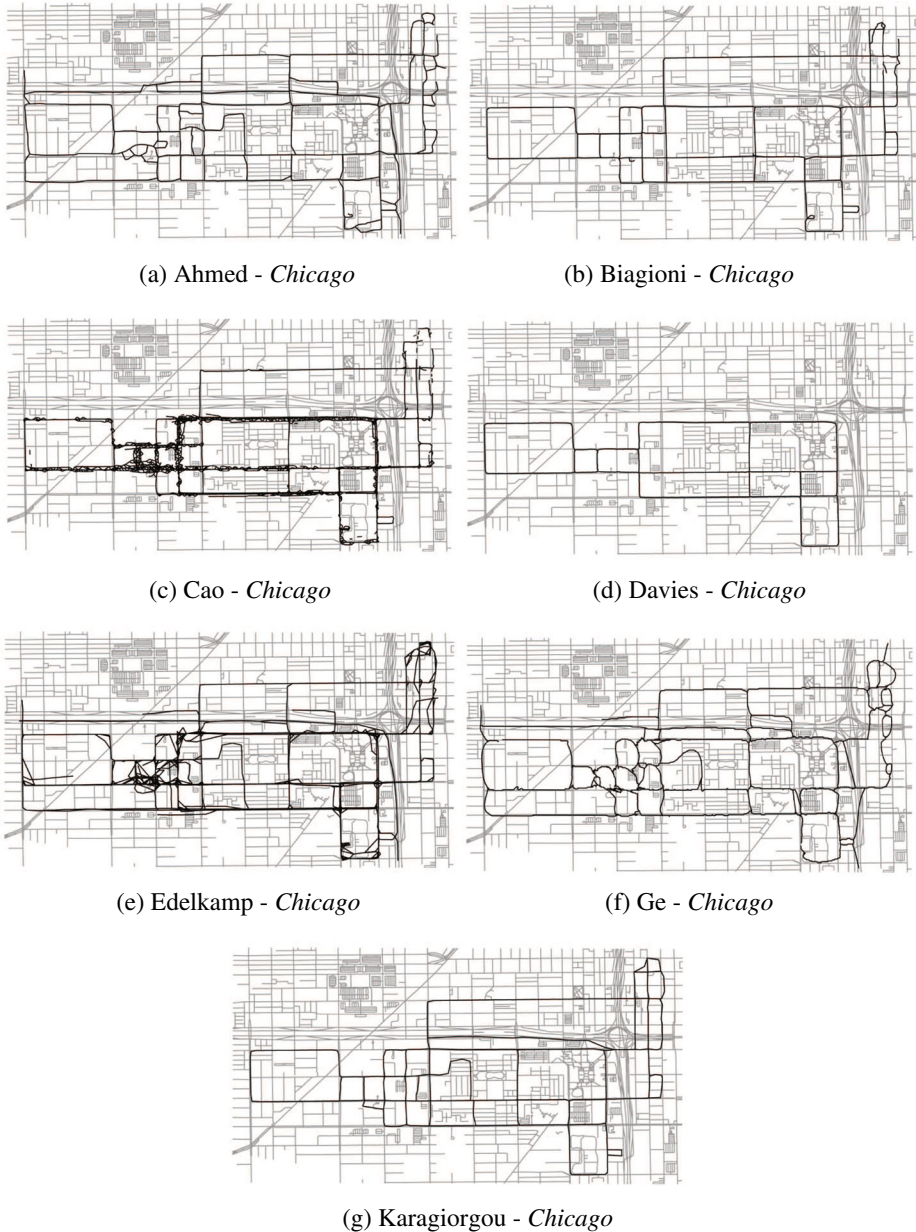
(a) Ahmed - *Chicago*

(b) Biagioni - *Chicago*

(c) Cao - *Chicago*

(d) Davies - *Chicago*

(e) Edelkamp - *Chicago*

(f) Ge - *Chicago*

(g) Karagiorgou - *Chicago*

Fig. 11 Constructed maps (*in black*) overlayed on ground-truth map (*in gray*) (*small dataset*)

bearing difference between two streets at any intersection is 45° [11]. The respective parameters of *intra cluster distance threshold* (minimum separation between streets) and *intra cluster bearing threshold* (minimum angular difference between two streets at intersection) for the $k-means$ algorithm is Edelkamp 50 *m* and 45° [17]. The respective parameters of *minimum density threshold* for the $KDE-$based algorithms are Biagioni 50 *m* [8] and
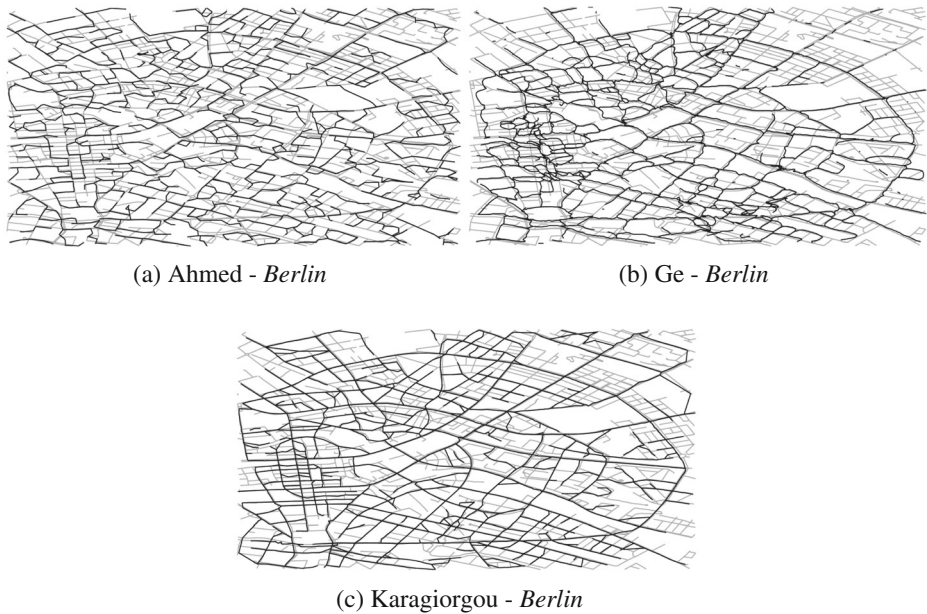
(a) Ahmed - *Berlin*   (b) Ge - *Berlin*

(c) Karagiorgou - *Berlin*

Fig. 12 Constructed maps (*in black*) overlayed on ground-truth map (*in gray*) (*large dataset*)

Davies 16 *m* [16]. For Karagiorgou and Pfoser [28] the values of *direction*, *speed* and *proximity* to extract intersection nodes and to merge trajectories into links are 15°, 40 *km/h* and 25 *m* accordingly. We evaluated all constructed maps using the distance measures described in Section 3.2.

A summary of the complexities of the constructed maps is shown in Table 3. Here, the number of vertices includes vertices of degree two (which may lie on a polygonal curve describing a single edge), the number of edges refers to the number of undirected line segments between these vertices, and the total length refers to the total length of all undirected line segments. It appears that the *point clustering algorithms based on kernel density estimation* such as Biagioni et al. [7, 8] and Davies et al. [16] produce maps with lower complexity (fewer number of vertices and edges) but often *fail to reconstruct streets that are not traversed frequently enough* by the input tracks. In particular, the maps reconstructed by Davies et al.'s algorithm are very small. On the other hand, the algorithm by Ge et al. [23] subsample all tracks to create a much denser output set, hence the complexity of their constructed maps is always higher.

Map construction algorithms based on *incremental track insertion*, such as Ahmed et al. [4] and Cao et al. [11] fail to cluster tracks together when the variability and error associated with the input tracks is large. As a result, the constructed street maps contain *multiple edges for a single street*, which implies larger values in the total edge length column in Table 3.

Several examples of generated maps are shown in Figs. 11 and 12. Since not all algorithms produced results for all maps, we showcase examples of the smaller *Chicago* map in Fig. 11. It can be clearly seen that the coverage and quality of the constructed map varies considerably. Three examples for the *Berlin* map are also given in Fig. 12. More examples can be found on the http://mapconstruction.org Web site.

Table 3 Complexities of the generated maps

Generated Map	# Vertices	# Edges	Length (km)
Athens large			
Ahmed	7067	7960	1358
Ge	20774	21626	9740
Karagiorgou	6584	5280	252
Athens small			
Ahmed	344	378	35
Biagioni	391	398	22
Cao	20	14	3
Davies	209	227	2
Edelkamp	526	1037	197
Ge	1936	1993	23
Karagiorgou	660	637	35
Berlin			
Ahmed	1322	1567	164
Ge	15450	16136	183
Karagiorgou	2542	2262	161
Chicago			
Ahmed	1195	1286	34
Biagioni	303	322	24
Cao	2092	2948	78
Davies	1277	1310	14
Edelkamp	828	1247	83
Ge	5893	6672	37
Karagiorgou	596	558	26

5.2 Path-based and hausdorff distance

For the *path-based distance measure* we generated all paths of link-length 3 for each generated map. For each path, we computed the Fréchet distance between the path and the ground-truth map. We then computed the minimum, maximum, median, average of all the obtained distances. We also computed the *d %-distance*, as the maximum of the distances after removing the *d* % largest distances ("outliers"). For the *Directed Hausdorff distance*, we computed all link-length 1 paths and computed the Directed Hausdorff distance of the union of all edges to the ground-truth map. Our results are summarized in Table 4. In the case of *Athens small*, the Cao algorithm produced a very small map and thus it was not possible to perform a quantitative evaluation.

The maps reconstructed using the algorithms by Karagiorgou et al. [28] and by Biagioni et al. [7, 8] generally have a better path-based distance than the others. Note that Davies et al.'s [16] map is unusually small for the *Athens small* dataset. Their idea of averaging trajectories, or computing skeletons, however, seems to help to improve the quality of the edges of the produced map.

For further analysis of the results, we selected the *Chicago* dataset as all map construction algorithms produced results for it. From Table 4 one can see that the path-based distance

Table 4 Path-Based and Directed Hausdorff distance measure evaluation

Generated Map	Path based distance (m)								Directed Hausdorff distance (m)							
	min	max	median	avg	2 %	5 %	10 %	15 %	min	max	median	avg	2 %	5 %	10 %	15 %
Athens large																
Ahmed	7	849	70	85	250	164	132	114	1	269	30	33	84	67	56	50
Ge	7	956	76	90	237	188	150	116	1	295	35	37	95	74	59	52
Karagiorgou	2	175	25	32	109	80	63	53	1	200	10	13	46	35	26	22
Athens small																
Ahmed	9	224	45	52	101	101	81	72	1	82	25	26	82	54	46	40
Biagioni	5	73	35	36	67	66	61	57	3	74	19	20	47	43	31	31
Cao									5	25	13	13	25	25	25	22
Davies	4	38	11	11	38	18	14	14	2	13	7	6	13	13	13	11
Edelkamp	2	229	36	39	89	72	68	61	1	86	18	21	63	50	42	37
Ge	19	251	52	59	142	113	89	76	3	81	21	23	80	59	39	35
Karagiorgou	7	229	32	38	113	68	59	57	2	84	14	17	54	40	33	30
Berlin																
Ahmed	9	540	66	74	207	147	120	107	1	219	30	33	95	70	60	53
Ge	13	808	65	75	214	157	117	103	4	562	36	37	73	62	55	51
Karagiorgou	4	306	28	37	120	85	65	52	1	232	14	18	59	42	34	30
Chicago																
Ahmed	7	201	35	42	127	100	85	76	1	81	14	19	72	59	43	35
Biagioni	3	71	15	18	71	38	27	26	2	53	9	11	29	25	23	17
Cao	1	126	24	27	79	61	49	42	1	78	9	12	44	35	28	25
Davies	2	92	12	14	57	24	22	21	2	20	8	7	20	14	13	12
Edelkamp	1	205	29	37	99	84	72	66	1	93	8	13	57	48	35	25
Ge	18	346	50	56	158	126	95	75	7	72	26	28	64	61	53	46
Karagiorgou	3	89	15	23	72	72	65	51	1	48	7	8	41	23	15	13

and the Directed Hausdorff distance are smaller for the generated maps by Biagioni, Davies and Karagiorgou compared to map generated using other algorithms. A visual inspection of the maps in Fig. 11 justifies the result. Note that Davies et al.'s [16] map is comparatively smaller than the other (cf. Table 3). Although the algorithms by Ahmed et al. and by Ge et al. produce maps with good coverage, their path-based distances are larger since they employ less aggressive averaging techniques that would help cope with noise in the input tracks.

To illustrate the appropriateness of the path-based distance, consider the path in Fig. 13 from the map generated by Biagioni et al. This is an example where the Fréchet-based distance measure is more effective than any point-based measure. As Fréchet distance ensures continuous mapping, the whole path needs to be matched with the bottom horizontal edge of the ground-truth map. The Fréchet distance for this path is 71 *m*. For the same path, the Hausdorff distance is 53 *m*, as this only requires for each point on the path to have a point on the graph close-by. So, to evaluate the connectivity of a map, the Fréchet distance is a more suitable distance measure than any point-based measure.
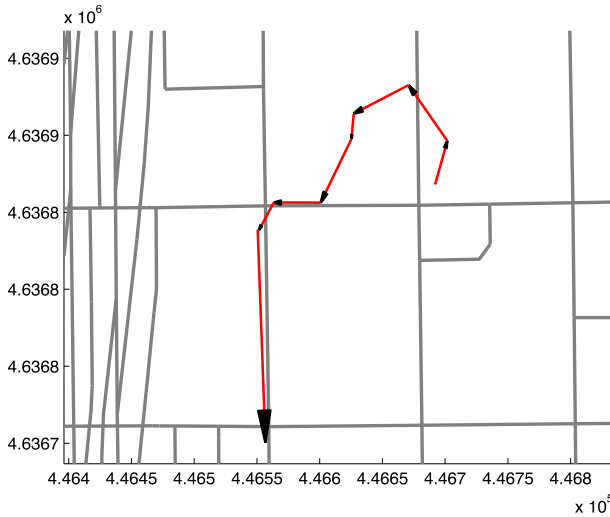
Fig. 13 A path with Fréchet distance greater than Hausdorff distance

In addition, if desired one can discard outliers by computing the d %-distance. Figure 14 shows the distribution of both the path-based measure and the Directed Hausdorff distance for Biagioni et al. In both cases, a very small number of paths have the maximum distance, and the distances for most of the paths are distributed within a small range. Removing only 5 % of the outliers (largest) brings the path-based distance from 71 m (max) to 38 m and the Directed Hausdorff distance from 53 m (max) to 25 m. Figure 15 shows edges of maps with smaller distances in lighter shades and larger distances in darker shades. Such visual representation helps to identify areas in the map that have higher distance to the ground-truth map.

5.3 Shortest path based measure

Another means to compare the constructed maps is the shortest path based distance. For each city, we computed a set of 500 random shortest paths with origin and destination nodes
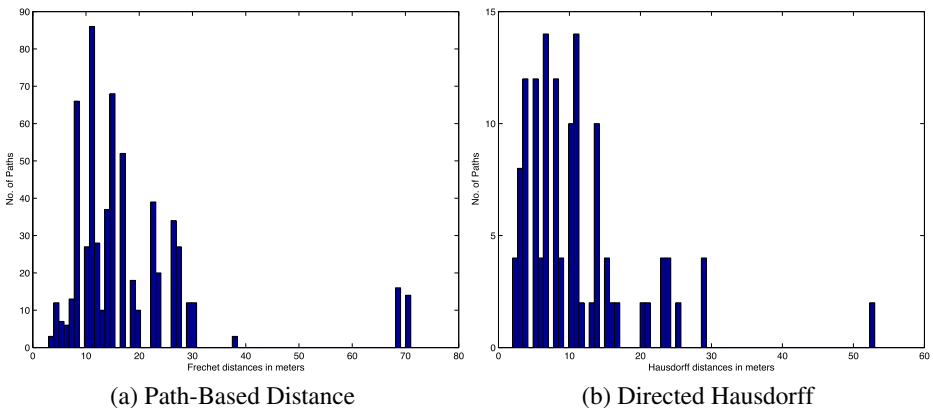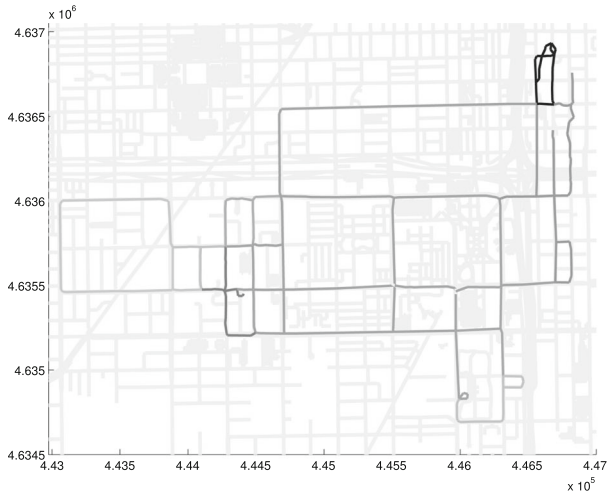

(a) Path-Based Distance

(b) Directed Hausdorff

Fig. 14 Distributions of individual path distances (Biagioni alg. - *Chicago*)

(a) Biagioni et al. Edges in lighter shades inidicate smaller distances (3*m* being the smallest) and darker shades indicate larger distances(71*m* being the largest).

(b) Ahmed et al. Edges in lighter shades indicate smaller distances (7*m* being the smallest) and darker shades indicate larger distances(201*m* being the largest).

Fig. 15 Reconstructed graph overlayed on ground-truth map (*light gray*). Based on link-length 3 paths, edges in lighter shades has smaller distance and darker shades has larger distance

uniformly distributed over the maps and compared the paths using the Discrete Fréchet and Average Vertical distance measure.

A first impression on how different constructed maps affect such paths is given in Fig. 16. Given a specific origin and destination for the *Chicago* map, the shortest path has length 3.66 *km* in the ground-truth map (black dotted line). The computed shortest path for the

(a) Ahmed

(b) Biagioni

(c) Cao

(d) Davies
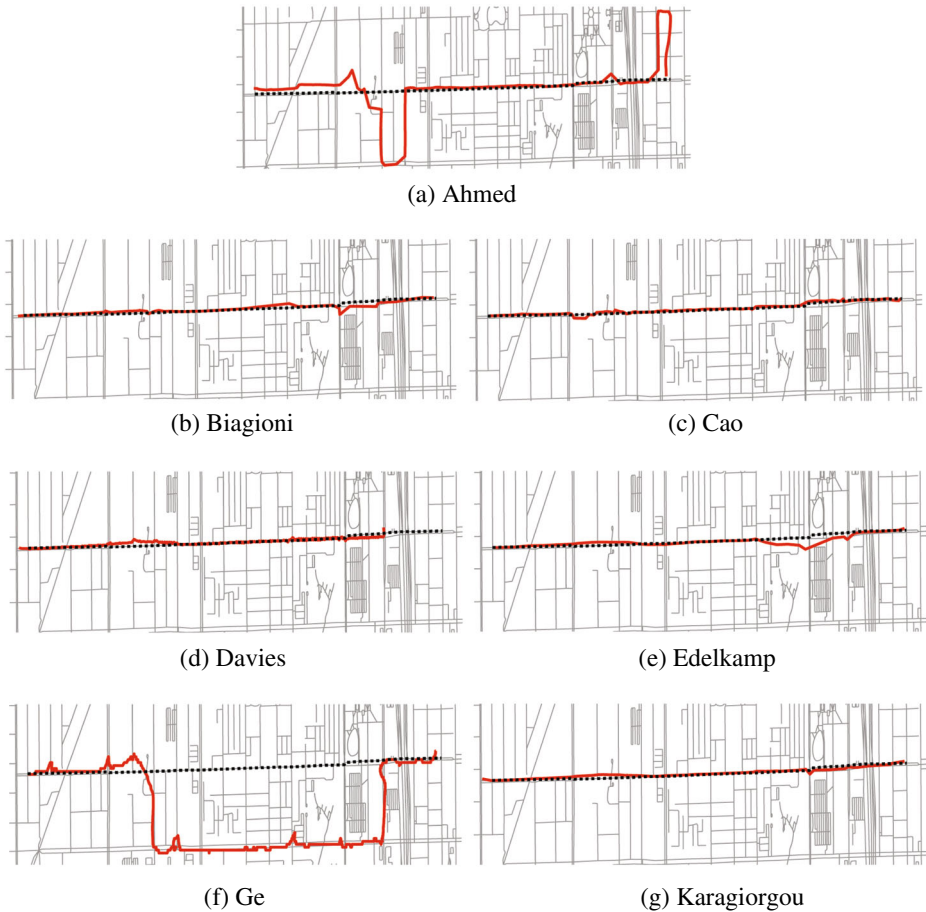

(e) Edelkamp

(f) Ge

(g) Karagiorgou

Fig. 16 Examples of shortest paths for the *Chicago* dataset

map generated by each algorithm is shown in red. In the map generated by Ahmed et al.'s algorithm the shortest path has length 4.67 *km* (a Discrete Fréchet distance with respect to the ground-truth map of 65 *m* and an Average Vertical distance of 21 *m*). The respective results for the other algorithms are Biagioni 3.71 *km* (36 *m*, 5 *m*), Cao 3.76 *km* (24 *m*, 6 *m*), Davies 3.39 *km* (35 *m*, 4 *m*), Edelkamp 3.64 *km*, (26 *m*, 8 *m*), Ge 7.33 *km*, (174 *m*, 98 *m*), and Karagiorgou 3.73 *km* (21 *m*, 5 *m*). For most algorithms the resulting paths have a small distance to the shortest path in the ground-truth map. However, in the case of Ahmed (Fig. 16a) and Ge (Fig. 16f), due to significant differences in the generated map, different shortest paths have been computed that have a larger distance to the shortest path in the ground-truth map. This result is in line with the path-based measure of Section 5.2, where also Biagioni, Davies and Karagiorgou produced the best constructed maps.

Figure 17a and b show the Discrete Fréchet distance and the Average Vertical distance measures for each of the 500 paths per algorithm for the *Athens large* map. The paths are ordered by increasing distance of the shortest path length with respect to the ground-truth map. Some paths could not be computed for some maps due to connec-
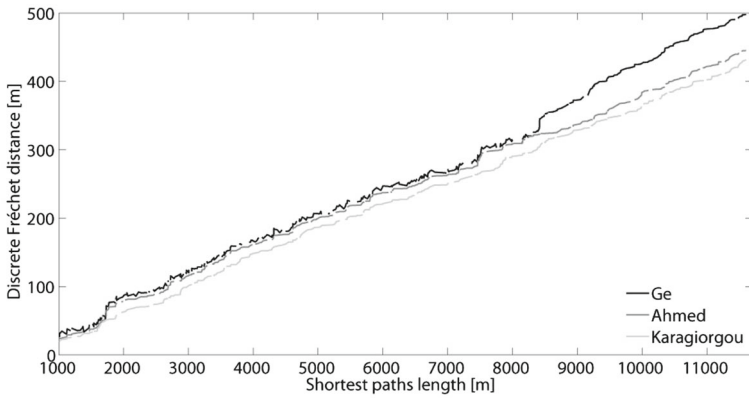
(a) Discrete Fréchet distance - *Athens large*
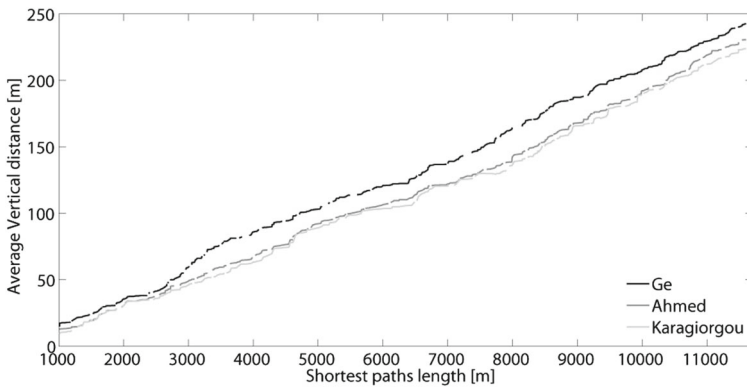
(b) Average Vertical distance - *Athens large*

Fig. 17 Map comparison

tivity problems (missing links). Some other paths experience greater distance measures due to spatial accuracy problems. The graph shows that some algorithms produce maps which resemble the actual map more closely, as assessed by this shortest path sampling approach.

Finally, the shortest path based evaluation is summarized in Table 5. The first column shows the percentage (%) of shortest paths that in each case could be computed, i.e., an algorithm might find an accurate, but small map. The second and the third column show the two different distance measures used to compare the resulting paths. The fourth column gives some statistics with respect to the computed shortest paths. Considering the example of *Berlin* and here the Ahmed algorithm result in Table 5, this algorithm produces a map that in turn generates paths that have a min, max, and avg. Discrete Fréchet distance of 21 *m*, 469 *m*, and 192 *m*, respectively. An aspect not captured by these distances are missing paths due to *limited map coverage*. Consider the case of Davies for *Chicago* and Cao for *Athens small*. In both cases, the distance measures suggest good map quality. However, in both cases the constructed map has a small coverage, as only 92.6 % and 7.0 % of the 500 total paths were computed. In this evaluation, Karagiorgou produces maps that

Table 5 Shortest path measure evaluation summary

Generated Map	Found (%)	Discrete Fréchet distance dist. (m)				Average Vertical dist. (m)				Shortest path dist. (km)			
		min	max	avg	stddev	min	max	avg	stddev	min	max	avg	stddev
Athens large													
Ahmed	92.6	23	445	137	103	12	230	106	62	1.12	11.84	6.93	2.92
Ge	92.8	25	497	149	112	14	241	120	65	1.47	11.91	7.13	3.18
Karagiorgou	94.2	19	432	125	96	9	225	98	58	1.01	11.62	6.84	2.86
Athens small													
Ahmed	97.6	13	234	96	62	6	91	38	24	1.28	5.72	3.11	1.84
Biagioni	94.2	7	214	84	50	4	80	28	21	0.79	5.23	2.97	1.41
Cao	7.0	7	26	10	11	4	13	6	5	0.17	0.31	0.22	0.21
Davies	22.6	9	258	102	69	5	81	31	22	0.85	5.25	2.99	1.47
Edelkamp	97.2	15	228	97	64	6	93	40	26	0.93	5.29	3.02	1.51
Ge	93.4	21	290	123	75	11	127	63	33	1.43	5.93	3.41	1.92
Karagiorgou	96.8	7	212	81	48	3	81	27	20	0.78	5.21	2.95	1.39
Berlin													
Ahmed	93.2	21	469	191	123	12	231	121	63	1.56	5.88	3.49	1.96
Ge	92.4	25	475	194	128	15	236	127	64	1.85	5.93	3.84	2.03
Karagiorgou	93.8	18	428	183	112	8	209	106	58	1.32	5.67	3.27	1.84
Chicago													
Ahmed	99.8	13	208	97	56	6	92	43	19	1.21	6.95	4.45	2.04
Biagioni	98.6	4	98	40	27	2	49	20	13	0.89	6.03	3.76	1.57
Cao	99.2	7	131	67	34	4	76	41	17	1.02	6.87	3.94	1.84
Davies	92.6	5	97	41	27	3	51	23	15	0.93	6.08	3.88	1.66
Edelkamp	99.0	12	211	98	58	5	89	41	18	1.19	6.88	4.32	1.97
Ge	99.8	19	241	127	63	8	94	49	22	1.58	6.98	4.69	2.25
Karagiorgou	99.2	4	103	41	28	2	50	21	14	0.90	6.05	3.82	1.59

have both good coverage and high path similarity (cf. dark-shaded entry for Berlin - good coverage and small distance measure indicating similar paths between constructed and ground-truth map).

Overall, shortest path sampling provides an effective means for assessing the quality of constructed maps as it not only considers *similarity*, but also the *coverage of the map*.

5.4 Graph-Sampling Based Distance

For this measure we use the source code obtained from the authors of [7]. We modified the code to use Euclidean distance as our data uses projected coordinate systems. The method that computes this measure has four parameters: (i) *sampling density*, how densely the map should be sampled (marbles for generated map and holes for ground-truth map), we use 5 *m*, (ii) *matched distance*, the maximum distance between a matched marble-hole pair, we vary

this distance from 10 to 120 m, (iii) *maximum distance from root*, the maximum distance from randomly selected start location one will explore, we use 300 m and (iv) *number of runs*, number of start locations to consider, we use 1000. According to the implementation a start location is selected on the *2D-plane*, then a corresponding set of start locations on each graph is selected based on *matched distance* and finally a union of all edges traversed from all corresponding start locations are used to produced set of sample points. A larger *matched distance* might yield a larger number of sample points. To make our comparison of all generated maps consistent, we generated a sequence of random locations for each dataset and used the first 1,000 locations from the same sequence for each algorithm for which both maps (ground-truth and generated) had correspondences within *matched distance*. When two maps are very similar, they should have very few unmatched marbles and holes, which implies the precision, recall and F-score values should be very close to 1. In our case, as we used a superset of the ground-truth map, there should be a large number of unmatched holes, which implies lower recall and F-score values than in [7], but still the relative comparison of F-score values should provide an idea of whether an algorithm performs better than another.

We chose matching distance thresholds upto 120 m to make it consistent with the error associated with input data. As mentioned in [7] some area in the Chicago dataset region traces show consistent errors well over 100 m.

Figure 18 shows F-score values for the *Chicago* dataset for different generated maps. As our ground-truth is essentially a superset of the actual ground-truth represented by the tracking dataset, a larger matching distance creates unexpected results for algorithms that generate extra edges and vertices. For example, in the case of Cao and Edelkamp for *Chicago*, the precision is low as there will be lots of unmatched marbles (cf. entry for Cao and Edelkamp for *Chicago* in Table 6). However, a larger matching distance decreases the number of unmatched marbles by matching these with available holes that probably are not part of the actual ground-truth. A higher recall value yields a higher F-score, which does not necessarily reflect better-quality maps (cf. Figs. 11 and 12).

In Fig. 18 we also see the performance based on F-score declines for Biagioni, Davies and Karagiorgou as the matching distance threshold increases. In investigating the reason of this unexpected behavior, we found that although precision increases with matching distances the recall declines for these three algorithms; and smaller recall indicates larger
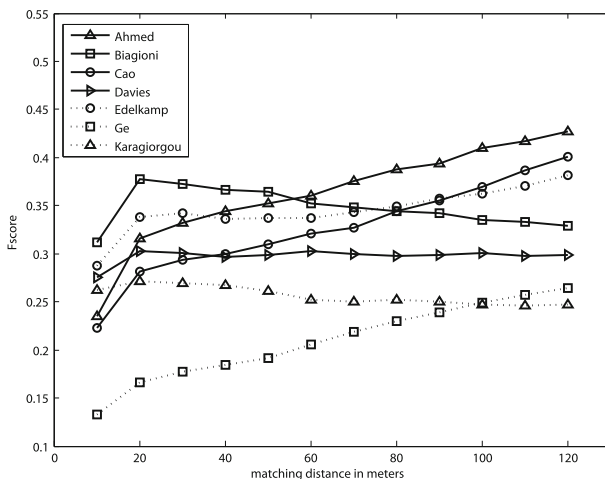

Fig. 18 Comparison of F-scores - *Chicago*

Table 6 Precisions for varying *matched distance*

Generated Map	Precision Value (for *matched distance* 10 *m*, 40 *m*, 70 *m*, 100 *m*)			
	10 *m*	40 *m*	70 *m*	100 *m*
Athens large				
Ahmed	0.216	0.407	0.497	0.591
Ge	0.149	0.368	0.507	0.635
Karagiorgou	0.394	0.559	0.630	0.711
Athens small				
Ahmed	0.265	0.442	0.503	0.579
Biagioni	0.450	0.586	0.662	0.727
Cao	0.415	.691	0.722	0.810
Davies	0.439	0.574	0.617	0.670
Edelkamp	0.106	0.156	0.197	0.232
Ge	0.409	0.527	0.624	0.708
Karagiorgou	0.343	0.489	0.561	0.647
Berlin				
Ahmed	0.123	0.326	0.422	0.485
Ge	0.142	0.457	0.534	0.584
Karagiorgou	0.294	0.590	0.633	0.649
Chicago				
Ahmed	0.312	0.563	0.658	0.738
Biagioni	0.491	0.699	0.730	0.775
Cao	0.209	0.321	0.376	0.456
Davies	0.488	0.650	0.690	0.739
Edelkamp	0.334	0.431	0.473	0.541
Ge	0.306	0.487	0.565	0.645
Karagiorgou	0.602	0.740	0.751	0.801

number of unmatched sample points on ground-truth (empty holes). Figure 11 and Table 3 show these three algorithms reconstruct less streets than others, which means they produce a smaller number of marbles to match with a larger number of holes. We explained earlier in this subsection how the total number of holes might increase with a choice of a larger *matched distance*.

Hence, in Table 6 we are ignoring F-score and recall values and *showcase only precision values*. According to precision values, the algorithms by Biagioni, Davies and Karagiorgou perform best for dataset *Chicago*, which is consistent with our findings using the other three distance measures.

5.5 Summary

The best way to characterize the constructed maps is in terms of coverage and accuracy. Here, it appears that KDE-based point clustering algorithms such as Biagioni and Davies

produce maps with lower complexity (fewer number of vertices and edges) and often fail to reconstruct streets that are not traversed frequently enough by the input tracks. On the other hand, the algorithm by Ge subsample all tracks to create a much denser output set, hence the complexity of their constructed maps is always higher. A similar observation can be made for algorithms based on incremental track insertion, such as the Ahmed and Cao algorithms. They fail to cluster tracks together when the variability and error associated with the input tracks is large. As a result, the constructed street maps contain multiple edges for a single street, which implies a larger constructed, but not necessarily more accurate road network.

In terms of map quality and accuracy, the maps reconstructed using the algorithms by Karagiorgou, Davis, and Biagioni generally have smallest path-based and Directed Hausdorff distances and their constructed maps can be considered more accurate. Although the algorithms by Ahmed and Ge produce maps with good coverage and provide quality guarantees, their path-based distances are larger, since they employ less aggressive averaging techniques that would help cope with noise in the input tracks. In an effort to assess both accuracy and coverage, the shortest path based measure shows for the cases of Davies and *Chicago* and Cao and *Athens small* good map quality, but at the same only limited coverage. In this evaluation, Karagiorgou produces maps that have both good coverage and high path similarity.

An overall observation to be made based on our experiments is that map construction algorithms tend to produce either accurate maps, or maps with good coverage, but not both. The algorithm of Karagiorgou however seems to be a good compromise, in that it produces maps of good coverage and accuracy at the same time.

6 Conclusions

This survey has considered the active field of road network construction and has considered a variety of such construction algorithms. In the past, the lack of benchmark data and quantitative evaluation methods has hindered a cross-comparison between algorithms. In this paper, the contribution of benchmark data sets and code for road network construction algorithms and evaluation measures for the first time enables a standardized assessment and comparison of road network construction algorithms. All data, road network construction, and evaluation algorithms are available with detailed execution instructions on the http://mapconstruction.org Web site. Directions for future work include the expansion of the web site towards the inclusion of more algorithms and source code. The final goal will be to provide an easy-to-use benchmark suite and automated quality measurements for generated maps.

References

1. Aanjaneya M, Chazal F, Chen D, Glisse M, Guibas LJ, Morozov D (2011) Metric graph reconstruction from noisy data. In: Proceedings 27th ACM symposium on computer geometry, pp 37–46
2. Agamennoni G, Nieto JI, Nebot EM (2011) Robust inference of principal road paths for intelligent transportation systems. Trans Intell Transport Sys 12(1):298–308
3. Ahmed M, Fasy BT, Hickmann KS, Wenk C (2013) Path-based distance for street map comparison. arXiv:1309.6131
4. Ahmed M, Wenk C (2012) Constructing street networks from GPS trajectories. In: Proceedings 20th annual european symposium on algorithms, pp 60–71
5. Alt H, Efrat A, Rote G, Wenk C (2003) Matching planar maps. J Algoritm:262–283
6. Alt H, Guibas L (1999) Discrete geometric shapes: matching, interpolation, and approximation-a survey. In: Sack JR, Urrutia J (eds) Handbook of Computational Geometry. Elsevier, New York, pp 121–154
7. Biagioni J, Eriksson J (2012) Inferring road maps from global positioning system traces: Survey and comparative evaluation. Transp Res Rec: J Transp Res Board 2291:61–71
8. Biagioni J, Eriksson J (2012) Map inference in the face of noise and disparity. In: Proceedings 20th ACM SIGSPATIAL, pp 79–88
9. Brakatsoulas S, Pfoser D, Salas R, Wenk C (2005) On map-matching vehicle tracking data. In: Proceedings 31st VLDB Conference, pp 853–864
10. Bruntrup R, Edelkamp S, Jabbar S, Scholz B (2005) Incremental map generation with GPS traces. In: Proceedings IEEE Intelligent Transportation System, pp 574–579
11. Cao L, Krumm J (2009) From GPS traces to a routable road map. In: Proceedings 17th ACM SIGSPATIAL, pp 3–12
12. Chen C, Cheng Y (2008) Roads digital map generation with multi-track GPS data. In: Proceedings workshops on education technology and training, and on geoscience and remote sensing. IEEE, pp 508–511
13. Chen D, Guibas LJ, Hershberger J, Sun J (2010) Road network reconstruction for organizing paths. In: Proceedings 21st ACM-SIAM symposium on discrete algorithms, pp 1309–1320
14. Cheong O, Gudmundsson J, Kim HS, Schymura D, Stehn F (2009) Measuring the similarity of geometric graphs. In: Proceedings of the 8th international symposium on experimental algorithms, SEA '09. Springer, Berlin, pp 101–112
15. Conte D, Foggia P, Sansone C, Vento M (2004) Thirty years of graph matching in pattern recognition. Int J Pattern Recognit Artif Intell 18(3):265–298
16. Davies JJ, Beresford AR, Hopper A (2006) Scalable, distributed, real-time map generation. IEEE Pervasive Comput 5(4):47–54
17. Edelkamp S, Schrödl S (2003) Route planning and map inference with global positioning traces. In: Computer Science in Perspective. Springer, pp 128–151
18. Efentakis A, Brakatsoulas S, Grivas N, Lamprianidis G, Patroumpas K, Pfoser D (2013) Towards a flexible and scalable fleet management service, pp 79–84
19. Efrat A, Itai A, Katz MJ (2001) Geometry helps in bottleneck matching and related problems. Algorithmica 31:1–28
20. Fathi A, Krumm J (2010) Detecting road intersections from GPS traces. In: Proceedings 6th international conference on geographic information science, pp 56–69
21. Gao X, Xiao B, Tao D, Li X (2010) A survey of graph edit distance. Pattern Anal Applic 13:113–129
22. Gati G (1979) Further annotated bibliography on the isomorphism disease. J Graph Theory 3(2):95–109
23. Ge X, Safa I, Belkin M, Wang Y (2011) Data skeletonization via Reeb graphs. In: Proceedings 25th annual conference on neural information processing systems, pp 837–845
24. Goodchild MF (2007) Citizens as voluntary sensors: spatial data infrastructure in the world of web 2.0. Int J Spat Data Infrastructures Res 2:24–32
25. Guo T, Iwamura K, Koga M (2007) Towards high accuracy road maps generation from massive GPS traces data. In: Proceedings IEEE international geoscience and remote sensing symposium, pp 667–670
26. Haklay M, Weber P (2008) Openstreetmap: User-generated street maps. IEEE Pervasive Comput 7(4):12–18
27. Jang S, Kim T, Lee E (2010) Map generation system with lightweight GPS trace data. In: Proceedings 12th international conference on advance communication technical, pp 1489–1493

28. Karagiorgou S, Pfoser D (2012) On vehicle tracking data-based road network generation. In: Proceedings 20th ACM SIGSPATIAL, pp 89–98
29. Kégl B, Krzyzak A, Linder T, Zeger K (2000) Learning and design of principal curves. IEEE Trans Pattern Anal Mach Intell 22(3):281–297
30. Liu X, Biagioni J, Eriksson J, Wang Y, Forman G, Zhu Y (2012) Mining large-scale, sparse GPS traces for map inference: comparison of approaches. In: Proceedings 18th ACM SIGKDD, pp 669–677
31. Mondzech J, Sester M (2011) Quality analysis of openstreetmap data based on application needs. Cartographica 46:115–125
32. Niehofer B, Burda R, Wietfeld C, Bauer F, Lueert O (2009) GPS community map generation for enhanced routing methods based on trace-collection by mobile phones. In: Proceedings 1st international conference on advances in satellite and space communication, pp 156–161
33. OpenStreetMap Foundation (2013) Bulk gpx track data. http://blog.osmfoundation.org/2013/04/12/bulk-gpx-track-data/
34. OpenStreetMap Foundation (2013) Openstreetmap: User-generated street maps. http://www.openstreetmap.org
35. OpenStreetMap Foundation (2014) http://blog.openstreetmap.org/2013/04/12/bulk-gpx-track-data/
36. Quddus M, Ochieng W, Noland R (2007) Current map-matching algorithms for transport applications: state-of-the art and future research directions. Transp Res Part C: Emerg Technol:312–328
37. Read RC, Corneil DG (1977) The graph isomorphism disease. J Graph Theory 1(4):339–363
38. Rogers S, Langley P, Wilson C (1999) Mining GPS data to augment road models. In: Proceedings 5th ACM SIGKDD, pp 104–113
39. Schroedl S, Wagstaff K, Rogers S, Langley P, Wilson C (2004) Mining GPS traces for map refinement. Data Min Knowl Discov 9:59–87
40. Shi W, Shen S, Liu Y (2009) Automatic generation of road network map from massive GPS vehicle trajectories. In: Proceedings 12th international IEEE conference on intelligent transportation systems, pp 48–53
41. Steiner A, Leonhardt A (2011) Map generation algorithm using low frequency vehicle position data. In: Proceedings 90th annals meeting of the transportation research board, pp 1–17
42. Wang Y, Liu X, Wei H, Forman G, Chen C, Zhu Y (2013) Crowdatlas: self updating maps for cloud and personal use. In: Proceedings 11th international conference mobile systems, applications and services
43. Worrall S, Nebot E (2007) Automated process for generating digitised maps through GPS data compression. In: Proceedings australasian conference on robotics and automation
44. Zeng Z, Tung AKH, Wang J., Feng J., Zhou L. (2009) Comparing stars: on approximating graph edit distance. In: Proceedings 35th VLDB conference, pp 25–36
45. Zhang L, Thiemann F, Sester M (2010) Integration of GPS traces with road map. In: Proceedings 3rd ACM SIGSPATIAL international workshop on computational transportation science, pp 17–22
46. Zheng Y, Xie X, Ma WY (2010) Geolife: a collaborative social networking service among user, location and trajectory. IEEE Data Eng Bull 33(2):32–39

Mahmuda Ahmed is a PhD candidate in the Department of Computer Science at the University of Texas at San Antonio. She received her BS at the Bangladesh University of Engineering and Technology, and she has gathered software development experience in industry. Her research area is in algorithms for geospatial data, computational geometry and geometric data mining.

Sophia Karagiorgou is a post-doctoral researcher at the School of Electrical and Computer Engineering, National Technical University of Athens. She received her Ph.D. in 2014 from the same university. Her research interests are in data mining of user-contributed geospatial data and specifically addressing map-construction problems for the case of GPS tracking data.

Dieter Pfoser is an associate professor at the Department of Geography and Geoinformation Science, George Mason University, USA and a senior researcher at the ATHENA Research Center, Greece (on leave). His research interest includes spatiotemporal databases and crowdsourcing of geospatial data. He is an editorial board member of Transactions in GIS and associate editor of Geoinformatica. He has been a PC co-chair of W2GIS 2014, organizer of the 2nd SIGSPATIAL GEOCROWD Workshop 2013, a PC co-chair of SSTD 2011, and a senior PC member of the ACM SIGSPATIAL GIS conference series.

Carola Wenk is an Associate Professor of Computer Science at Tulane University. She received her PhD from Freie Universität Berlin, Germany. Her research area is in algorithms and computational geometry, with a focus on shape matching algorithms and interdisciplinary applications ranging from computational biology to intelligent transportation systems. Dr. Wenk has won research, teaching, and service awards.