National Technical University of Athens

School of Rural and
Surveying Engineering

Topography Division

Inference of Transportation Networks from Sparse Tracking Data

PhD Thesis

of

Sophia Karagiorgou

Computer Science (2006), MSc (2009)

Athens, July 2014

National Technical University of Athens
School of Rural and Surveying Engineering
Topography Division

Inference of Transportation Networks from Sparse Tracking Data

PhD Thesis

of

Sophia Karagiorgou

Computer Science (2006), MSc (2009)

Supervising Committee: V. Vescoukis
R. Korakitis
T. Sellis

Approved by the Examination Committee, 31stth July 2014.

...	...	...
V. Vescoukis	R. Korakitis	T. Sellis
Assist. Prof. NTUA	Prof. NTUA	Prof. RMIT University
...	...	...
D. Pfoser	L. Tsoulos	B. Nakos
Assoc. Prof. GMU	Prof. NTUA	Prof. NTUA
	...	
	D. Delikaraoglou	
	Assoc. Prof. NTUA	

Athens, July 2014

...

Sophia Karagiorgou
Computer Science, MSc, PhD

ABSTRACT

The commoditization of tracking technology, e.g., smart-phone applications involving check-ins, real-time navigation applications, fleet management, etc., provides us with a wealth of tracking data that, when utilized properly, will allow us to derive road and transportation networks. The research challenges in the form of map inference methods have been addressed to a limited extent in literature. Existing methods are characterized by limited geographical scope, small-scale tracking datasets, and unconvincing map construction results. Thus, sophisticated map inference algorithms are needed to improve over the current shortcomings and provide methods that can also be used in a practical setting.

The present thesis contributes to this knowledge by proposing automatic transportation map inference algorithms for the simpler case of road networks, and the more complex case of semantically more expressive multimodal networks-of-interest. As a result of our study of automatic road network inference, we propose the TraceBundle algorithm, which extracts an arbitrary road network from large vehicle tracking datasets. A second method, the TraceConflation algorithm addresses noisy and sparse datasets. This method allows for the updates of existing road maps, both in terms of geometry and connectivity. The merit of these techniques is the automatic inference of navigable road networks of high spatial accuracy with respect to their geometry, enriched with additional attributes such as permitted maneuvers and road categories.

Besides GPS tracking data and road networks, this thesis also studies a novel technique for dealing with user generated geospatial tracking data derived from social media applications. The proposed method, i.e. the Network-of-Interest algorithm, allows us to discover transportation hubs and critical transportation infrastructure from geocoded tweets. To further motivate and facilitate researchers and practitioners working in this area, we have created the `http://www.mapconstruction.org` *which is an online repository containing source code of the state-of-the-art algorithms, as well as datasets for testing and evaluation. Finally, to investigate and demonstrate the applicability of our map inference algorithms in additional application domains, the map-construction approach has been applied to the visualization of eye tracking data.*

ΠΕΡΙΛΗΨΗ

Οι τεχνολογίες εντοπισμού θέσης που συναντώνται σε εφαρμογές έξυπνων κινητών, πραγματικού χρόνου πλοήγησης ή διαχείρισης στόλου οχημάτων, όχι μόνο επιτρέπουν την παροχή μιας σημαντικής και συστηματικής πηγής δεδομένων παρακολούθησης που δίνει τη δυνατότητα να αντλήσει κανείς πληροφορία σχετικά με οδικά και μεταφορικά δίκτυα εν γένει, αλλά εισάγουν επίσης ένα πλήθος ανοιχτών ερευνητικών προκλήσεων σε σχέση με την αξιοποίηση τέτοιων πλούσιων δεδομένων. Στη βιβλιογραφία έχουν προταθεί ορισμένες μέθοδοι, ως μέθοδοι αυτόματης παραγωγής χαρτών, αλλά δεν έχουν μελετηθεί επαρκώς τα προβλήματα που απορρέουν από μεγάλης κλίμακας δεδομένα και γενικευμένης γεωμετρίας οδικά δίκτυα. Έτσι, προκύπτουν νέες τάσεις που δίνουν έμφαση στην ανάγκη για βελτιστοποιημένους αλγορίθμους αυτόματης εξαγωγής χαρτών για την παροχή εφαρμογών και υλοποιήσεων που θα είναι εύρωστες ως προς τον όγκο και την ποιότητα των δεδομένων και θα επιτρέπουν ενημερώσεις σχετικά με τη γεωμετρία και τη συνδεσιμότητα των οδικών δικτύων.

Στην κατεύθυνση αυτή, η παρούσα διατριβή συνεισφέρει στη γνώση, προτείνοντας αλγορίθμους αυτόματης παραγωγής χάρτη για οδικά δίκτυα και δίκτυα μεταφοράς γενικευμένου ενδιαφέροντος. Αρχικά μελετάται το πρόβλημα της αυτόματης παραγωγής του χάρτη ενός οδικού δικτύου. Στο πλαίσιο αυτό, προτείνεται ο αλγόριθμος TraceBundle που είναι πρωτότυπος και εξάγει ένα οδικό δίκτυο τυχαίας γεωμετρίας από μεγάλης κλίμακας δεδομένα, σε σχέση με άλλες προσεγγίσεις που προϋποθέτουν αυστηρούς γεωμετρικούς κανόνες καθετότητας των δρόμων. Στη συνέχεια, προτείνεται ο αλγόριθμος TraceConflation που χρησιμοποιεί δυναμικές παραμέτρους εφαρμογής καθώς και χαμηλής ακρίβειας, μη συστηματικά δεδομένα και επιτρέπει ενημερώσεις σε οδικούς χάρτες, από πλευράς γεωμετρίας και συνδεσιμότητας. Το όφελος από αυτές τις δυο τεχνικές είναι η αυτόματη εξαγωγή οδικών δικτύων υψηλής γεωμετρικής ακρίβειας καθώς και επιπρόσθετων χαρακτηριστικών, όπως είναι οι επιτρεπόμενες στροφές, οι κατηγορίες των δρόμων κλπ. Επίσης, μελετάται και παρουσιάζεται μια νέα τεχνική που αξιοποιεί γεωχωρικά δεδομένα από εφαρμογές κοινωνικής δικτύωσης. Αυτή η μέθοδος επιτρέπει την εξαγωγή κρίσιμων υποδομών μεταφορών και σημαντικών κόμβων από γεωκωδικοποιημένα tweets. Προκειμένου να δοθεί το κίνητρο σε ερευνητές να εργαστούν και να βελτιώσουν τους εν λόγω αλγορίθμους, έχει δημιουργηθεί ένα διαδικτυακό αποθετήριο (http://www.mapconstruction.org) που καθιστά διαθέσιμα τον πηγαίο κώδικα αλγορίθμων κατασκευής χαρτών καθώς και τα δεδομένα καταγραφής ιχνών οχημάτων για έλεγχο και αξιολόγηση. Τέλος, παρουσιάζεται η επέκταση της τεχνικής σε εφαρμογές για την απεικόνιση δεδομένων καταγραφής του ανθρώπινου οφθαλμού σε παρατηρήσεις χαρτογραφικών γραμμών και σχετίζεται με την αξιοποίηση γνώσης σε τροχιές κινούμενων αντικειμένων.

ACKNOWLEDGMENTS

There are several people who helped and supported me during the work on this thesis. First of all, I am grateful to Prof. Vassilios Vescoukis for his support without which this thesis could not have been accomplished. His suggestions on several issues were valuable throughout these years, especially during the most critical periods, at the beginning of my work and during the last months towards its completion. I also need to thank Prof. Dieter Pfoser for his inspiration, encouragement and patience throughout all the stressful moments during this work, for his advice on various issues and valuable comments, for his constant urging and focus on detail, and for always finding the time to discuss and review all the parts of the work presented in this thesis. I also want to thank Dr. Dimitris Skoutas for the very fruitful collaboration we had, which helped me to significantly improve the quality of this work. Moreover, I would like to thank Prof. Timos Sellis who took a keen interest not only in this thesis but all aspects of my work.

I also want to thank my examination committee and all the members of the Institute for the Management of Information Systems for our discussions and for creating a very friendly working environment. I am very grateful to them, and for having the chance to work in such a pleasant and inspiring environment. Also, my warmest thanks to all my colleagues in the Geoinformatics Group of IMIS for their help and support.

Finally, I would like to thank my family for their support and patience during the hard times of this research work.

PREFACE

This thesis presents our approach for automatic road and transportation networks inference by proposing a set of novel heuristics based algorithms. The thesis presents and proposes new insights and techniques to the problem of automatic map construction.

It is submitted in fulfillment of the requirements for the degree of Doctor of Philosophy, in the School of Rural and Surveying Engineering, National Technical University of Athens (NTUA), Greece. The presented work describes a set of algorithms to infer transportation networks from sparse tracking data and has been carried out the last four years in the Institute for the Management of Information Systems (IMIS) of R.C. Athena.

Sophia Karagiorgou
Athens, July 2014

LIST OF PUBLICATIONS

From this thesis, the following papers were published:

Books

1. Ahmed M., Karagiorgou S., Pfoser D., Wenk C., Map Construction Algorithms, Springer Verlag (to be published in 2015).

Journals

1. Ahmed M., Karagiorgou S., Pfoser D., Wenk C., A Comparison and Evaluation of Map Construction Algorithms, GeoInformatica, 2014.

2. Crooks A., Pfoser D., Jenkins A., Croitoru A., Karagiorgou S., Efentakis A., Lamprianidis G., Smith D., Stefanidis A., Crowdsourcing Urban Form and Function, Int'l Journal of Geographical Information Science, 2014.

Conferences / Workshops

1. Karagiorgou S., Krassanakis V., Vescoukis V., Nakos B., Experimenting with polylines on the visualization of eye tracking data from observations of cartographic lines, Proc. of the 2nd Int'l Workshop on Eye Tracking for Spatial Research, 2014.

2. Karagiorgou S., Pfoser D., Skoutas D., Geosemantic Network-of-Interest Construction Using Social Media Data, Proc. of the 8th Int'l Conference on Geographic Information Science, 2014.

3. Karagiorgou S., Pfoser D., Skoutas D., Segmentation-Based Road Network Construction, Proc. of the 21th Int'l Conference on Advances in Geographic Information Systems, 2013.

4. Karagiorgou S., Pfoser D., On Vehicle Tracking Data-Based Road Network Generation, Proc. of the 20th Int'l Conference on Advances in Geographic Information Systems, 2012.

Contents

1 Introduction **1**
- 1.1 Challenges and Contributions ... 3
  - 1.1.1 Preliminaries, Requirements and Problem Definition ... 4
  - 1.1.2 Inference of Road Networks from Tracking Data ... 11
  - 1.1.3 Dealing with Noisy Tracking Data ... 13
  - 1.1.4 Inference of Transportation Networks from Social Media Data 14
  - 1.1.5 Going Beyond Typical Map Inference Tasks ... 16
- 1.2 Thesis Outline ... 16

2 Related Work **19**
- 2.1 Moving Objects and Spatiotemporal Data ... 19
  - 2.1.1 Sub-Sequences Extraction from Moving Objects ... 19
  - 2.1.2 Knowledge Extraction-Based Techniques of Spatiotemporal Data ... 20
- 2.2 Map Inference Algorithms ... 20
  - 2.2.1 Trajectory Clustering ... 21
  - 2.2.2 Point Clustering ... 21
  - 2.2.3 Incremental Track Insertion ... 22
  - 2.2.4 Intersection Linking ... 23
  - 2.2.5 Other Approaches ... 23
- 2.3 Quality Measures for Map Comparison ... 23
  - 2.3.1 Distance Measures based on Point Sets ... 24
  - 2.3.2 Distance Measures based on Sets of Paths ... 24

3 Inference of Road Networks from Tracking Data **27**
- 3.1 Intersections and Links - The *TraceBundle* Algorithm ... 28
  - 3.1.1 Turns and Intersections ... 28
  - 3.1.2 Connecting Intersection Nodes ... 31
  - 3.1.3 Compacting Links ... 33
  - 3.1.4 Post Processing ... 35
- 3.2 Shortest-Path Based Distance ... 35
  - 3.2.1 Underlying Road Network Extraction ... 36
  - 3.2.2 Shortest-Path Queries ... 36
- 3.3 Map Inference Algorithms ... 37
  - 3.3.1 Compared Algorithms ... 38
  - 3.3.2 Quality Measures used for Map Comparison ... 43
  - 3.3.3 Comparison of Distance Measures ... 46
- 3.4 Experimental Evaluation ... 47

3.4.1 Datasets 47
3.4.2 Results 49
3.4.3 Evaluation Comparison of Map Construction Algorithms 51
3.5 Summary 65

4 Dealing with Noisy Tracking Data 67
4.1 The *TraceConflation* Algorithm 68
4.1.1 Segmentation of Trajectories 68
4.1.2 Construction of Network Layers 69
4.1.3 Conflation of Network Layers 71
4.2 Experimental Evaluation 74
4.2.1 Datasets 75
4.2.2 Evaluation Measures 75
4.2.3 Results 76
4.3 Summary 78

5 Inference of Transportation Networks from Multimodal Crowdsourced Data 81
5.1 Network-of-Interest Layer Construction 81
5.1.1 Segmentation of Trajectories 82
5.1.2 Geometric Layer Construction 84
5.1.3 Semantic Layer Construction 85
5.2 Network-of-Interest Layer Fusion 85
5.2.1 Network Hubs 85
5.2.2 Layer Fusion 86
5.3 Experimental Evaluation 87
5.3.1 Datasets 89
5.3.2 Evaluation Measures 90
5.3.3 Results 91
5.4 Summary 92

6 Going Beyond Typical Map Inference Tasks 93
6.1 Visualization of Eye Tracking Data from Observations of Cartographic Lines 94
6.2 Inference of Polylines from Eye Tracking Data 95
6.2.1 The Proposed Algorithm 96
6.2.2 Results 97
6.3 Summary 97

7 Conclusions and Future Work 99
7.1 Conclusions 99
7.2 Future Work 101

Bibliography 103

8 Curriculum Vitae 111

List of Figures

1.1 Vehicle Tracking Data, Actual Road Network, Actual Map. 2
1.2 The Location of a Point on Earth. 5
1.3 WGS 84 Reference Frame. 6

3.1 Tracking Data. 28
3.2 Angular Difference. 29
3.3 Turn Model. 30
3.4 Computing Intersection Nodes. 31
3.5 Computing Link Samples. 32
3.6 Road Network Inference. 34
3.7 Triangular Intersections. 35
3.8 Relative Weights Comparison. 36
3.9 Clipping of the Underlying Road Network. 37
3.10 Clustering-Based Map Construction Algorithm (images from [35]). ... 39
3.11 Reeb Graph based Map Construction (images from [44]). 40
3.12 KDE-Based Map Construction using Threshold Ranges (images from [14]). 40
3.13 Clustering-Based Map Construction Algorithm (images from [30]). ... 41
3.14 Incremental Track Insertion Algorithm (images from [7]). 42
3.15 Incremental Track Insertion Algorithm (images from [21]). 42
3.16 The *TraceBundle* Algorithm. 43
3.17 Graph G (dotted edges) overlayed on H (gray). G and H differs in the shaded squared region. The distance measure in [14] fails to capture the broken connection in G, as there is always detour available to reach every edge and sample it. 46
3.18 Tracking Data. 48
3.19 Experimental Setup. 49
3.20 Parameters Selection. 50
3.21 Inferred Road Network - The *TraceBundle* Algorithm - (*Athens large*). 51
3.22 Inferred Road Network - The *TraceBundle* Algorithm - Close Ups. ... 52
3.23 Constructed Maps - in black - Overlayed on Ground-Truth Map - in gray - (small dataset). 54
3.24 Constructed Maps - in black - Overlayed on Ground-Truth Map - in gray - (large dataset). 55
3.25 A path with Fréchet distance greater than Hausdorff distance. 57
3.26 Distributions of Individual Path Distances (Biagioni alg. - *Chicago*). . 57
3.27 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. 58

3.28 Shortest-Path Examples - *Chicago* Dataset. 59
3.29 Assessing Inferred Road Network. 60
3.30 Similarity Results. 60
3.31 Link Failures. 61
3.32 Map Comparison. 61
3.33 Comparison of F-scores - *Chicago*. 63

4.1 Tracking Data. 68
4.2 Turn Samples Clustering. 70
4.3 Intersections Inference. 71
4.4 Stitching Different Network Layers. 72
4.5 Inferred Road Networks - The *TraceConflation* Algorithm. 77
4.6 Road Networks Comparison. 79

5.1 Twitter Trajectories and OSM Network - London. 83
5.2 Network Reduction Example. 84
5.3 Stitching Network-of-Interest Layers. 88
5.4 Hubs Statistics. 91
5.5 Networks of Interest. 91

6.1 Eye Tracking Data from 3 Subjects. 95
6.2 Inference of Hubs. 96
6.3 Compacting Links and Final Inferred Polyline. 98

List of Tables

3.1 Algorithm Categories. 38
3.2 Dataset Statistics. 49
3.3 Parameter Summary. 50
3.4 Complexities of the Map Construction Algorithms. 53
3.5 Path-Based and Directed Hausdorff Distance Measure Evaluation. . . . 56
3.6 Shortest-Path Based Measure Evaluation Summary. 62
3.7 Precisions for Varying *Matched Distance*. 64

4.1 Shortest-Path Comparison Summary 78

5.1 Parameter Summary. 89
5.2 Evaluation Summary. 90

Chapter 1

Introduction

The commoditization of tracking technology, e.g., smart-phone applications involving check-ins, real-time navigation applications, fleet management, etc., provides us with a considerable tracking data source that enables us to derive not only road networks, but transportation networks in general.

In this thesis, we investigate and address the map inference problem, which refers to the process of constructing and/or maintaining the map of a transportation network from tracking data of objects moving along it. Despite the apparent abundance of maps, both commercial and open, that exist nowadays, this problem has many applications. In particular, two main categories of needs can be identified as follows. The first potential use of this wealth of tracking data includes cases in which the transportation network is known, but needs to be maintained, updated, or enhanced with additional properties. Commercial road network datasets are very expensive and are static in the sense that it is not possible to update them e.g., due to the construction of a new road. Maps of road networks are traditionally created through the use of aerial imagery, 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 accidents or natural phenomena, etc. To assess change, the map data vendors typically commission costly on-site surveys to assess the change. The second scenario refers to cases where the entities move along specific trails, which have not yet been mapped. Example scenarios include the movements of hikers, tourists, or animals, cases in which a map exists but is not publicly available or is too costly to acquire. Example scenarios also include cases of natural disasters in which the road network infrastructure is wiped out and has been replaced with an ad-hoc infrastructure created in the wake of relief efforts (cf. Haiti earthquake scenario). The goal in these cases is to track the movements of the entities and use the extracted trajectories to infer a map of the movement network or to identify implicit movement patterns within a behavioral, historic or socioeconomic context.

In these emerging environments, classical maps are no longer sufficient to keep up with sudden changes. Instead, the users typically would not need a high quality map, but only a generalized map which shows specific aspects such as landmarks and important routes. To that end, two primary challenges can be identified, which are to propose and evaluate novel techniques for:

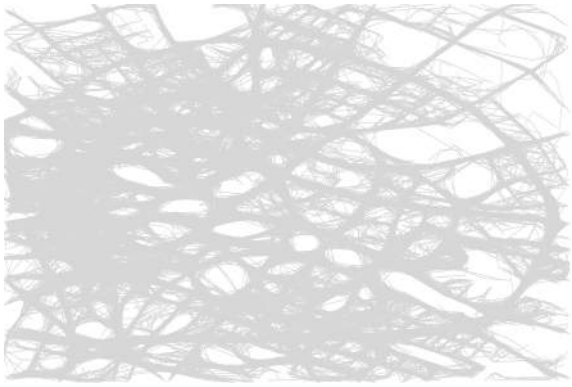
constructing transportation networks with quality guarantees

and

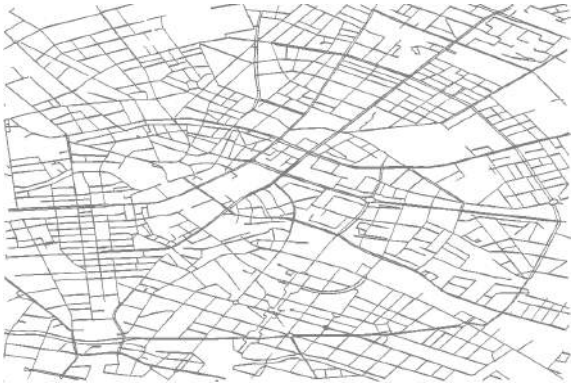
constructing geosemantically enriched networks-of-interest.

This thesis deals with a type of geometric reconstruction problem that is aimed at discovering the underlying geometric structure described by a set of movement-constrained trajectories and additional properties and characteristics to semantically enrich this data inference. This crowdsourced tracking data allows us also to gain novel insights into form and function in urban spaces, i.e., what infrastructures exist (form) and how are they utilized (function). Here, we focus in particular on information harvested from social media and other open-source and volunteered datasets (e.g. trajectories and OpenStreetMap data).

The inherent inaccuracies and errors of the collected tracking data (GPS errors, transmission errors, check-in frequency, etc.) make the map construction problem a very challenging one. Consider the example of Figure 1.1, which plots the trajectories from vehicle tracking data in Berlin (Figure 1.1a), the actual road network (Figure 1.1b) from OpenStreetMap (OSM) and the rendered OSM map (Figure 1.1c). *Our goal is to infer the road network of Figure 1.1b from the tracking data of Figure 1.1a.* Clearly, inferring a road network from trajectories is not a trivial task.



(a) Vehicle tracking data - Berlin



(b) Corresponding ground-truth road network (OSM)

(c) Corresponding map (OSM)

Figure 1.1: Vehicle Tracking Data, Actual Road Network, Actual Map.

Summarizing, in this thesis we will investigate and address the following three main problems:

- *how vast amounts of trajectory data can be harvested and exploited,*

- *how novel algorithms can benefit a wealth of map handling applications, and*
- *how to facilitate the provision of evolving and updated maps.*

By addressing this very timely problem, this thesis advances the knowledge in map construction, as well as in related fields such as databases and geographic information systems. Next, we discuss in more detail these topics, and we present the specific challenges that arise and the contributions made by this thesis to address them.

1.1 Challenges and Contributions

Towards our overall map construction goal, which is the provision of evolving and updated maps from vast amounts of tracking data, this thesis identifies and focuses on four major issues: (i) the inference of road networks from tracking data, and the comparative study of existing map-inference algorithms, (ii) the inference and maintenance of road networks from sparse tracking data, (iii) the inference of transportation networks from tracking data involving high uncertainty (social media and check-in data), and (iv) the generalized applications that utilize our approach of bundling sets of traces into geometries.

The proposed research addresses the problem of evolving and updated transportation maps. Given a finite set of tracking data, and optionally additional parameters defining accuracies, thresholds, and other prior knowledge, compute a transportation network that represents all tracking data. Tracking data are recorded as trajectories. A trajectory is a finite point sample of a continuous curve, given as a finite sequence $T = \{p_0, \ldots, p_n\}$ 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 $t_0 < t_1 < t_2 < \ldots < t_n$. Each p_i minimally consists of a time stamp as well as a position measurement, but depending on the application may contain additional data such as speed or acceleration. When using GPS to obtain the measurements, the positions are obtained from a standard projection from latitude/longitude data into the plane, and GPS measurements often also provide instantaneous speed information.

The most basic representation of a road network is as a geometric graph embedded in the plane. Each edge represents a road segment and is embedded as a polygonal curve. Each vertex represents a junction and is embedded as a single point. Edges can be directed or bi-directed, higher-level road segments such as highways are often represented as two separate directed edges. This models the basic adjacency information in road networks and is therefore widely used. This geometric graph models the underlying topology of the road network and its basic embedding in the plane.

In this thesis, we focus on exploiting the continuity of the input trajectories by using approaches from computational geometry and combine them with ideas from trajectory clustering. This includes developing appropriate models for trajectories and road networks including realistic input models, and developing algorithms to address the transportation network inference task.

In the following, we describe the challenges that arise for each one of these topics, and we present the contributions of the thesis.

1.1.1 Preliminaries, Requirements and Problem Definition

As already discussed, map construction is the process of constructing a symbolic depiction highlighting relationships between entities at any space. Especially, geographic maps of a territory have a very long tradition and exist from ancient times and mostly refer to a two-dimensional representation of the surface of the world. Recently, the problem of automatic map inference has attracted a lot of research interest. The mere existence of a wealth of tracking data has stirred the interest in algorithms that might offer a tremendous advantage to the process of modern map construction. Crowdsourced GPS data could be used to generate entirely new sections of a road map at very low cost.

Map inference can also be valuable in cases, in which a road map does exist. Here, they may not only help to increase the accuracy of existing map data, but also help in detecting changes to the road network.

From a theoretical point of view, the map construction task poses a new class of geometric shape-handling problems dealing with sets of continuous curves that are subject to noise. In this thesis, we investigate geometric and clustering models for trajectories, road networks, and transportation networks and we apply ideas from geometric shape matching, trajectory clustering and attributes extraction.

1.1.1.1 Data Collection

Data gathering can be done using a GPS enabled devices mounted on any vehicle, bicycle or smart-phone. In our case, we use data from GPS enabled devices on a fleet of school buses and taxis, and user check-ins from social media applications like Twitter. This section explains the *Geodetic coordinate system* and the *Reference systems* that we use in this thesis, surveys GPS positioning, and gives an example of the collected data set along with its visualization.

1.1.1.2 Geodetic Coordinates

The position of an object in the earth's sphere can be described in terms of Geodetic coordinates consisting of its longitude, latitude, and height (see Figure 1.2). In more details:

- Longitude of a point is the angle from the plane of the prime meridian to a plane passing through the point, both planes being perpendicular to the Equator;
- Latitude of a point is the angle from the equatorial plane to the vertical direction of a line normal to the earth's ellipsoid and passing through the point;
- Height is the distance between the point and the earth's ellipsoid.

In this thesis, we use coordinates based on the World Geodetic System 1984 (WGS84), on the Hellenic Geodetic Reference System 1987 (HGRS87) and on the Universal Transverse Mercator (UTM) reference systems.

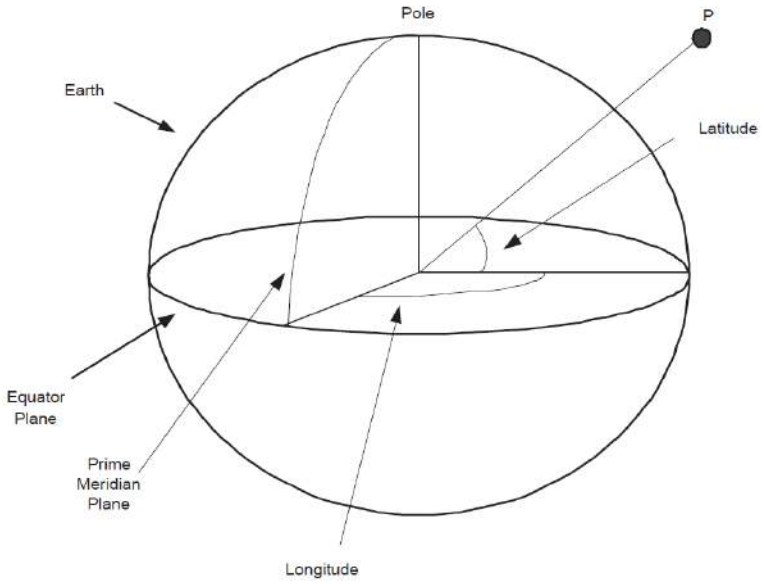


Figure 1.2: The Location of a Point on Earth.

World Geodetic System 1984 (WGS84) WGS84 is an Earth-centered, Earth-fixed terrestrial reference system and geodetic datum. WGS84 is based on a consistent set of constants and model parameters that describe the Earth's size, shape, and gravity and geomagnetic fields (Figure 1.3). WGS84 is the standard U.S. Department of Defense definition of a global reference system for geospatial information and is the reference system for the Global Positioning System (GPS). It is compatible with the International Terrestrial Reference System (ITRS). The current realization WGS84 (G1674) follows the criteria outlined in the International Earth Rotation Service (IERS) Technical Note 21 (TN 21). The responsible organization is the National Geospatial-Intelligence Agency (NGA). NGA conducted a WGS84 reference frame network adjustment in 2013 to incorporate IERS Conventions 2010 Technical Note 36 (TN 36). The latest revision is WGS 84 (aka WGS 1984, EPSG:4326), established in 1984 and last revised in 2004 [4]. Earlier schemes included WGS 72, WGS 66, and WGS 60.

- Origin: Earth's center of mass being defined for the whole Earth including oceans and atmosphere;
- Z-Axis: The direction of the IERS Reference Pole (IRP). This direction corresponds to the direction of the BIH Conventional Terrestrial Pole (CTP) (epoch 1984.0) with an uncertainty of 0.005";
- X-Axis: Intersection of the IERS Reference Meridian (IRM) and the plane passing through the origin and normal to the Z-axis. The IRM is coincident with the BIH Zero Meridian (epoch 1984.0) with an uncertainty of 0.005";
- Y-Axis: Completes a right-handed, Earth-Centered Earth-Fixed (ECEF) orthogonal coordinate system;
- Scale: Its scale is that of the local Earth frame, in the meaning of a relativistic theory of gravitation. Aligns with ITRS;

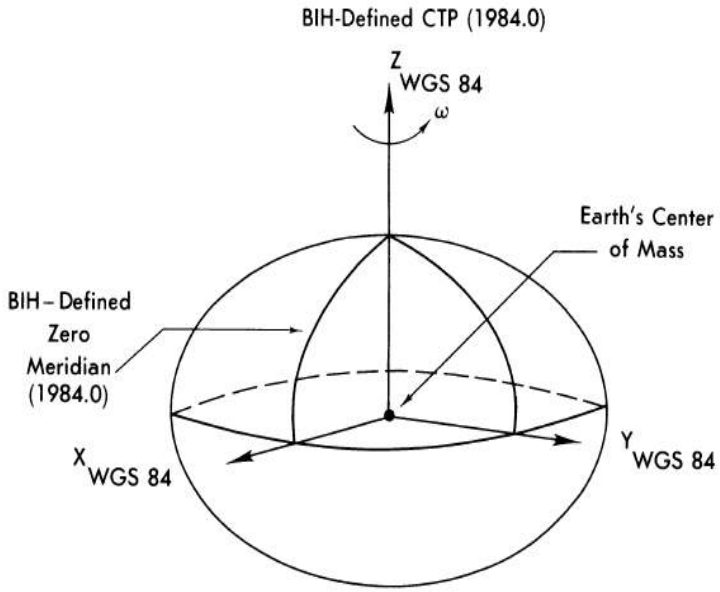


Figure 1.3: WGS 84 Reference Frame.

- Orientation: Given by the Bureau International de l'Heure (BIH) orientation of 1984.0;
- Time Evolution: Its time evolution in orientation will create no residual global rotation with regards to the crust.

Hellenic Geodetic Reference System 1987 (HGRS87) The Hellenic Geodetic Reference System 1987 or HGRS87 is a geodetic system commonly used in Greece. The system specifies a local geodetic datum and a projection system. In some documents it is called Greek Geodetic Reference System 1987 or GGRS87.

HGRS87 specifies a non-geocentric datum that is tied to the coordinates of the key geodetic station at the Dionysos Satellite Observatory (DSO) northeast of Athens. Although HGRS87 uses the GRS80 ellipsoid, the origin is shifted relative to the GRS80 geocenter, so that the ellipsoidal surface is best for Greece [33]. The specified offsets relative to WGS84 (WGS84-HGRS87) are: $\delta x = -199.87m$, $\delta y = 74.79m$, $\delta z = 246.62m$.

The HGRS87 datum is implemented by a first order geodetic network, which consists of approximately 30 triangulation stations throughout Greece and is maintained by the Hellenic Military Geographical Service. The initial uncertainty was estimated as 0.1 ppm. However there are considerable tectonic movements that move parts of Greece towards different directions, causing incompatibilities between surveys taking place at different times.

HGRS87 replaced an earlier de facto geodetic system. The datum of that system was based on the Bessel ellipsoid, with an accurate determination of the geodetic coordinates at the central premises of the National Observatory of Athens supplemented by an accurately measured azimuth from the observatory to Mount Parnes. Cartographic projections for civilian use were based on the Hatt projection system, with different projection parameters for each 1:100000 map.

HGRS87 also specifies a transverse Mercator cartographic projection (TM) with m_0=0.9996, covering 6° of longitude either side of 24° east ($18 - 30^\circ$ east). This

way all Greek territory (stretching to approximately 9° of longitude) is projected in one zone. References are in meters. Northings are counted from the equator. A false easting of $500,000m$ is assigned to the central meridian (24° east), so eastings are always positive.

Universal Transverse Mercator (UTM) The Universal Transverse Mercator (UTM) conformal projection uses a 2-dimensional Cartesian coordinate system to give locations on the surface of the Earth. It is a horizontal position representation, i.e. it is used to identify locations on the Earth independently of vertical position, but differs from the traditional method of latitude and longitude in several respects.

The UTM system is not a single map projection. The system instead divides the Earth into sixty zones, each a 6° band of longitude, and uses a secant transverse Mercator projection in each zone.

A position on the Earth is given by the UTM zone number and the easting and northing coordinate pair in that zone. The point of origin of each UTM zone is the intersection of the equator and the zone's central meridian, but to avoid dealing with negative numbers the central meridian of each zone is set at 500,000 meters East. In any zone a point that has an easting of 400,000 meters is 100 km west of the central meridian, measured on the transverse Mercator projection (or slightly more than 100 km measured on the actual surface of the earth). UTM eastings range from about 167,000 meters (near the poles) to 833,000 meters at the equator. In the northern hemisphere positions are measured northward from zero at the equator; the maximum northing value is about 9,300,000 meters at latitude 84° North, the north end of the UTM zones. In the southern hemisphere northings decrease southward from the equator to about 1,100,000 metres at 80° South, the south end of the UTM zones; the northing at the equator is set at 10,000,000 meters so no point has a negative northing value.

For instance, the CN Tower (CA) is in UTM zone 17, and the grid position is $630084m$ east, $4833438m$ north. Two points in Zone 17 have these coordinates, one in the northern hemisphere and one in the south; one of two conventions is used to say which: Append a hemisphere designator to the zone number, "N" or "S", thus "17N 630084 4833438". This supplies the minimum information to define the position uniquely. Supply the grid zone, i.e., the latitude band designator appended to the zone number, thus "17T 630084 4833438". The provision of the latitude band along with northing supplies redundant information. Because latitude band "S" is in the northern hemisphere, a designation such as "38S" is unclear. The "S" might refer to the latitude band (32°N–40°N) or it might mean "South". It is therefore important to specify which convention is being used, e.g., by spelling out the hemisphere, "North" or "South".

1.1.1.3 GPS

GPS is a mechanism for finding out the Geodetic coordinates of an object relative to the earth in 3-space. A set of satellites that are orbiting around the earth are used to find the position of an object through a GPS receiver. A GPS receiver first tries to find the maximum number of satellites around the user. Then it computes the distance between satellites and user by the time taken for the signals to reach from satellites to the user's device. Each satellite induces a sphere in its coverage region -

with itself being the center of the sphere - and gives a set of points in 3-space where the user might be. The intersection of three spheres plus the earth's sphere gives the actual position of the user. The more satellites visible, the more accurate the data. This process is called trilateration [29].

The GPS device when connected to an external machine emits continuous signals at a user-defined frequency. These signals can be collected using different software.

1.1.1.4 Accuracy of GPS Data

GPS is an outcome of the cold war era. Two major technologies emerged at that time. One is this GPS - designed and maintained by United States and the other one is Glonass - designed by Soviet Union. GPS was later permitted to be used by civilians. But because of its military importance the data provided to the civilians was not very accurate and the United States Department of Defense has all the rights to turn off the satellite coverage at any time. The current accuracy numbers in space and time for the basic GPS are the following:

- 3 meter horizontal accuracy;
- 10 meter vertical accuracy;
- 40 nanoseconds time accuracy.

In the later times, base stations were set up all around North America to enhance the accuracy of GPS data for civilian use, and the technologies such as Differential-GPS (DGPS) and Wide Area Augmenting System (WAAS) came into being. In Europe, navigation relies on the old GPS system and cannot make use of DGPS and WAAS technologies. The European Union started a project called Galileo for this purpose [93]. It was operational by 2008 and constituted of a network of 30 satellites and base stations giving highly accurate data of up to 1 meter accuracy.

GPS data are generally subject to two basic kinds of signal distortion, the *noise* and the *uncertainty* [77]. A measurement error caused by limited accuracy of the measuring device, and a sampling error representing the interpolation uncertainty caused by the finite sampling. Although the GPS measurement error can be substantial in certain situations because of shadowed and reflected signals, with the use of GPS signal augmentation (WAAS, EGNOS) the accuracy lies generally among 1, 3 and 10 meters, but can be as high as 30 meters [20, 89, 73]. A typical noise model is to assume that the measurement error has a Gaussian distribution. Modeling the sampling error is trickier because it amounts to modeling the transition between two sample points along possible original curves, and clearly it depends on the sampling rate.

Trajcevski [90] gives an overview about the different models for uncertainty of planar trajectories that have been developed in the spatio-temporal database community. A simple region-based model associates buffers of fixed radius around each trajectory [91]. If the maximum speed along each edge is known (which is a reasonable assumption for GPS data and often is part of the input) then the trajectory can be modeled as a sequence of beads in space-time [53], which project to a sequence of ellipses in the plane [77]. Each ellipse or bead encodes all possible paths from one point sample to the next. If distances are measured as shortest paths along a known road network then this restricts the possible paths to different shapes [31, 64].

Several different probabilistic movement models based on stochastic processes or random walks have been developed in ecology to study animal movement. The most popular ones are the Brownian Bridges movement model [52] and Levy walks [81].

The standard geometric graph model of a road network is an embedded geometric graph in which each edge represents a road segment embedded as a polygonal curve and each vertex represents a road junction embedded as a single point. A variant of this model associates a width w with each edge and represents each vertex and edge as a region in the plane. Commercial street map vendors such as Nokia or TomTom use the geometric graph model to represent their core "centerline street network" data. They also provide various non-geometric additional attributes such as street names, house numbers, points of interest, speed limits, travel times, as well as geometric attributes such as road category (ranging from highways to neighborhood roads) and less frequently the number of lanes and/or the width of the roads.

1.1.1.5 GPS Data Format

The original data format from the GPS follows NMEA (National Marine Electronics Association) standard. It constitutes of about 20 different attributes, out of which Geodetic coordinates, UTC (Universal Coordinated Time) and date are essentially important at least for any application specific purpose. A typical example of data collected from GPS device in WGS84 geodetic system is in the following form:
<longitude>, <latitude>, <date>, <time> e.g.
48.0070783,7.8189867,20030409,100156
48.0071067,7.8190150,20030409,100158
...

1.1.1.6 Trajectories and Movement Constraints

The ubiquitous availability of positioning technologies such as the Global Positioning System (GPS) and WiFi-based positioning (WPS) has led to the collection of vast amounts of geo-referenced movement data for different kinds of moving entities in an ever expanding range of application domains. Analyzing the resulting space-time trajectory data, however, is an immense challenge.

Movement data is generally collected in the form of trajectories. An input trajectory is a finite sequence of time-stamped position samples. It represents a finite noisy sample of a continuous curve. This thesis focuses on trajectories capturing constrained movement in the plane or in space. The movement constraints can be in the form of an explicit road network or a multimodal transportation network that the trajectories move in. In this scenario the trajectories represent curves that sample the geometric domain, and the task is to reconstruct the domain from the curve samples. In other applications the movement might be constrained by non-geometric reasons such as behavioral patterns of animals, or geocoded tweets from Twitter, or the geometric domain may be implicitly given by wind or sea currents enabling efficient flight routes for birds or swim routes for sea turtles. In this scenario the road network represents a common path structure described by the set of input trajectories.

Depending on the data capturing mechanism, trajectories are subject to different kinds of noise. Generally, the measurements of the position samples are only accurate within certain bounds (measurement error), and the movement transition

in between position samples can be modeled with varying accuracies depending on the application (sampling error) [76]. From a geometric reconstruction perspective, an (implicitly or explicitly given) geometric domain is sampled with organized data points in the form of inexact one-dimensional trajectories, and the task is to reconstruct the geometric domain. Trajectories are therefore a very unique kind of organized data representing continuous curves with inherent noise.

1.1.1.7 Maps, Road Networks and Transportation Networks

A road network is the structure given by the topology and geometry of roads, streets and transport links within a certain area. Historically, maps depicting road networks have a long tradition, dating back to a time as early as Ancient Egypt where maps such as the Turin Papyrus Map showed pedestrian routes along dry river beds [51, 50]. Around the year 1500, the cartographer Erhard Etzlaub published his Rom-Weg map designed to help pilgrims find their way to Rome, which introduced the representation of a route by a sequence of points inter-connected with lines, where each line between two neighboring points had a fixed distance of one German mile ($7.4km$) [37]. This development finally led to the modern road maps of present day, which offer an accurate depiction of cross-city as well as inner city routes for automobiles enriched with semantic information such as gas stations or speed restrictions. While Etzlaub had to rely on the testimony of traveling merchants, nowadays advanced methods of geo-referencing are employed in order to accurately assign the geographical objects used in a road map to geographical locations. Maps are increasingly authored and maintained in digital form, which may either be raster or vector based [48].

In recent times, digital vector maps have gained importance in the field of automotive navigation. These maps organize geo-referenced information in several layers. The topological layer consists of a representation of the road network given by its induced graph, where crossings and intersections correspond to nodes in the graph, and the routes interconnecting these crossings correspond to edges. By defining several classes of roads, ranging all the way from international highways down to small city streets, and assigning these classes to the edges of the graph, a routing such as the shortest or the fastest way between two point locations on the map can efficiently be computed using shortest path algorithms such as the Dijkstra algorithm [34].

In contrast, the geometrical layer of a digital map contains a description of the geometrical shape of the objects stored in the map. Often, the road geometry is approximated via a sequence of shape points, where each shape point is given by its coordinates and the road geometry is defined as the polyline constituted by the shape points. While a road is usually represented as a one-dimensional entity, more detailed maps, e.g. those required for driver assistance systems, may store additional information such as the width of a road or the boundaries of different lanes. In addition, digital road maps may contain polygonal two-dimensional objects such as footprints of buildings or special areas like parking zones.

All geographical entities found in a digital road map are geo-referenced using a geodetic reference frame such as WGS-84 [17]. Digital road maps usually enforce internal consistency, i.e. there is only one unique representation of a single geographical object. However, multiple maps of the same region may vary greatly, to the extent that a geographical object present in one map may be entirely missing in the other, or may be assigned to a different geo-reference. Also, it may (partly)

correspond to multiple objects in the other map, e.g. if a road with lanes separated by a physical divider is modeled as a single road in one map and as two one-way roads in the other.

1.1.2 Inference of Road Networks from Tracking Data

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 post-processing. Proprietary data vendors such as Navteq (now Nokia), TeleAtlas (now TomTom) and Google therefore dominated the market. While more recently, community-driven efforts have begun to challenge commercial efforts, at least for the context of non-critical applications (cf. [74]).

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. Over the last years, Volunteered Geographic Information (VGI) [46] efforts such as OpenStreetMap (OSM) [49, 74] 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.

Besides the use of such data in traffic assessment and forecasting [36], i.e., map-matching vehicle trajectories to road networks to obtain travel times [18], 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., [2, 3, 7, 13, 14, 19, 21, 22, 23, 30, 35, 39, 44, 47, 55, 58, 66, 84, 85, 87, 94, 97]. Among those only a few algorithms give theoretical quality guarantees [2, 7, 23].

An example of a constructed map is given in Figure 1.1, which shows (a) the vehicle trajectories collected for Berlin in grey color (1.1a) and (b) the respective constructed map (1.1b), shown in black color with an OpenStreetMap background map, shown in grey color. Obviously, inferring the road network from the trajectories is not a trivial task.

Advances in mobile computing have essentially led to a commodization of online navigation services with a considerable number of users now being able to determine and communicate their location. The advent of Web 2.0 applications that have positioning as their core theme, further increases the amount of tracking data that is currently available for data analysis. From this point of view, algorithms that take tracking data as input and produce map data sets are relevant, especially when considering disaster scenarios in which existing infrastructure is wiped out and ad-hoc networks need to be recorded. Besides deriving road networks, the proposed approach can be used to identify implicit movement patterns in any kinds of spatiotemporal tracking data, e.g., animal migration, historic trade routes, etc.

Existing approaches to the road network generation problem do exist. As we will see in the following, both the GIS and more recently the computational geometry communities have addressed this problem each having their specific strengths and limitations. The most common limitations of the current algorithms are that they pose strict conditions on the characteristics of data such as high sampling rate or high spatial data density. In addition, they mostly work better for well aligned road networks or can not cope with large scaled data. Besides, 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 [5, 13, 14, 58, 66]. 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 [13]. In addition to providing an extensive survey of eleven map construction algorithms, they have performed a quantitative evaluation of three representative map construction algorithms. Also, they have made their implementations of these algorithms, as well as their dataset, publicly available.

Summarizing, we can identify the following main challenges for automatic map inference techniques and the evaluation comparison of the various map construction algorithms: (a) how to cope with large scaled and low sampling rate tracking data for arbitrary road networks, (b) how to construct a robust algorithm who takes as input this tracking data and produces a final road map of high accuracy with quality guarantees, and (c) how to compare and evaluate more map construction algorithms on more diverse datasets using various quality measures suitable for different applications.

In this thesis we address these issues, making the contributions outlined below:

- We propose the *TraceBundle* algorithm, which exploits the ubiquitous trajectory data in order to analyze, reconstruct and extract road network geometries enriched by attributes. Our heuristics-based approach relies on "bundling" the trajectories around intersection nodes. Intersection nodes are derived by detecting clusters in changes to movement patterns. Essentially, we identify areas in which different types of turns are detected and designate them as intersection nodes. Linking the trajectories to intersections allows us then to derive links and consequently the entire geometry of the road network. *TraceBundle* addresses the challenges of evolving map data sets, specifically by working towards *automatic map and attribute generation* from massive amounts of vehicle tracking data. The objective is to derive an algorithm that automatically extracts the road network graph and related attributes such as road categories from tracking data obtained using GPS-based position samples (floating car data - FCD) for large vehicle fleets.

- We propose the *Shortest-Path Based Distance Measure*, which assesses the quality of a constructed road network by means of spatial accuracy and network connectivity. It does so by sampling the constructed and the actual road networks using distinct and random shortest path queries. Comparing the shortest paths generated in both networks gives us an indication of the quality of the inferred road network.

- We provide an extensive evaluation comparison of seven map construction algorithms using four benchmark tracking datasets and four different distance measures. Such an effort is only sustained in a culture of sharing that makes data, methods and source code publicly available on the internet at `mapconstruction.org`. It complements and significantly expands existing

benchmarking efforts to provide an evaluation and comparison of more map construction algorithms on more diverse datasets using various quality measures suitable for different applications. The main goal is to provide a common platform to do comparative analysis of map construction algorithms.

1.1.3 Dealing with Noisy Tracking Data

During the last years, the widespread adoption and use of GPS enabled devices in conjunction with the increasingly popular phenomenon of crowdsourcing [1, 74], has opened up new opportunities for tracking the movement of various types of entities, including vehicles, humans and animals. Consequently, this has enabled a wide spectrum of novel applications and services. One such novel use is to process the traces of moving objects in order to infer a map of a transportation network.

As a huge set of GPS vehicle traces represents information on roads and can be collected easily and quickly, some researchers conduct studies on using GPS vehicle traces to complement absent information for existing road maps. In [83], a method was developed using GPS vehicle traces to add lane information for existing road maps. Bruntrup et al. [19] introduced an incremental map-generation method, which infers the unknown road geometry based on both GPS vehicle traces and existing road maps. However, few researchers study how to update road maps with GPS vehicle traces, and there is still little information available in literature about it.

Besides, the inherent inaccuracies and errors of the collected tracking data (GPS errors, transmission errors, etc.) make the map construction problem very challenging. We focus on the scenario of inferring a road network from vehicle tracking data, which is also the case mostly studied in the literature. This scenario also has the advantage that one can use an existing map of the road network, e.g., from OSM as in the aforementioned example, as ground-truth for evaluating the accuracy of the results. Nevertheless, our approach is generic and can be applied also to other types of movement and transportation networks.

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 Section 1.1.2, we introduced the problem that the *TraceBundle* algorithm solves by taking vehicle tracking data in the form of trajectories as input and producing a road network graph. The method relies on detecting changes in the direction of movement to infer intersection nodes, and then "bundling" the trajectories around them to create the network edges. Although this approach is more robust w.r.t. noisy GPS traces and different sampling rates, it requires the tuning of several parameters to adapt to different network characteristics. Besides, *TraceBundle* algorithm uses global values for the several parameters and becomes inefficient in terms of spatial accuracy in the case of sparse data with low sampling rates.

At this stage, we address the challenges of *map generation from noisy, low-sampled tracking data*, by performing a layered construction of the network map. Here we propose the *TraceConflation* algorithm, which exploits the ubiquitous trajectory data in order to analyze, segment and reconstruct the underlying movement network in a layered form. Following this layered approach allows us to segment the input dataset into groups of trajectories based on their characteristics, and then treat each group accordingly. Moreover, it makes it possible to deal with changes

and incorporate updates in an incremental fashion. We also show that this method is more robust and provides more accurate results when dealing with noisy and heterogeneous datasets with low and non-uniform sampling rates.

In summary, we can identify the following main challenges for automatic map inference techniques using sparse and noisy tracking data: (a) how to cope with large scaled, low sampling rate and sparse tracking data for arbitrary road networks, (b) how to construct a robust algorithm who takes as input this tracking data and produces a final road map of high accuracy by using dynamically determined parameters, and (c) how to provide a mechanism to accommodate automatic map maintenance on updates. Below we outline our main contributions to these problems:

- We present *TraceConflation*, which is a new map generation algorithm that segments the input dataset into different groups of trajectories based on their characteristics and then infers the network in a layered fashion using dynamically determined values for the parameters.

- We introduce a proximity-based expansion algorithm around turn samples based on turn similarity, which allows us to create intersection nodes based on the available data by using sets of trajectories that belong to the same speed category.

- We present a detailed experimental evaluation of our method, using three real-world datasets of vehicle tracking data, which shows that the proposed method outperforms the current state-of-the-art.

1.1.4 Inference of Transportation Networks from Social Media Data

An important resource in today's mapping efforts, especially for use in mobile navigation devices, is an accurate collection of point-of-interest (POI) data. However, by only considering isolated locations in current datasets, the essential aspect of how these POIs are connected is overlooked.

Currently, the only datasets that consider connectivity of locations are typically road networks, which connect intersection nodes by means of road links purely on a geometric basis. POIs however, encode both geometric and semantic information and it is not obvious how to create meaningful links and networks between them.

The objective of this approach is to take the concept of POIs to the next level by computing *Networks-of-Interest* (NOIs) that encode different types of connectivity between POIs and capture peoples type of movement and behavior while visiting these POIs. This new concept of *Networks of Interest* has a wide array of application potential, including traffic planning, geomarketing, urban planning, and the creation of sophisticated location-based services, including personalized travel guides and recommendation systems.

We propose to capture, both *geometric* and *semantic* information in one NOI by analyzing social media in the form of spatial check-in data. We use the concept of check-in as a generic term for users actively volunteering their presence at a specific location.

Existing road maps and POIs encode mostly geometric information and consist of street maps, but may also include subway maps, bus maps, and hiking trail maps. To complement this dataset, *geometric trajectories* consist of geo-referenced trajectory data, such as GPS tracking data obtained from people moving on a road network. This type of data is assumed to have a relatively high sampling rate. Typical examples include vehicle tracking data sampled every 10 or $30s$. Such datasets are constructed using *map construction* (cf. [6], [14] for surveys).

At this stage, we use *behavioral trajectories* as a datasource. They are obtained from social media in the form of spatial check-in data, such as geocoded tweets from Twitter. Similar to GPS tracking, the user contributes a *position sample* by checking in at a specific location. Compared to geometric trajectories, such check-in data result in very low-sampling rate trajectories that when collected for many users provide for a less dense, but semantically richer "movement network" layer.

The main challenge arises from the fact that trajectories composed from geocoded tweets differ technically and semantically from raw GPS-based type of trajectories. Unlike trajectories obtained from GPS devices in typical tracking applications, such data is typically quite sparse since individuals tend to publish their positions only at specific occasions. However, we advocate that by combining and analyzing time and location of such data, it is possible to construct event-based trajectories, which can then be used to analyze user mobility and to extract visiting patterns of places.

The expectation towards behavioral trajectories is that by integrating them into a Network-of-Interest, the resulting dataset will go beyond a homogeneous transportation network and will provide us with a means *to construct an actual depiction of human interest and motion dependent on user context and independent of transportation means.*

As early maps were traces of people's movements in the world, i.e., view representations of people's experiences, *Networks of Interest* try to fuse different qualities of such trace datasets obtained through intentional (e.g., social media, Web logs) or unintentional efforts (e.g., routes from their daily commutes, check-in data) to provide for a *consequent modern map equivalent.*

Overall, we can identify the following main challenges for extracting a Network-of-Interest from noisy, low-sampled geocoded tweets: (a) how to cope with large scaled and noisy geocoded tracking data from social media applications; (b) how to extract transportation hubs, such as subways and bus stations, or bus stops in a Network of Interest; (c) how to extract and evaluate visiting patterns from event-based and behavioral trajectories. To address this problem, we propose the following approach, as outlined below:

- We introduce a new *Network-of-Interest* construction algorithm that segments the input dataset based on sampling rate and movement characteristics and then infers the respective network layers.

- We introduce a semantics-based algorithm that takes position samples (check-ins) to create network hubs and to fuse the semantic and geometric network layer into a *Network of Interest.*

- We apply a detailed experimental evaluation which uses two real-world datasets of geocoded tweets and discusses the NOI construction results in terms of quality and significance.

1.1.5 Going Beyond Typical Map Inference Tasks

Several visualization methods for eye tracking data exist to help researchers from many disciplines depict data collected in eye tracking experiments. Focusing on eye tracking data from observations of cartographic lines, we propose a new visualization of eye tracking data using polylines inferred from the analysis of samples. This visualization depicts the average line that is actually seen by subjects; such a line can be useful in the study of various optical representation concepts, such as the assessment of the effects of alternative cartographic line attributes, distractions, abstraction levels and more, as well as in other cases such as the study of visual computer interfaces.

In summary, we can identify the following main challenges for automatic inference of polylines from eye tracking data: (a) how to cope with noisy eye tracking data; (b) how to identify hubs fro eye tracking spatial fixation areas of interest; (c) how to infer similar scanning patterns of importance by tracking the eye cognitive process. To address this problem, we propose a modified *TraceBundle* approach and present some experimental results.

1.2 Thesis Outline

The remainder of this thesis is structured as follows.

Chapter 2 presents some preliminaries and a literature review of the problem of map construction and the solutions proposed by several approaches in recent years. It also presents related work on moving objects and data mining on spatiotemporal data. It concludes with quality measures and methods for map comparison and evaluation.

Chapter 3 presents the methods for the *TraceBundle* algorithm, which is an automatic road network generation algorithm that takes vehicle tracking data in the form of trajectories as input and produces a road network graph. This approach automatically extracts the intersection nodes and the road network links embedded in the trajectory data. Next, we propose a new evaluation measure to assess the quality, the spatial accuracy and the connectivity of the constructed road networks. Finally, we present a cross-comparison and an evaluation of road network construction algorithms and show the *www.mapconstruction.org* site that was established given the lack of algorithms and constructed maps being publicly available.

Chapter 4. This chapter deals with a novel methodology, the *TraceConflation* algorithm, which converts movement trajectories into a hierarchical transportation network from sparse tracking data. First, we present our technique for an improved map construction algorithm on segmented input data based on types of movement. Segmentation addresses the challenges imposed by noisy, low-sampling rate and sparse trajectories and provides for a mechanism to accommodate automatic map maintenance on updates. Then, we describe the method to hierarchically construct road network layers, based on different types of movement in an urban context, which are then combined into a single network. Finally, we present our results based on large scaled datasets, which show significant improvements in terms of quality of the constructed road network over existing approaches.

Chapter 5. This chapter takes the automatic transportation network extraction to the next level. It introduces the *Network-of-Interest* concept, which is a novel

approach that converts geocoded social media data into a mixed geosemantic (NOI). It concentrates on a novel network construction algorithm using segmented input data based on discovered mobility types. The generated network layers are then combined into a single network. Finally, we show that the proposed method allows for the discovery of critical transportation infrastructure.

Chapter 6. This chapter presents the application of map-construction algorithms to other problems, such as eye tracking data. A modified version of the *TraceBundle* algorithm is used to automatically extract polylines for eye tracking data. This use case should motivate future work and research directions for map-construction algorithms.

Chapter 7. This chapter presents our conclusions and directions for future work.

Chapter 2

Related Work

In the following, we discuss related work in the area of moving objects and spatiotemporal data, outlining the limitations of existing approaches. We also present works in the fields of map inference algorithms and quality measures for map comparison which are closely related to our approach.

Automatic transportation network inference and automatic map construction for improved navigation services have been tackled using a variety of methods including algorithms from GIS communities, probabilistic models, computational geometry algorithms such as shape matching or curve similarity, but also image processing techniques.

This thesis mostly relates to various approaches which have been proposed for using GPS traces and tracking data to either construct digital maps or refine and enhance existing ones with additional attributes. 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 [14]), incremental track insertion, and intersection linking. 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.

2.1 Moving Objects and Spatiotemporal Data

Various approaches have been proposed for using spatiotemporal data to extract useful knowledge, such as identifying travel sequences, interesting routes or socio-economic patterns. In the following, we present a review of the literature on moving objects and spatiotemporal data.

2.1.1 Sub-Sequences Extraction from Moving Objects

Several methods focus on *sub-sequence extraction (routes) from moving object trajectories* by mining spatiotemporal movement patterns in tracking data. Kisilevich et al. [61] present an automatic approach for mining semantically annotated travel sequences using geo-tagged photos by searching for sequence patterns of any length. In [24], Chen et al. extract important routes between two locations by observing the traveling behaviors of many users. Although, they mine a transfer network of important routes, they accept that the distance between any two consecutive points in a trajectory does not exceed $100m$, which becomes unrealistic.

Zheng et al. [100] use online photos from Flickr and Panoramio to analyze people's travel patterns at a tour destination. They extract important routes, but no transportation network. Asakura et al. [10] investigate the topological characteristics of travel data, but they focus on identifying a simple index of clustering tourist's behavior. Mckercher and Lau [69] identify styles of tourists and movement patterns within an urban destination.

Our approach analyzes, both traffic patterns and topological characteristics of travel routes, while most existing work focuses on traffic patterns only. Choudhury et. al [32] explore the construction of travel itineraries from geo-tagged photos. In contrast, in our approach an itinerary is defined as a spatiotemporal movement trajectory of much finer granularity.

2.1.2 Knowledge Extraction-Based Techniques of Spatiotemporal Data

Characterized by its spatial and temporal dimension, tracking data can be regarded as one kind of spatiotemporal data, which also connects this thesis to the knowledge extraction-based techniques of the spatiotemporal data mining domain. Crandall et. al [27] investigate ways to organize a large collection ($\sim$ 35 million) of geo-tagged photos and determine important locations of photos, such as cities, landmarks or sites, from visual, textual and temporal features. Kalogerakis et. al [56] estimate the geo-locations of a sequence of photos.

Similarly, Rattenbury et. al [79] and Yanai et. al [95] analyzed the spatiotemporal distribution of photo tags to reveal the inter-relation between word concepts (photo tags), geographical locations and events. Girardin et al. [45] extract the presence and movements of tourists from cell phone network data and the geo-referenced photos they generate. Similarly, [62] proposes a clustering algorithm of places and events using collections of geo-tagged photos. These approaches efficiently deliver focal spatial data extractions from diverse data sources, while the aim of this thesis is to also extract *how this data is connected (links)*. In [63], Kling studies urban dynamics based on user generated data from Twitter and Foursquare using a probabilistic model. However, these dynamics have not been translated to a (transportation) graph structure.

All these works target the extraction of some kind of knowledge and patterns from photos or geo-referenced sources with textual and spatiotemporal metadata, while we focus on mining transportation and mobility patterns from tracking data. Overall, what sets this work aside is that we use *social media data as a tracking data source*. We use it not only to extract features or knowledge patterns of human activities, but a complete multimodal transportation network.

2.2 Map Inference Algorithms

In this section, we present various methods which have been applied either to cluster trajectories or to infer transportation networks or road maps.

2.2.1 Trajectory Clustering

There also exist various methods based on *trajectory clustering*. The majority of the proposed algorithms such as k-means [68], BIRCH [98] and DBSCAN [38] work strictly with point data and do not take the temporal aspect into consideration. Several approaches match some sequences by allowing some elements to be unmatched as in the Longest Common Sub Sequence (LCSS) similarity measure [16]. However, our goal in this thesis is rather to apply a trajectory clustering approach and also take into consideration the temporal aspect of the data.

Similarity measures for trajectories that take the time and derived attributes, such as speed and direction, into account have been proposed in [75]. This approach is close to this thesis with respect to the examined aspects of temporal dimension, however, we apply clustering techniques in order to infer the connectivity of transportation networks. Besides, our approach differs in that it deals with uncertain tracking data by taking into account the spatial as well as the temporal dimension to derive a transportation network.

2.2.2 Point Clustering

Algorithms in this general category assume the input consists of a set of points which are then clustered in various different ways to obtain street segments which finally connect to 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.

2.2.2.1 Methods Based on K-Means Clustering

Some approaches employ 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. These include Edelkamp et al. [35], who develop algorithms for road segmentation, map-matching, and lane clustering. In [84], the k-means algorithm was used to refine an existing map rather than building it entirely from scratch.

Guo et al. [47], make use of statistical analysis of GPS tracks, assuming that the GPS data follows a symmetric 2D Gaussian distribution. This assumption may become unrealistic, especially in error-prone environments. Worrall et al. [94] 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 on the 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.

Similarly, Jang et al. [55] proposed a system of map construction with less than ten traces and presented in a very small scale and without any reference to the data features (i.e., sampling rate, GPS error). Agamennoni et al. [3] presented 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. [66] 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.

2.2.2.2 Methods Based on Kernel Density Estimation

Another approach related to map inference methods 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) [22], or when there is a lot of data redundancy [14, 87, 85, 30]. In [14], a dataset of university shuttle buses was used, being sampled very frequently ($2s$-$6s$), while in [30] GPS samples are obtained every $1s$. In [87] although it is stated that low-frequency vehicle position data is used, the required sampling interval is at most $15s$. A similar approach to [14] is presented in Liu et al. [66].

Generally, KDE algorithms have a hard time overcoming the problem of noisy samples when they accumulate in an area. Recently, Wang et al. [92] addressed the problem of map updates by applying their approach to OpenStreetMap data using a KDE-based approach.

Recently, Wang et al. [92] addressed the problem of map updates by applying their approach to OpenStreetMap data using a KDE-based approach. The KDE algorithms are also quite sensitive with respect noise.

2.2.2.3 Methods Based on Computational Geometry

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 [23, 44], or other neighborhood complexes such as the Vietoris-Rips complex [2] and also providing quality guarantees. 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. [2] 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. [23] focus on detecting "good" street portions in the road network and connect them subsequently. The theoretical quality guarantees, however, assume dense point sample coverage and error bounds, and make assumptions on the road geometry. Both approaches are based on sub-sampling the trajectory data and then using an unordered set of points to derive the complete road network.

2.2.3 Incremental Track Insertion

Algorithms in this category construct a street map by incrementally inserting tracks into an initially empty map [72], often making use of map-matching ideas [78]. Distance measures and vehicle headings are also used to perform additions and deletions during the incremental construction of the map.

One of the first algorithms in this category [82] clusters the tracks merely to refine an existing map and not to compute it from scratch. Cao and Krumm [21] 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. [19] propose a spatial-clustering based algorithm that requires high quality tracking data (sampling rate and positional accuracy), while Liu et al. [66] efficiently build a road network, but require accurate data and high sampling rates of 1Hz. The work in [97] 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 [7] present an incremental method that employs the Fréchet distance to partially match the tracks to the map. While they give partial quality guarantees, their approach does not address the basic connectivity problem and how to measure the respective quality of a generated network.

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 $5m$. The result quality improves with the amount of available data (redundancy) rather than the data quality itself.

2.2.4 Intersection Linking

While related to point clustering, the intersection linking approach is to first detect the intersection vertices of the street map, and in a second step link those intersections together by identifying suitable street segments.

Fathi and Krumm [40, 39] 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 well aligned maps and it uses frequently sampled data of 1s or 5s. Our approach in Chapter 3 relates to this category. It 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.

2.2.5 Other Approaches

Road networks can be derived from satellite or aerial images by means of image processing techniques [65, 54, 67, 12, 99, 71]. For example, Tavakoli et al. [88] group together edges found by an edge detector into shapes representing buildings and roads. As such they differ from the present approach, which relies on trajectory data.

2.3 Quality Measures for Map Comparison

There are two key ingredients for evaluating the quality of a constructed map: (1) the availability of an adequate ground-truth map G as part of the benchmark data, and (2) 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 [13, 14, 58, 66]. By using graph topologies resulting from 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.

In the graph theory literature, there are various distance measures for comparing two abstract graphs, that do not necessarily have a geometric embedding [26, 43, 80]. 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 [42, 96] 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. [25] 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.

2.3.1 Distance Measures based on Point Sets

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 [9]. The main drawback of such an approach is that it does not use the topological structure of the graph. Biagioni and Eriksson [13, 66], use two distance measures that essentially both use a variant of a partial one-to-one bottleneck 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.

2.3.2 Distance Measures based on Sets of Paths

For path-based distance measures on the other hand, the underlying idea is to represent the graphs by sets of paths, and then 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 [70] use shortest paths to compare the suitability of two

road networks for pedestrian navigation by considering basic properties such as respective path length. Our approach [58] 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 [5] 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 *local signature* to identify *how* and *where* two graphs differ.

Chapter 3

Inference of Road Networks from Tracking Data

Road networks and, more generally, transportation networks are an interesting research field in that they represent the principal data set for a large range of applications, including GIS, location-based services, transportation systems and Web mapping.

Map construction 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 chapter presents an automatic road network inference algorithm that takes vehicle tracking data in the form of trajectories as input and produces a road network graph. This effort addresses the challenges of evolving map data sets, specifically by focusing on (i) automatic map-attribute generation, (ii) automatic road network inference, and (iii) providing a quality assessment. The steps for the automatic road network inference algorithm constitutes the *TraceBundle* algorithm. Also, the steps for the quality assessment of a constructed road network constitutes the *Shortest-Path based Distance* measure.

Besides, it presents 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 `mapconstruction.org`. This site has been established as a repository for map construction data and algorithms to motivate other researchers to contribute by uploading code and benchmark data supporting their contributions to map construction algorithms.

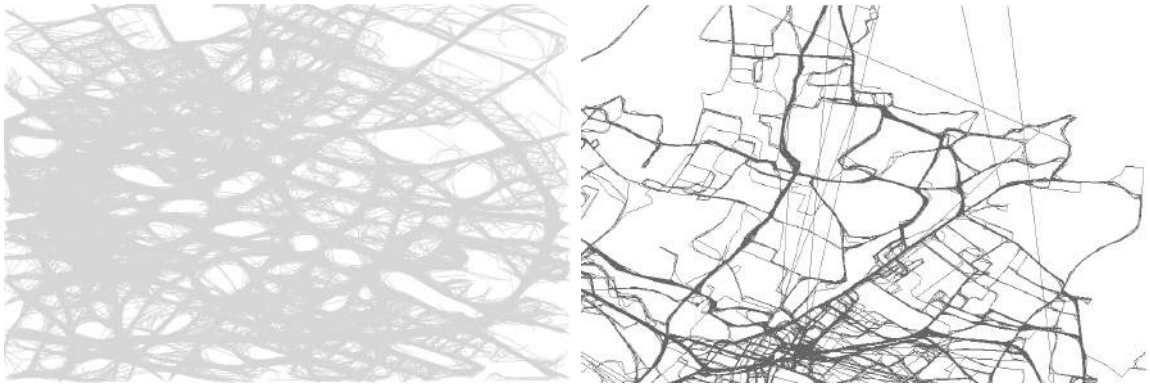
The remainder of this chapter is organized as follows. Section 3.1 discusses the methods developed regarding trajectories clustering and calculation of intersection nodes, connecting intersection nodes and the inference of a road network. All these methods constitute the *TraceBundle* algorithm. Additionally, we suggest an evalua-

tion approach in order to provide quality guarantees of the inferred road network in association with the underlying network. This is discussed in Section 3.2. We continue in Section 3.3 by presenting the algorithms which are used for the evaluation comparison of map construction approaches. In Section 3.4, we present experimental, comparison and evaluation results. Finally, Section 3.5 presents our conclusions.

Our results in this chapter have been published in [58, 6, 28].

3.1 Intersections and Links - The *TraceBundle* Algorithm

The contribution of this work is to derive a road network by sampling it using vehicles and GPS tracking. By means of the (set of) algorithms that is discussed in the following, the tracking data are essentially reduced to the actual road network geometry. In addition, road categories are derived based on the amount of data that is available for particular road network portions. Our task is to align the vehicle trajectories, so as to derive the actual road network underlying it. Figures 3.1a, 3.1b plot such tracking data with the principal roads of the actual road network being (at least visually) evident.



(a) Berlin, taxi fleet data (b) Athens, school bus data

Figure 3.1: Tracking Data.

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

The remaining of this section describes the methods applied in order to refine input data sets and continues by presenting the three steps of the *TraceBundle* algorithm.

3.1.1 Turns and Intersections

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

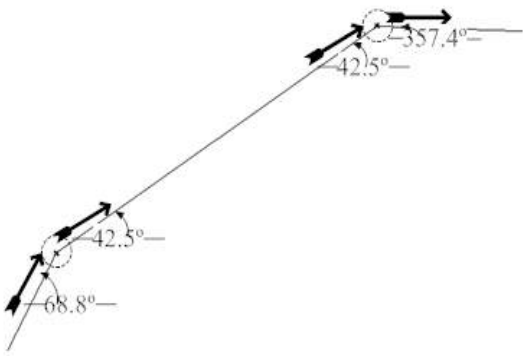


Figure 3.2: Angular Difference.

3.1.1.1 Indicators

Deriving from a common-sense understanding of vehicular movement, when turning, a vehicle (i) reduces its speed and (ii) changes its direction. Our approach uses a threshold of 40km/h as a reduced-speed indicator in combination with a change of direction. Figure 3.2 gives an example of *a trajectory with two position samples* and the respective *direction vectors*. Through experimentation, we established a threshold of 15° as the proper value for the specific road network case. Algorithm 1 gives the pseudo-code of the *Intersection Detection* algorithm. Here, the direction and speed difference are considered in Lines 11 and 12, respectively.

The Intersection Detection algorithm scans all trajectories in a position-by-position and an edge-by-edge manner taking into consideration the above conditions (Lines 9-16). We record all positions that satisfy the turn conditions (Line 14) and label them *turn samples*.

3.1.1.2 Clustering Turns

Categorizing turns by means of a *turn model* will enable us to cluster turn samples stemming from different trajectories and deriving intersections. The turn model describes all the possible movement patterns, using math degrees (0 is east, degrees increase counter-clockwise), in relation to an intersection. The turn samples are classified by using *eight types of turns*. The turn types captures all the possible combinations of incoming and outgoing links at a candidate intersection. Odd numbers are used for outgoing turns and even numbers for incoming turns resulting in four turn pairs as shown in Figure 3.3.

Using the turn model, the turn samples can be grouped according to (i) spatial proximity and (ii) turn similarity. All discovered turn samples are organized using a turn identifier, as an attribute for each of the eight turns. Within each turn category now, we use an agglomerative hierarchical clustering method and a distance threshold of 50m (cf. Algorithm 1, Line 18) to identify *turn clusters*, i.e., turn samples clustered together based on location (candidate intersection location) and turn type.

Figure 3.4a shows the calculated result for three roads that are met at an intersection. X and O markers are used for "odd" and "even" turn types, respectively. Using colour we further distinguish turn types. Yellow is used for types 1 and 2, orange for 3 and 4, red for 5 and 6, and black for 7 and 8. The turns model supports efficiently up to 4-way intersections.

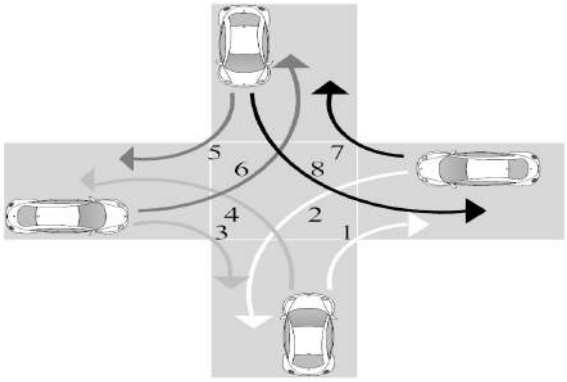


Figure 3.3: Turn Model.

3.1.1.3 Intersections

Having identified turns, the question that remains to be answered is how we actually derive intersections. Again, performing agglomerative hierarchical clustering in connection with a distance threshold, (in our case 25m), we translate the turn clusters established in the previous step, into intersection nodes (Algorithm 1, Line 20).

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

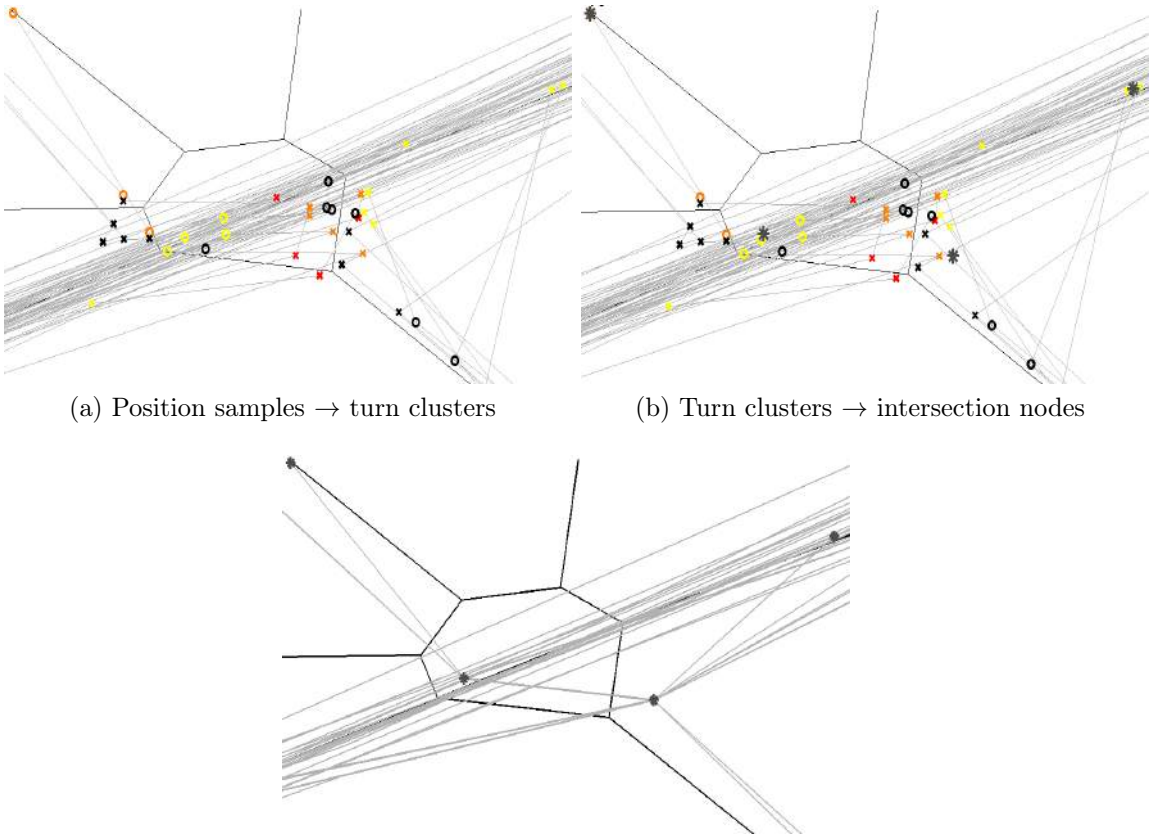
Algorithm 1: Finding Intersections.

Input: A set of trajectories T
Output: A set of Intersection nodes I

```
1  begin
2      /*Intersection nodes extraction based on trajectory data*/
3      P ← ∅ // Position sample in trajectory
4      P_S ← ∅ // Turn samples
5      P_C ← ∅ // Turn clusters
6      I ← ∅ // Intersection nodes
7      Angle, Speed, Dist // Parameter thresholds
8      // Process all position samples in all trajectories
9      while (T[i] ≠ null) do
10         P ← T[i] // Positions samples of a single trajectory
11         a_p ← AngularDiff(P[i − 1], P[i], P[i + 1])// Angular Difference
           v_p ← δx(P[i − 1], P[i]) / δt(P[i − 1], P[i]) // Mean speed
12
13         if (a_p ∈ Angle and v_p ∈ Speed) then
14             P_S.insert(P[i], TurnType(P[i]))
15         end
16     end
17     // Cluster turn samples into turn clusters
18     P_C ← ClusterTurns(P_S, Dist)
19     // Cluster turn clusters into intersection nodes
20     I ← ClusterIntersections(P_C, Dist)
21 end
```

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

too many intersection nodes. Essentially, we establish turn clusters based on distance and type of turn, whereas we then group these turn clusters into intersection nodes. Figure 3.4b shows intersection nodes as grey $*$ markers.



(a) Position samples → turn clusters

(b) Turn clusters → intersection nodes

(c) Connecting intersection nodes

Figure 3.4: Computing Intersection Nodes.

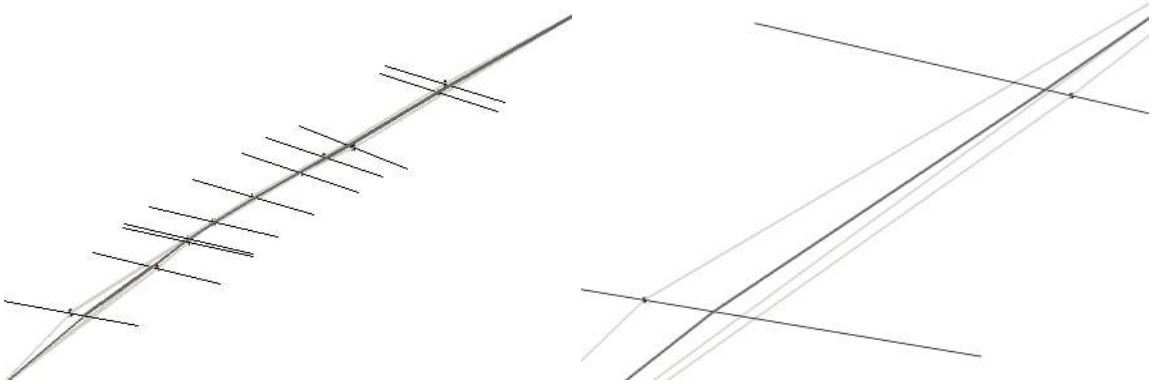
3.1.2 Connecting Intersection Nodes

At this stage in the network inference process, we succeeded in deriving isolated intersection nodes. In the following, we connect them, i.e., create links, by using the trajectory data. A fringe benefit of the intersection nodes computation based on turns is the connection of trajectory portions to these nodes, i.e., for all trajectories we know which samples helped constituting intersection nodes. To derive links we exploit this knowledge.

We record for each intersection node the outgoing and/or incoming trajectory portions connecting this node to other nodes by essentially scanning all trajectories, whether they contain sequences of intersection nodes. The result of this step is the creation of a road network that connects nodes (intersections) with (trajectory portions) links. The algorithm is simple in that it essentially examines all trajectories based on whether they contain turn samples (with turn samples constituting intersections nodes) and "marking" the respective trajectory portions.

In our data structure handling the trajectory data, all position samples that are also turn samples have been marked as such. Hence, performing a linear scan of

all trajectories reveals the respective portions of the trajectories that connect turn samples, and, hence, intersection nodes (Algorithm 2, Lines 6-14).



(a) Computing link samples between two intersections

(b) Average positions of link samples

Figure 3.5: Computing Link Samples.

Essentially, two intersection nodes in question will typically be connected by a number of trajectories, i.e., vehicles that have passed more than once from an intersection to the other. In terms of network geometry, at this stage of the overall road inference process, we introduce redundant links between intersection nodes as we simply identify how trajectories connect intersection nodes. Merging these links will be the next step. We refer to trajectory portions connecting intersections at this stage as *link samples*.

To establish link samples, we merge the spatial portion of trajectories using a sweep-line algorithm (Algorithm 2, Lines 16-26). Given a set of trajectories, at each position sample we compute an average position based on the normal distance of the position sample to all other trajectories. Figure 3.5a shows a set of positions that comprise trajectory portions and the resulting link (grey) that was derived for connecting the two intersection nodes (black crosses). Horizontal lines indicate positions samples at which the average position is computed. Figure 3.5b shows a close up of the resulting link (grey) at the first two position samples, on the left of Figure 3.5a. In addition, for each link sample (i) a *weight* is derived representing the number of the trajectories comprising a link sample and (ii) a *width* is computed as the maximum spatial extent of the trajectories (thickness of a link sample is derived by the bundled trajectories). This link sample width will be used in the next step when compacting the road network as a size parameter of the bounding box that is used in the process. Figure 3.4c shows how intersection nodes are connected by various trajectories. It also shows that trajectories that "pass through", i.e., do not turn at the intersection, have so far not been considered (but will be in the next section). In general, the number of generated intersection nodes depends highly on the parameter setting of the algorithm, i.e., choosing lower or higher threshold values will generate more or fewer intersection nodes, respectively. As also discussed in Section 3.4, the parameters need to be tuned to the network and the data in question.

Algorithm 2: Connecting Intersection Nodes.

Input: Set of trajectories T and Intersection nodes I
Output: A set of link samples L_S

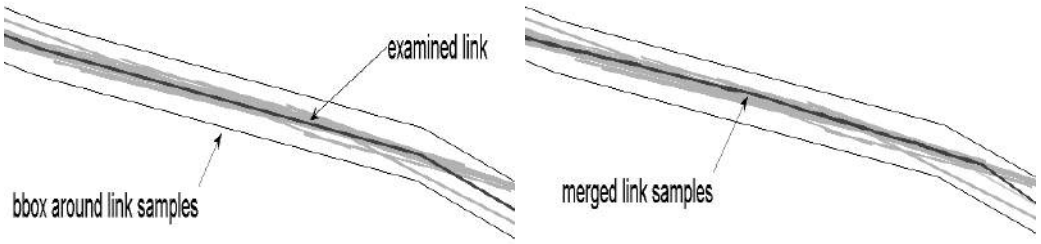
```
1  begin
2      /*Connecting intersection nodes using trajectories*/
3      I_S ← ∅ // Intersection sequence
4      L_S ← ∅ // Link samples
5      // Identify intersection sequences from trajectories
6      foreach t ∈ T do
7          foreach p ∈ t do
8              // position sample is mapped to an intersection node
9              if p ∈ I then
10                 // record prev., current intersection node
11                 I_S ← {i⁻, i, p⁻, p, t}
12             end
13         end
14     end
15     // Collect and merge link samples
16     foreach {i⁻, i} ∈ I_S do
17         // add all trajectory portions for this intersection pair
18         foreach t ∈ I_S do
19             L_S ← {t, p⁻, ..., p}
20         end
21         // cluster link samples
22         Width ← Width(L_S)
23         Weight ← Weight(L_S)
24         L_S ← SweepMerge(L_S)
25     end
26 end
```

3.1.3 Compacting Links

The state of the inferred road network at this point is that we have intersection nodes connected by links derived from trajectories that exhibit turns at these intersections. This also means that a large portion of the data, trajectories "passing through" at intersections (bulk of the data shown in Figure 3.4c), has not been considered yet with respect to all link samples. In a nutshell, the algorithm identifies trajectory portions that are close to existing links by means of a bounding box and merges their geometry onto the existing link geometry. In this step, we neither introduce new intersections nor do we add new links. We only adjust the geometry of existing links. The three steps of the algorithm include (i) sorting existing link samples with respect to their length, (ii) using a bounding box around link samples to determine relevant trajectory portions, and (iii) adjusting the geometry of links based on the trajectories' geometry.

A first step is to sort all links according to their length (Algorithm 3, Line 1) so as to process longer links first as they are more significant for link construction. I.e., the longer a link, the more selective will be the match for a longer trajectory portion to fit in a bounding box. In trying to identify portions of link samples that match other link samples expressed by spatial proximity and direction similarity, the algorithm uses a bounding box around the *examined link sample* and retrieves all intersecting portions of other links (Algorithm 3, Line 7). The size of the bounding box is determined by the *width* of the respective link sample as described in Section 3.1.2. In addition to containment in a bounding box, a threshold is used to assess direction similarity. In our experimentation, an adequate measure for direction similarity is equals to 45°. Figure 3.6a shows in black the bounding box of an examined link. The examined link is shown in grey and respective portions of other candidate links

are shown in light grey.



(a) Bounding box enclosed area (b) Merging link samples

Figure 3.6: Road Network Inference.

As a pre-cursor to merging link samples, we record for each examined link its similar links and the portions that exhibit similarity. The latter is important in order to manage partially similar link samples. As the similar link samples can be located at the beginning, the end, or the middle, the remaining portions are preserved by splitting the respective links (Algorithm 3, Line 14).

Merging link samples follows an approach similar to the one of Section 3.1.2, when connections of intersection nodes were established. The method is applied to every portion of the examined link that exhibits partial similarity to other links (Algorithm 3, Lines 10 & 16). New links are created by interpolating link samples and introducing intersection nodes. In addition, new links preserve a weight that is the sum of the weights of the merged links. Link samples are updated several times during this stage. While the examined links are reconstructed, new link samples are created and the existing are removed, i.e., additions, deletions and updates to the connectivity of the road network.

Algorithm 3: Road Network Extraction Algorithm.

Input: Set of link samples LS
Output: A set of links L

```
 1  begin
 2      LS ← sort(LS, length) // Sorting link samples by length
 3      CLS ← ∅ // Candidate link samples
 4      Width // width of link samples
 5      Angle // direction threshold
 6      foreach l ∈ LS do
 7          CLS ← Find(bbox(l, Width(l), Angle))
 8          foreach cl ∈ CLS do
 9              if Contains(l, cl) then
10                  L ← SweepMerge(l, cl)
11              end
12              else
13                  // partial overlap
14                  cl_in, cl_out ← Split(l, cl)
15                  CLS.add(cl_out)
16                  L ← SweepMerge(l, cl_in)
17              end
18          end
19      end
20  end
```

3.1.4 Post Processing

While the road extraction algorithm so far has already created a road network graph, the following heuristics-based post-processing step should further improve the quality of the road network. This is the last applied step which results in the *TraceBundle* algorithm.

The basic idea in our road extraction process is the use of turns to identify turn clusters, which in turn create intersection nodes. The underlying trajectory data is recorded by means of taking position samples at regular time intervals. In the case of turns this is especially critical, in that a position sample might create turn clusters well in advance or after the actual turn and hence introduce additional intersections. We call this phenomenon *triangular intersections*.

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

Figure 3.7 gives an example by showing in red colour link samples with high relative weight and in yellow link samples with low relative weight.

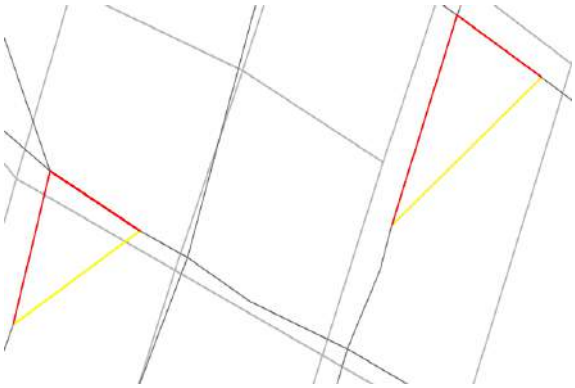


Figure 3.7: Triangular Intersections.

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

Figure 3.8 shows the distribution of such relative weight ratios (sorted by descending high relative weight ratio) for the 162 triangular intersections detected in the inferred road network described in Section 3.4.

3.2 Shortest-Path Based Distance

Essential for any automated process is the evaluation of its results. In the case of road network inference, this encompasses the assessment of the quality of the resulting road network. Ideally, we would like to compare the inferred with the existing road network graph, i.e., how do the roads and the intersections we found by means of our algorithm line up with the actual road network.

Provided that the tracking data does not cover all the road network, such a comparison should only be with respect to the corresponding portion of the network.

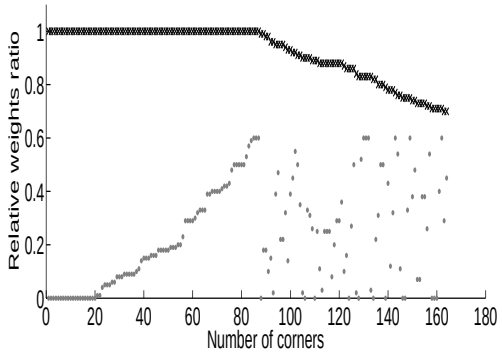


Figure 3.8: Relative Weights Comparison.

To this effect, we defined a method that extracts a subset of a road network based on given tracking data samples.

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

3.2.1 Underlying Road Network Extraction

The trajectory data covers a certain spatial area and we use this extent to derive the covered portion of the road network. I.e., the reduced network only comprises links of areas also covered by the tracking data. To find this partial network, we represent the geometry of the underlying road network with bounding boxes of a 50m distance threshold. Experimentation showed that this distance threshold is not too small to exclude road portions and not too large to include all the road network.

Now, to derive the partial network, we use the trajectories as a query set with the condition that either they cross or they are spatially contained in the bounding boxes representing the links of the actual road network. In the example of Figure 3.9, the dashed black lines represent the bounding box, the grey lines the partial road network and the yellow lines an instance of the trajectory data. As this process is by no means perfect, in this example, several redundant links have been falsely identified.

3.2.2 Shortest-Path Queries

To assess the quality of the inferred road network, we use a large number of shortest path queries in the process. We compute shortest paths for randomly selected pairs of origin and destination nodes in (a) the inferred and (b) the actual road network.

Given the constructed and ground-truth networks C and G respectively, a common set of node pairs (origin, destination) is selected in both using the nearest

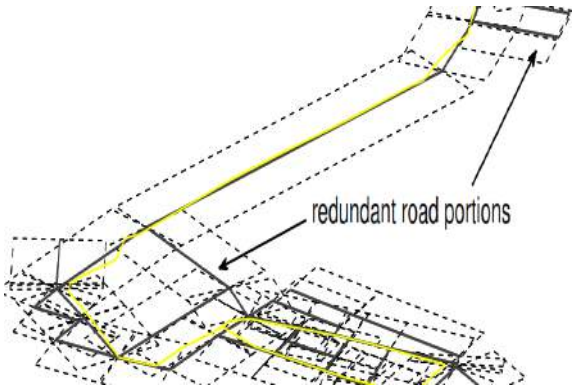


Figure 3.9: Clipping of the Underlying Road Network.

neighbor search. For all node pairs, shortest paths are computed in both networks. The geometric difference/similarity between the respective shortest paths is used to assess the similarity between C and G and consequently as a means to assess the quality of the inferred network.

Using Dijkstra's algorithm, besides the actual path, we also record the total cost of the path in terms of distance and the number of links. We compare the similarity of the shortest paths by using (i) the Discrete Frèchet distance and (ii) the Average Vertical distance to the corresponding pairs of shortest paths. The similarity measures are not applied to individual links, but to the entire paths to able to draw conclusions regarding more extensive portions of the road network. To compute these distance measures, readily available routines are used in MATLAB.

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 inferred 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 Based Distance* measure can be assessed by plotting the distance of all paths against each other, or by comparing average values for the entire set of paths. We employ both approaches in our experiments below.

The results of this evaluation can be found in Section 3.4.

3.3 Map Inference Algorithms

In this section, we present 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 we evaluate including the *TraceBundle* algorithm presented in Section 3.1, are the recent algorithms by Ahmed and Wenk [7], by Ge et al. [44], in addition to the algorithms by Cao and Krumm [21], Davies et al. [30], Edelkamp and Schrödl [35], and Biagioni and Eriksson [14]. Among those, the algorithms by [21], [30] and [35] were previously compared by Biagioni and Eriksson [13]. We have used their publicly available implementations of the algorithms by [21], [30, 35] and by [14], and the authors of [44] ran their algorithm for us. The implementations of the algorithms by [7, 58] have been made publicly available on the internet at `mapconstruction.org`.

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 [9] and the path-based distance measure presented by Ahmed et al. [5]. We also use our distance measure based on shortest paths [58] and the graph-sampling-based distance measure by Biagioni and Eriksson [14]. The implementation of the latter distance measure [14] has been made available to us by the authors.

The *tracking datasets* include the *Chicago* dataset provided by Biagioni and Eriksson [13, 14], and three additional tracking datasets: two from *Athens*, Greece and one from *Berlin*, Germany (see detail in Section 3.4.1). 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 [7] and by the *TraceBundle* algorithm [58], as well as the metric code for computing the three distance measures: the Directed Hausdorff distance [9], the path-based distance [5] and shortest-path based measure [58] publicly available on the internet at `mapconstruction.org`. We expect that such a central repository will encourage a culture of sharing and will enable the development of improved map construction algorithms.

The main goal 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. [30] 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 measure) this algorithm will not be the best choice as it ignores tracks in sparse areas as outliers/noise.

3.3.1 Compared Algorithms

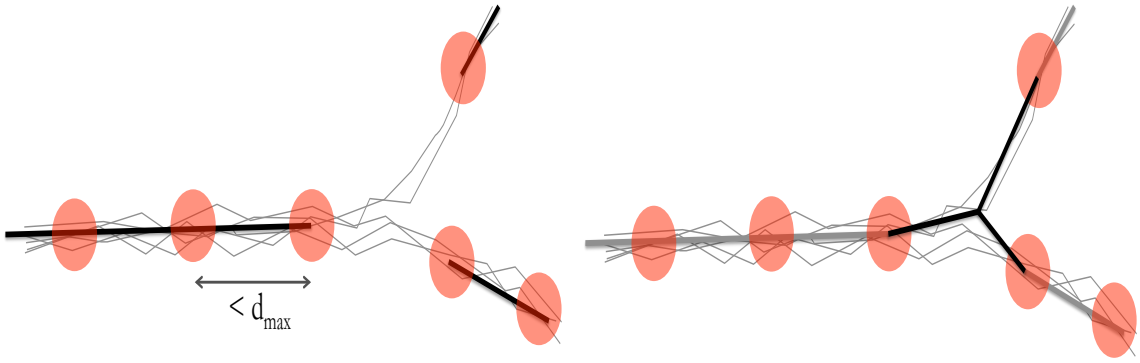
Here we give some more details on the map construction algorithms that we compare in Section 3.4. The algorithms categories are also provided in Table 3.1.

Algorithm	Point Clustering	Incremental Track Insertion	Intersection Linking
Ahmed and Wenk [7]		✓	
Biagioni and Eriksson [14]	✓		
Cao and Krumm [21]		✓	
Davies et al. [30]	✓		
Edelkamp and Schrödl [35]	✓		
Ge et al. [44]	✓		
Karagiorgou and Pfoser [58]			✓

Table 3.1: Algorithm Categories.

3.3.1.1 Point Clustering Algorithms

Edelkamp and Schrödl [35] Edelkamp and Schrödl [35] 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 finding. The lanes are found by clustering tracks based on their distance from the road center line.



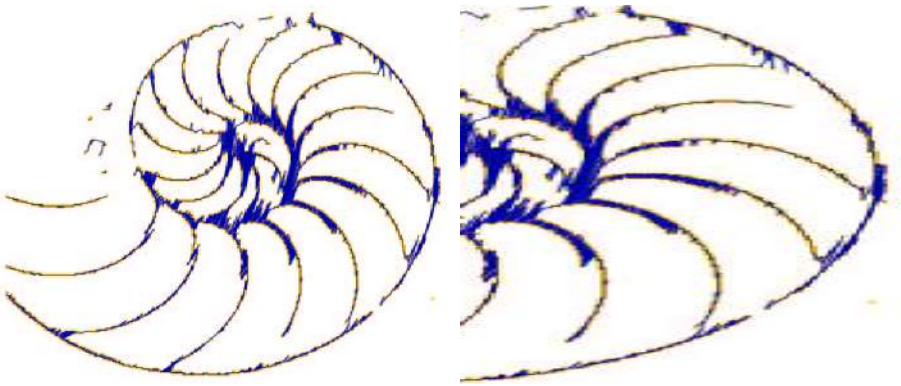
(a) Input trajectories, clusters, and segments (b) Centerlines, refined graph

Figure 3.10: Clustering-Based Map Construction Algorithm (images from [35]).

Ge et al. [44] This algorithm is a point clustering approach that applies topological tools to extract the underlying 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. They provide runtime guarantees as well as partial quality guarantees for correspondences of cycles. An embedding for the edges is then obtained by using a principal curve algorithm [60] that fits a curve to the points contributing to the edge. Figure 3.11 gives an example of a constructed graph based on a point cloud shown as light (yellow) dots.

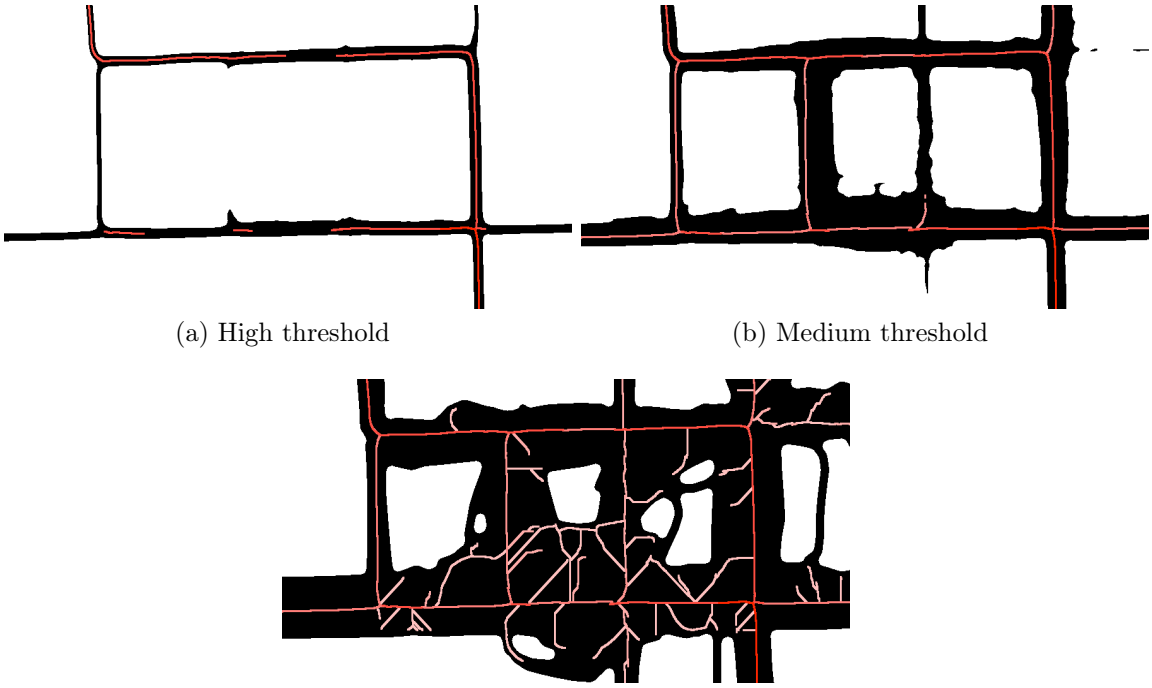
Biagioni and Eriksson [14] Biagioni and Eriksson [14] describe a point clustering-based algorithm that uses KDE methods. Their algorithm proceeds in using KDE with various thresholds to compute successive versions of a skeleton map. They annotate the map by performing a map-matching pass of the input tracks with the skeleton map. Figure 3.12 gives three example stages of the skeleton construction process using high to low KDE thresholds.

Davies et al. [30] 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



(a) Input points and initial graph (b) Graph after smoothing

Figure 3.11: Reeb Graph based Map Construction (images from [44]).

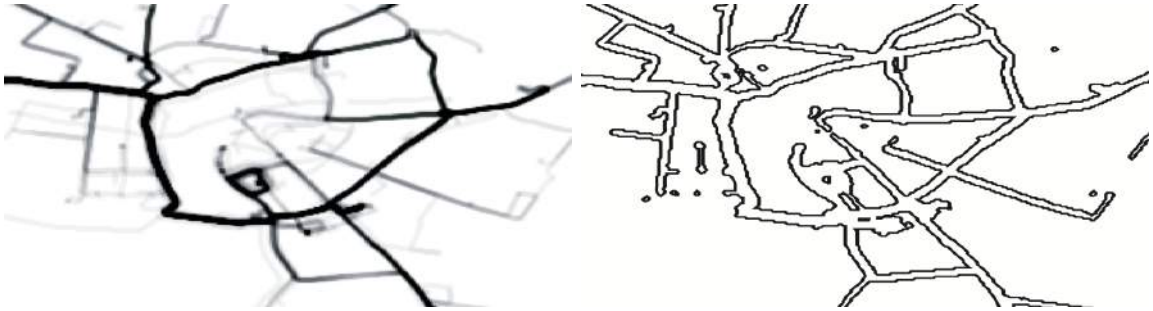


(a) High threshold (b) Medium threshold

(c) Low threshold

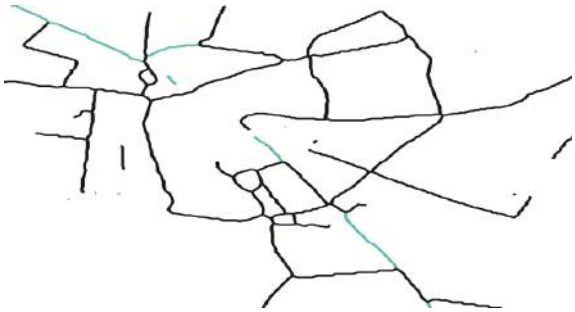
Figure 3.12: KDE-Based Map Construction using Threshold Ranges (images from [14]).

example of Figure 3.13a). Then it computes the contour of the resulting bit map (Figure 3.13b), and then it uses the Voronoi diagram of the contour to compute a center line representation, followed by additional cleanup (Figure 3.13c).



(a) Blurred trajectory histogram

(b) Contours

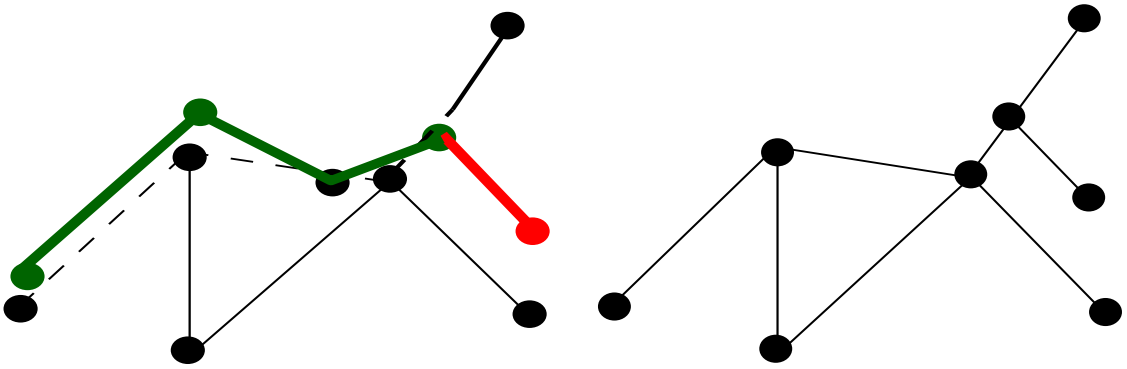


(c) Centerlines, graph

Figure 3.13: Clustering-Based Map Construction Algorithm (images from [30]).

3.3.1.2 Incremental Track Insertion Algorithms

Ahmed and Wenk [7] The algorithm by Ahmed and Wenk [7] is a simple and practical incremental track insertion algorithm. 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 3.14a gives an example of a track with its matched portions shown in dark green and its unmatched portions shown in red. 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 by 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. Figure 3.14b). 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 3.4, 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.

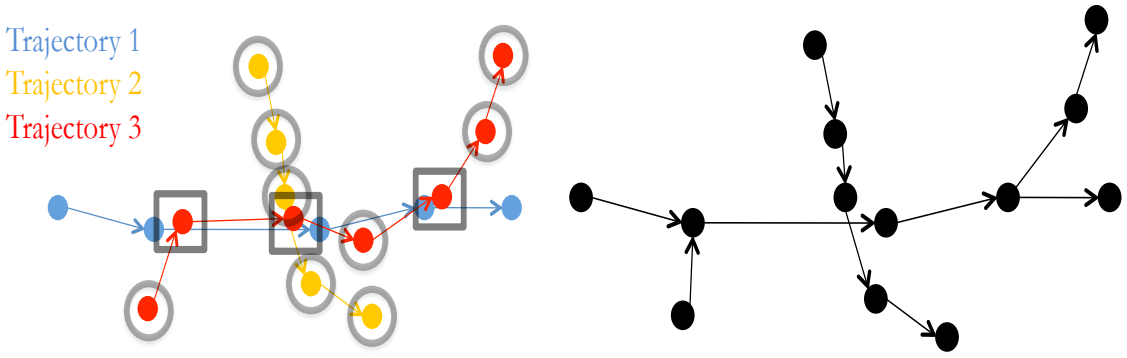


(a) Existing graph and trajectory to be added (b) Merged graph

Figure 3.14: Incremental Track Insertion Algorithm (images from [7]).

Cao and Krumm [21] This incremental track insertion approach 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 different lanes 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 3.15 gives a respective map construction example. The three trajectories of Figure 3.15a are used to incrementally build the graph in Figure 3.15b 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).



(a) Three input trajectories (b) Merged graph

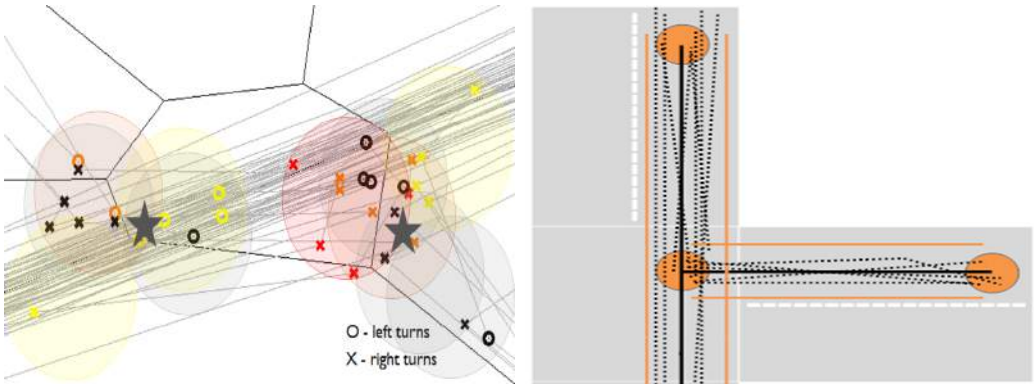
Figure 3.15: Incremental Track Insertion Algorithm (images from [21]).

3.3.1.3 Intersection Linking Algorithms

The *TraceBundle* Algorithm [58] This intersection-linking map construction algorithm is a heuristic approach that "bundles" trajectories around intersection

nodes and is presented in Section 3.1. The main contribution of this *TraceBundle* algorithm is its methodology to derive intersection nodes. The basic heuristic relies on detecting changes in movement and then clustering "similar" nodes. A change in direction and speed is considered a turn indicator. Clustering these turns based on (i) spatial proximity and (ii) turn type results in turn clusters. The centroid location of each of these turn clusters represents an intersection node. Links, and consequently the entire geometry of the map, are generated by connecting the intersection nodes with trajectories, and compacting the trajectories. Figure 3.16 presents the steps of this algorithm. Figure 3.16a shows the constructed intersection nodes as gray stars from turn clusters (x and o markers) and Figure 3.16b shows as black lines the created links after compacting the trajectories. The essential steps of the *TraceBundle* algorithm are as follows:

- *Turn samples* - given a trajectory, each node (position sample) at which a significant change in direction and speed (parameters) occurs becomes a turn sample;
- *Turn clusters* - clustering turn samples based on (i) proximity (*static parameter*) and (ii) a *turn model*;
- *Intersection nodes* - centroid of turn clusters;
- *Connecting intersection nodes* - using constituting turn samples, connect trajectories to respective intersection nodes;
- *Compacting links* - merge connecting trajectory portions between intersection nodes to generate links.



(a) Intersection nodes (b) Compacting links

Figure 3.16: The *TraceBundle* Algorithm.

3.3.2 Quality Measures used for Map Comparison

Here we give some more details on the quality measures that we use in Section 3.4 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 [9], the path-based distance measure presented by Ahmed et al. [5], the distance measure based on shortest paths [58] and graph-sampling based distance measure by Biagioni and Eriksson [13]. The first two measures have not been used for comparative evaluations of road network constructions before.

3.3.2.1 Distance Measures based on Point Sets

Hausdorff Distance [9] 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.

Graph-Sampling Based Distance [13] Biagioni and Eriksson [13] 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 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 and counting the unmatched points in each set. 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". 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 inferred road network with respect to the ground truth according to two metrics. The first metric is the proportion of spurious marbles. This is:

$spurious = spurious\_marbles / (spurious\_marbles + matched\_marbles)$

and the second is the proportion of missing locations (empty holes), where the equation is the following:

$missing = empty\_holes / (empty\_holes + matched\_holes)$.

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

$$F\text{-}score = 2 * \frac{precision * recall}{precision + recall} \tag{3.1}$$

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

3.3.2.2 Distance Measures based on Sets of Paths

Path-Based Distance [5] 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:

$$\delta_F(f,g) = \inf_{\alpha,\beta:[0,1]\to[0,1]} \max_{t\in[0,1]} d(f(\alpha(t)), g(\beta(t))), \tag{3.2}$$

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:

$$\overrightarrow{d}_{C,G}(\pi_C, \pi_G) = \max_{p_C\in\pi_C} \min_{p_G\in\pi_G} \delta_F(p_C, p_G) \tag{3.3}$$

Ideally, π_C and π_G should be the set of all paths in C and G, which however has exponential size. In [5] 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 [8], 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 [5], under some assumptions on C:

- $k = 1$: For each edge in C, there is a path in G which is within Fréchet distance $\overrightarrow{d}_{C,G}(\Pi_C^1, \Pi_G)$.
- $k = 2$: 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.
- $k = 3$: $\overrightarrow{d}_{C,G}(\Pi_C^3, \Pi_G)$ 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]

[1] The degree assumption is only a technical requirement for the theoretical quality guarantees, and the authors have shown [5] that similar approximation guarantees appear to hold in practice as well.

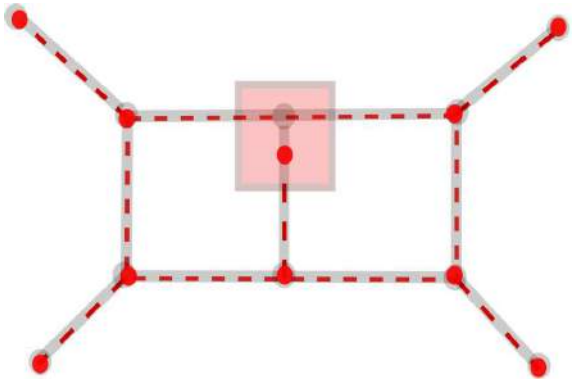


Figure 3.17: Graph G (dotted edges) overlayed on H (gray). G and H differs in the shaded squared region. The distance measure in [14] fails to capture the broken connection in G, as there is always detour available to reach every edge and sample it.

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.

Shortest-Path Based Distance [58] As it was extensively presented in Section 3.2, we proposed a measure that essentially samples each graph using random sets of shortest paths. Given the constructed and ground-truth networks C and G respectively, a common set of node pairs (origin, destination) is selected in both using the nearest neighbor search if necessary. For all node pairs, shortest paths are computed in both networks. The geometric difference/similarity between the respective shortest paths is used to assess the similarity between C and G and consequently as a means to assess 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 inferred network. The more "similar" the shortest paths in the inferred network are to the ground-truth network, the higher also the quality of the network. The results of this shortest path based distance measure can be assessed by plotting the distance of all paths against each other, or by comparing average values for the entire set of paths. We employ both approaches in our experiments below.

3.3.3 Comparison of Distance Measures

All the distance measures described in Section 3.3.2 capture different properties of graphs. Based on the desired kind 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 [70, 58] 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 [9] would be the distance measure to choose.

On the other hand, the two distance measures described in [14] and [5] maximize the use of topology in comparing graphs. Using the concept of *local signature* described in [5] one can visualize the exact differences in graphs using any of these two measures. Figure 3.17 shows an example where the graph sampling based distance [14] 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 [5] exploits every adjacency transition around a vertex, it verifies all connectivities.

Among these four measures only the graph sampling based distance [14] 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.

3.4 Experimental Evaluation

Having devised an algorithm to derive road networks from collected trajectories, this section will showcase various results with a focus on the quality of the inferred road networks. It will also show an extensive evaluation comparison of the most recent map construction algorithms.

3.4.1 Datasets

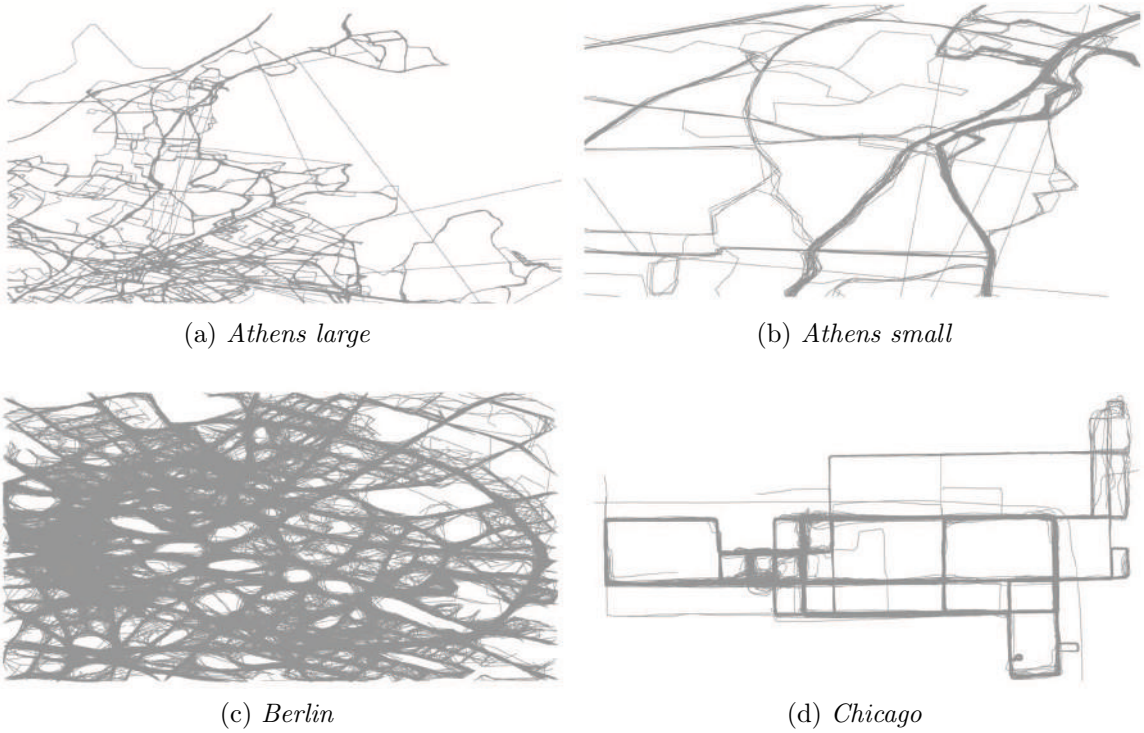
A basic means for assessing map construction algorithms is the underlying dataset comprising vehicle trajectories and ground-truth map datasets. The datasets are in a projected coordinate system (UTM, GGRS87) which are extensively presented in 1.1.1.2. The online repository `http://www.mapconstruction.org` makes available all the visualizations of the datasets. The statistics of the datasets are provided in Table 3.2.

3.4.1.1 Tracking Data

Our experiments use several tracking datasets from three different cities, i.e. Athens, Berlin and Chicago (Figure 3.18). While other publicly available GPS-based vehicle tracking datasets exist, e.g., GeoLife [101] and OpenStreetMap GPX track data [41], the selected range covers the various types of existing datasets produced by different types of vehicles, at varying sampling rates and representing different network sizes.

The *Athens large* dataset consists of 511 trajectories with a total length of 6,781km (mean: 13.27km) obtained from school buses covering an area of 12km $\times$ 14km; the tracks range from 32 to 80 position samples, with a sampling rate of 20s to 30s (mean: 30.14s) and a mean speed of 20.16km/h.

The *Athens small* dataset consists 129 tracks with a total length of 443km (mean: 3.82km) obtained from school buses covering an area of 2.6km $\times$ 6km; the tracks range from 13 to 47 position samples, with a sampling rate of 20s to 30s (mean: 34.07s) and a mean speed of 19.55km/h.



(a) *Athens large* (b) *Athens small*

(c) *Berlin* (d) *Chicago*

Figure 3.18: Tracking Data.

The *Berlin* dataset consists of 26,831 tracks with a total length of 41,116km (mean: 1.53km) obtained from a taxi fleet covering an area of 6km × 6km; the tracks comprise from 22 up to 58 position samples, with a sampling rate of 15s to 127s (mean: 41.98s) and a mean speed of 35.23km/h.

The *Chicago* dataset [13, 14] consists of 889 tracks with a total length of 2,869km (mean: 3.22km) obtained from university shuttle buses covering an area of 7km × 4.5km; the tracks range from 100 to 363 position samples, with a sampling rate of 1s to 29s (mean: 3.61s) and a mean speed of 33.14km/h.

3.4.1.2 Ground-Truth Map Data

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

In *Athens large*, the map consists of 39,699 edges and 32,212 vertices. It covers an area of 12km × 14km. The edges have a length of 2,000km.

In *Athens small*, the map consists of 3,436 edges and 2694 vertices. It covers an area of 2.6km × 6km. The edges have a length of 193km.

In *Berlin*, the map consists of 6,839 edges and 5,894 vertices. It covers an area of 6km × 6km. The edges have a length of 360km.

For *Chicago*, the map covered by the trajectories consists of 11,801 edges and 9,429 vertices. It covers an area of 7km × 4.5km. The edges have a length of 61km.

See Table 3.2 for summary statistics on the ground-truth map data as well as on the tracking data.

Tracking Data	Trajectories	Sampling rate (s)	Trajectory length (km)	Speed (km/h)
Athens large	120	30.14	6,781	20.16
Athens small	129	34.07	443	19.55
Berlin	26,831	41.98	41,116	35.23
Chicago	889	3.61	2869	33.14

OSM Network	Vertices	Edges	Length (km)	Area (km^2)
Athens large	32,212	39,699	2,000	12 × 14
Athens small	2,694	3,436	193	2.6 × 6
Berlin	5,894	6,839	360	6 × 6
Chicago	9,429	11,801	61	7 × 4.5

Table 3.2: Dataset Statistics.

3.4.2 Results

All algorithms have been developed in MATLAB given its impressive high-level routines for statistics including clustering and visualization, essentially allowing us to focus on core algorithms and data structures. The experimental setup in Figure 3.19 shows the methods developed in order to automatically derive and assess the inferred road network.

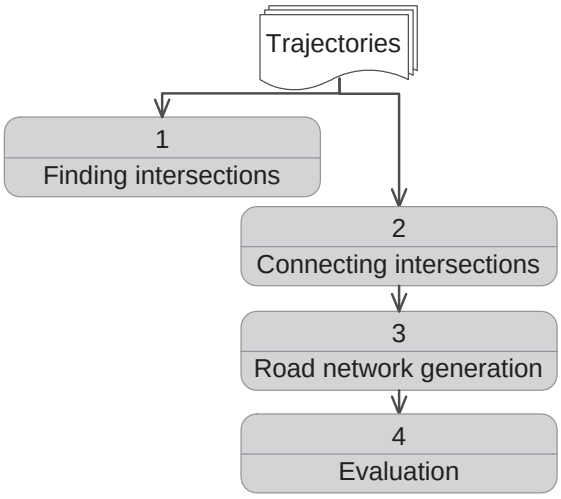


Figure 3.19: Experimental Setup.

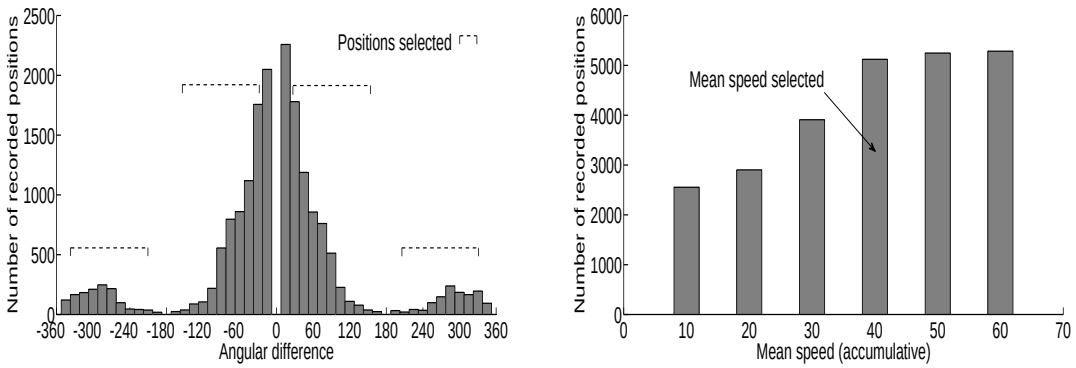
A series of initial experiments established the proper parameter setting for our road inference algorithm. Turn clustering and intersection inference (cf. Section 3.1.1) employs four parameters. For turn clustering, (i) the angular difference threshold was set to 15°. Figure 3.20a gives the histogram and the selected angle differences (dashed areas) for all trajectory samples. As typically vehicles move straight ahead, do not change their movement by more than 15°. If they do and in combination with (ii) the mean speed threshold of 40km/h that vehicles may experience while turning, then we consider this as an indication for a turn. Both measures have been chosen very conservatively as false positives will be eventually be eliminated and they should not be backed up by additional samples. As vehicles transporting persons are allowed to drive with up to 40km/h in residential areas, this can be also realized in Figure 3.20b. For instance, the selected mean speed of 40km/h conveys the transportation conditions of the tracking data.

In addition, (iii) a maximum time constraint of 35s and a distance threshold of 50m was used to cluster position samples. With respect to clustering turn samples

Turns and Intersections	
Angular difference	15°
Mean speed	40km/h
Time constraint	35s
Turn clusters	50m
Intersection nodes	25m
Compacting Links	
Direction threshold	45°
Shortest-Path Based Distance	
Actual road network (bounding box)	50m

Table 3.3: Parameter Summary.

into intersection nodes, (iv) a 25m distance threshold was used. The *TraceBundle* algorithm (cf. Section 3.1) uses a bounding box that encloses candidate road network portions. While the size of the bounding box is dynamically established, we initialize it with a maximum width of 20m. All parameters were established empirically by running a great number of road network inference experiments and assessing the quality of the respective results. The above parameters represent the best setting for the specific case.



(a) Angular difference (b) Mean speed

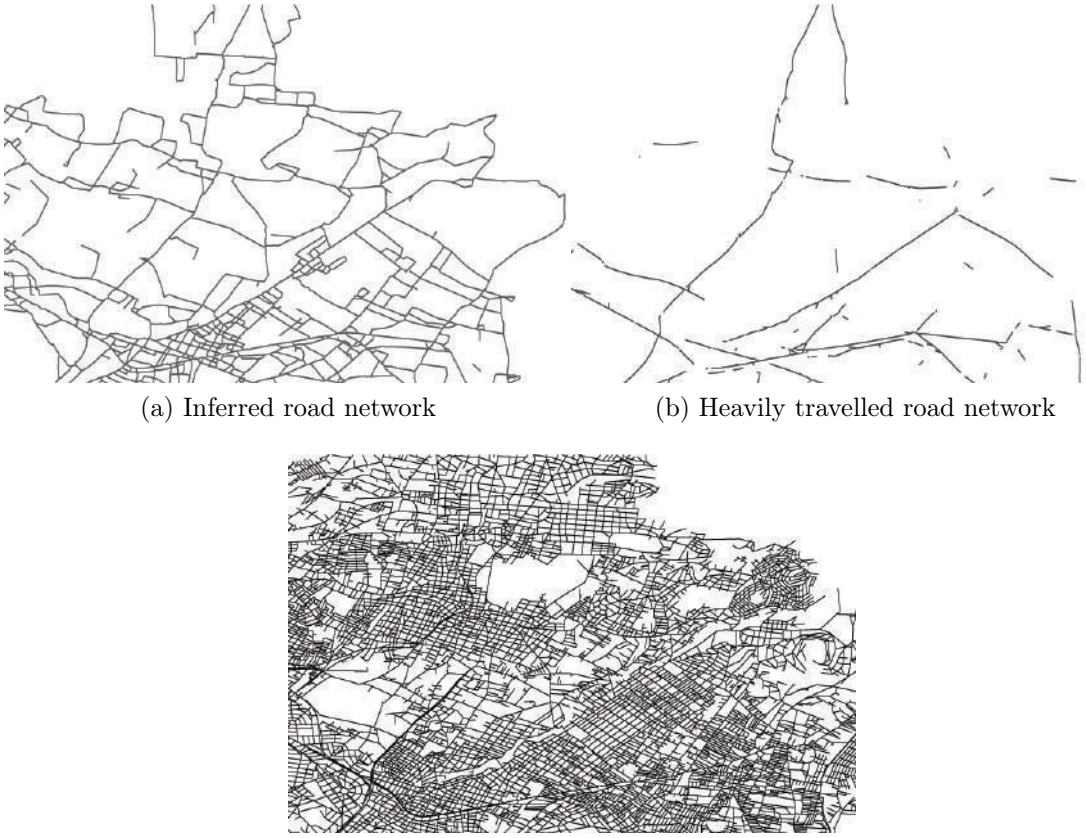
Figure 3.20: Parameters Selection.

In what follows, we describe the results of the *TraceBundle* algorithm applied to trajectory data covering a portion of the road network in Athens, Berlin and Chicago.

To give some insights on the elimination of the tracking data that the *TraceBundle* algorithm following the various stages brings to the inferred result, we present in terms of numbers the following output for the dataset of *Athens large*. During the first phase, i.e., intersection extraction and connection, 4995 intersection nodes and 5983 link samples are generated. All links combined have a length of 2700km. The second road network generation phase produces 5124 intersection nodes, 6219 links and a length of 710km. This result shows that during the second phase of the algorithm, the number of nodes remains largely constant but only the length of the links connecting them is significantly reduced since we radically merge links in this phase.

The overall time to compute these results in MATLAB/Windows7 on an Intel Core2Duo processor running at 2.2GHz is 56mins. This time is achieved without performing any optimization on the data structures used in the implementation.

Figure 3.21a visualizes the inferred road network. In addition, Figure 3.21b illustrates how the computed link weight information can be used to identify major roads of the network. In this visualization, links are shown that are traversed at least 10 times. The gaps in the road network are due to uneven distribution of weights during stage two of the algorithm when compacting the road network and the untypical traversal of roads by school buses in the *Athens large* dataset, i.e., turning off major roads to drop of kids.



(a) Inferred road network

(b) Heavily travelled road network

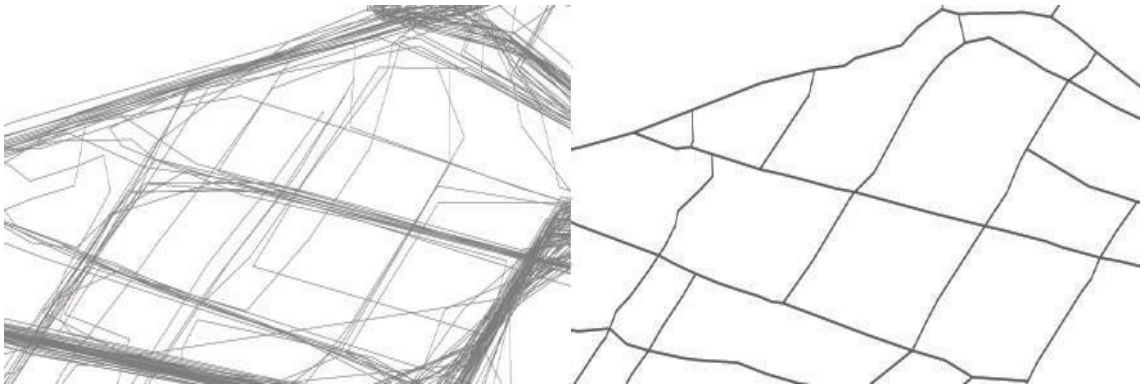
(c) Original road network

Figure 3.21: Inferred Road Network - The *TraceBundle* Algorithm - (*Athens large*).

Zooming in on the *Athens large* data, Figure 3.22 shows the tracking data, the inferred road network, and the actual road network of a smaller area. At this scale, it can be clearly seen that the traversed portion of the road network was correctly identified.

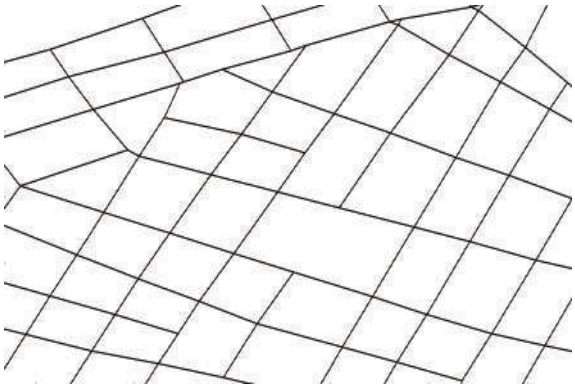
3.4.3 Evaluation Comparison of Map Construction Algorithms

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*. The seven algorithms used in this experimentation are implemented in C, Java, Python and MATLAB. The experiments



(a) Raw data

(b) Inferred road network



(c) Original road network

Figure 3.22: Inferred Road Network - The *TraceBundle* Algorithm - Close Ups.

for six algorithms have been performed by the authors and the implementations have been made available at the `mapconstruction.org` web site. The authors of [44] 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, e.g., 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 $10min$ to $20h$. For the larger *Berlin* dataset, the running times range from $2h$ to $4days$. 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. 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*.

3.4.3.1 Constructed Maps

What follows is an initial overview of the experimentation in terms of constructed maps and the respective result quality. Figure 3.23 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 3.3.1.1, 3.3.1.2 and 3.3.1.3. Figure 3.24 illustrates the ground-truth map (light gray) and the generated maps (black) for the case of the larger *Berlin* dataset.

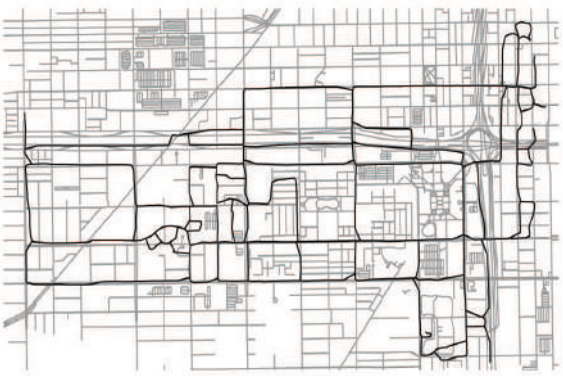
Each of the algorithms uses different parameter settings. For Ahmed and Wenk [7] the values of ε to cluster subtrajectories are: 180, 90, 170 and 80 meters for *Athens large*, *Athens small*, *Berlin* and *Chicago*, respectively. The respective parameters of *proximity* and *bearing* for the other algorithms are Biagioni $50m$ [14], Cao $20m$ and 45° [21], Davies $16m$ [30] and Edelkamp $50m$ and 45° [35]. For the *TraceBundle* algorithm [58] the values of *direction*, *speed* and *proximity* to extract intersection nodes and to merge trajectories into links are 15°, 40km/h and $25m$ accordingly (Table 3.3). We evaluated all constructed maps using the distance measures described in Section 3.3.2.

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

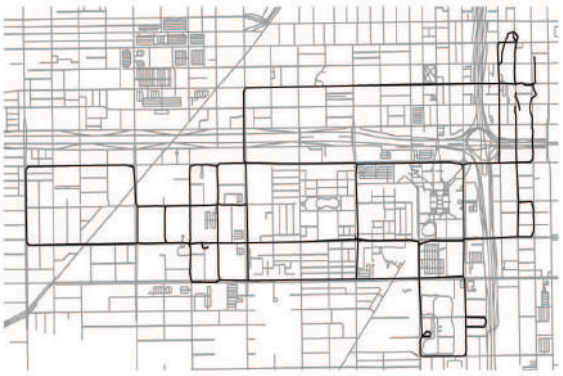
Table 3.4: Complexities of the Map Construction Algorithms.

A summary of the complexities of the constructed maps is shown in Table 3.4. 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. [13, 14] and Davies et al. [30] 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. [44] subsample all tracks to create a much denser output set, hence the complexity of their constructed maps is always higher.

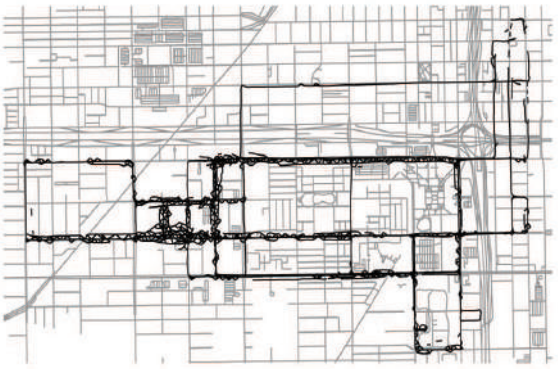
Map construction algorithms based on *incremental track insertion*, such as Ahmed et al. [7] and Cao et al. [21] fail to cluster tracks together when the variability and error associated with the input tracks is large. As a result, the constructed street maps contain *multiple edges for a single street*, which implies larger values in the total edge length column in Table 3.4.



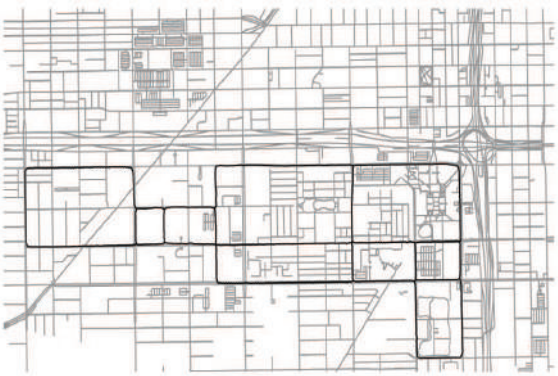
(a) Ahmed - *Chicago*



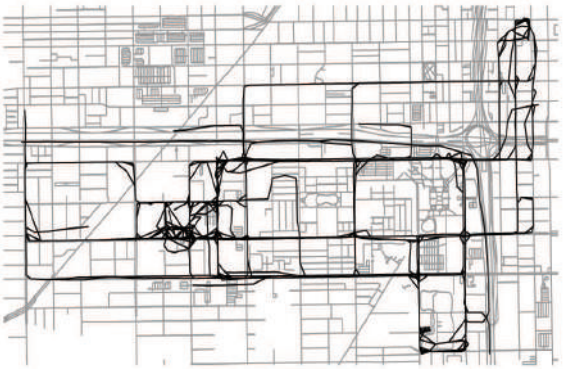
(b) Biagioni - *Chicago*



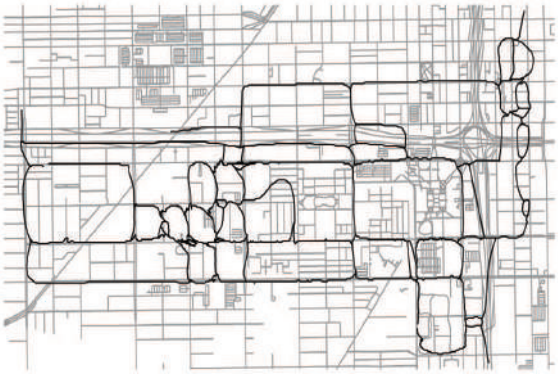
(c) Cao - *Chicago*



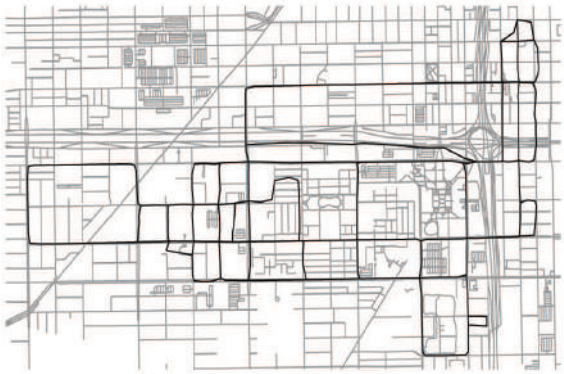
(d) Davies - *Chicago*



(e) Edelkamp - *Chicago*

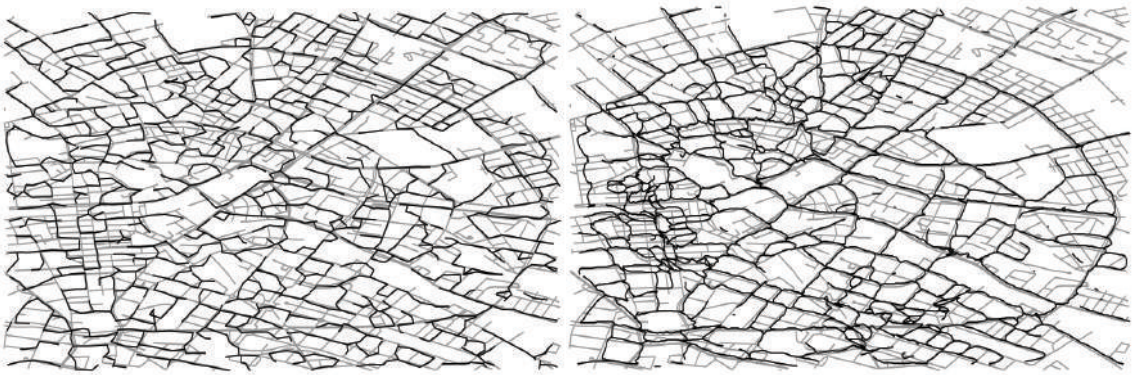


(f) Ge - *Chicago*



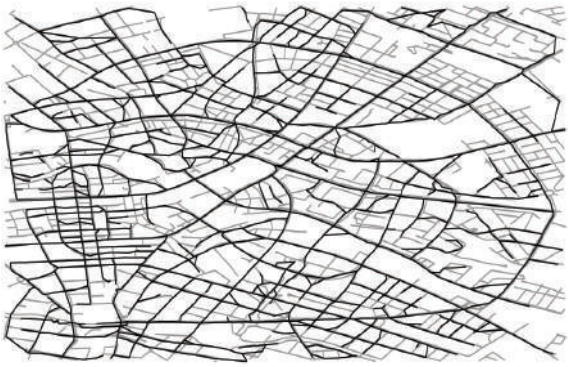
(g) Karagiorgou - *Chicago*

Figure 3.23: Constructed Maps - in black - Overlayed on Ground-Truth Map - in gray - (small dataset).



(a) Ahmed - *Berlin*

(b) Ge - *Berlin*



(c) Karagiorgou - *Berlin*

Figure 3.24: Constructed Maps - in black - Overlayed on Ground-Truth Map - in gray - (large dataset).

Several examples of generated maps are shown in Figure 3.23 and Figure 3.24. Since not all algorithms produced results for all maps, we showcase examples of the smaller *Chicago* map in Figure 3.23. 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 Figure 3.24.

3.4.3.2 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 3.5. 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 constructed using the *TraceBundle* algorithm [58] and by Biagioni et al. [13, 14] generally have a better path-based distance than the others. Note that Davies et al.'s [30] 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.

Generated Map	Path based distance (m)								Directed Hausdorff distance (m)							
Athens large	min	max	median	avg	2%	5%	10%	15%	min	max	median	avg	2%	5%	10%	15%
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	min	max	median	avg	2%	5%	10%	15%	min	max	median	avg	2%	5%	10%	15%
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	min	max	median	avg	2%	5%	10%	15%	min	max	median	avg	2%	5%	10%	15%
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	min	max	median	avg	2%	5%	10%	15%	min	max	median	avg	2%	5%	10%	15%
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

Table 3.5: Path-Based and Directed Hausdorff Distance Measure Evaluation.

For further analysis of the results, we selected the *Chicago* dataset as all map construction algorithms produced results for it. From Table 3.5 one can see that the path-based distance and the Directed Hausdorff distance are smaller for the generated maps by Biagioni, Davies and the *TraceBundle* algorithm (shaded gray) compared to map generated using other algorithms. A visual inspection of the maps in Figure 3.23 justifies the result. Note that Davies et al.'s [30] map is comparatively smaller than the other, see Table 3.4. 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 Figure 3.25 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 $71m$. For the same path, the Hausdorff distance is $53m$, 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.

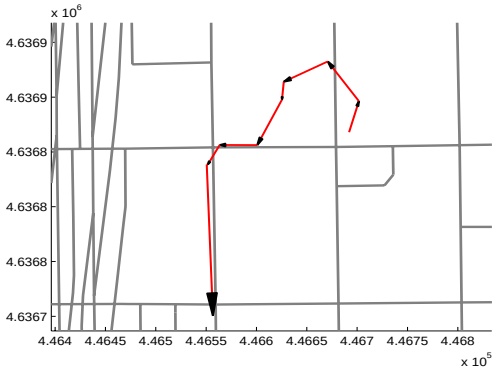
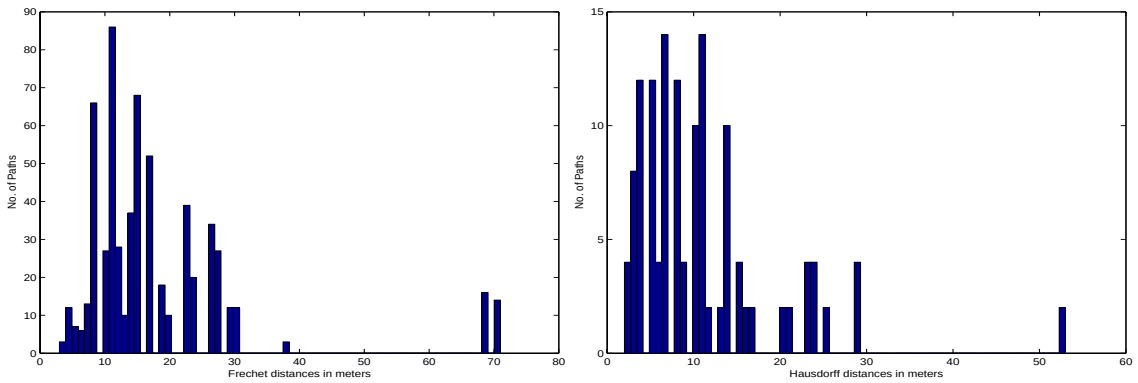


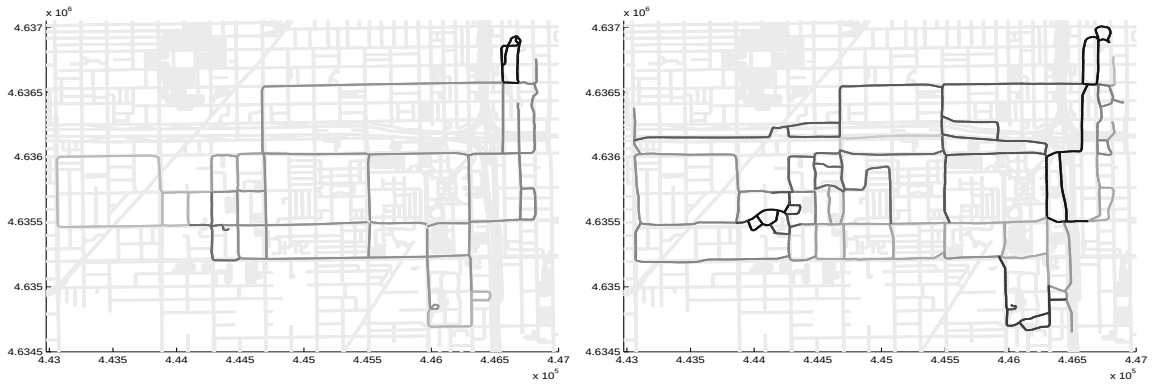
Figure 3.25: A path with Fréchet distance greater than Hausdorff distance.



(a) Path-Based Distance (b) Directed Hausdorff

Figure 3.26: Distributions of Individual Path Distances (Biagioni alg. - *Chicago*).

In addition, if desired one can discard outliers by computing the $d\%$-distance. Figure 3.26 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 $71m$ (max) to $38m$ and the Directed Hausdorff distance from $53m$ (max) to $25m$. Figure 3.27 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.



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

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

Figure 3.27: 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.

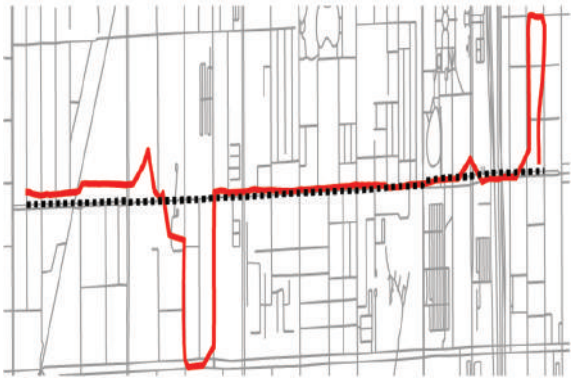
3.4.3.3 Shortest-Path Based Measure

Assessment of the *TraceBundle* Results 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 uniformly distributed over the maps and compared the paths using the Discrete Fréchet and Average Vertical distance measure. This approach enables to evaluate the quality of the constructed maps in terms of road network accuracy and connectivity.

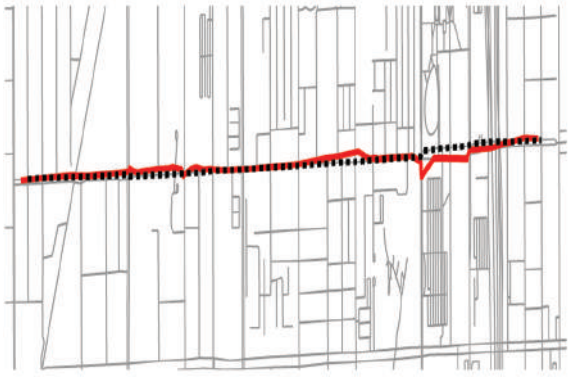
In our case, the quality of the generated result is expressed by the distance measure between computed shortest-paths in the constructed and the actual road network. We give an example of how the distance measure is applied in the case of the *Athens large* dataset. In *Athens large* dataset, Figure 3.29a displays an excerpt of the network coverage of the randomly created routes. Also, Figure 3.29b gives an example of how similar two randomly created routes are in the actual (light grey) and in the generated (dark grey) road networks.

Besides visual inspection, we need to come up with an indicator for dissimilarity and a way to assess the quality of the inferred network. Dissimilarity can be expressed as an increasing distance between the two paths. Figure 3.30 shows the Discrete Frèchet distance (in light grey) and Average Vertical distance (in dark grey) of the 500 computed paths, in the *Athens large* dataset. Combining visual inspection with path distance, it was empirically established that a Discrete Frèchet distance greater than 300m is an indicator of such dissimilarity. In other words, such distances are an implicit indication for dissimilarity between the inferred and the actual road network.

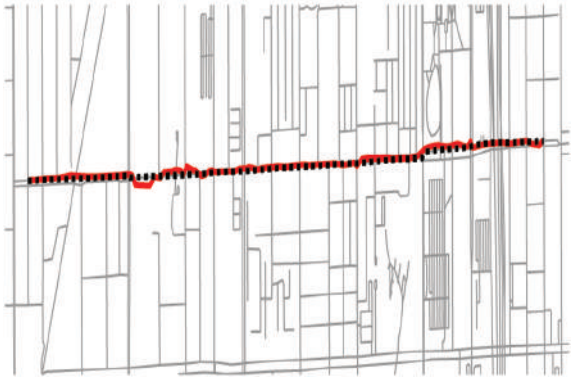
Shortest-path pairs exceeding this distance use portions of the inferred road network with connectivity problems that do not exist in the actual road network. Connectivity problems are mostly due to falsely created links. In the case of *Athens large*, in our experimentation, we found that 5% of the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



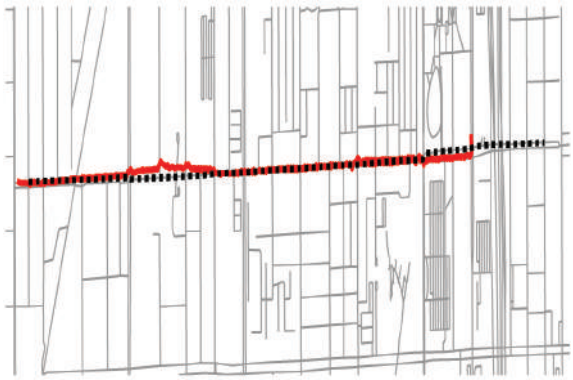
(a) Ahmed



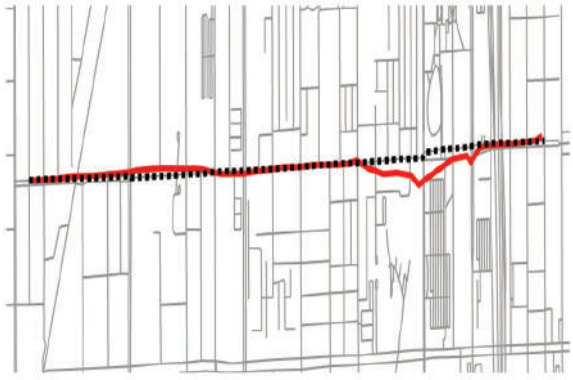
(b) Biagioni



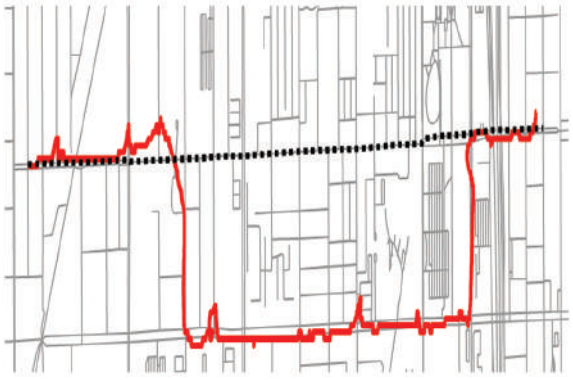
(c) Cao



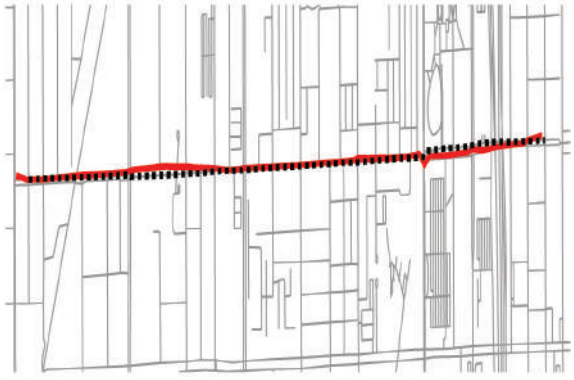
(d) Davies



(e) Edelkamp

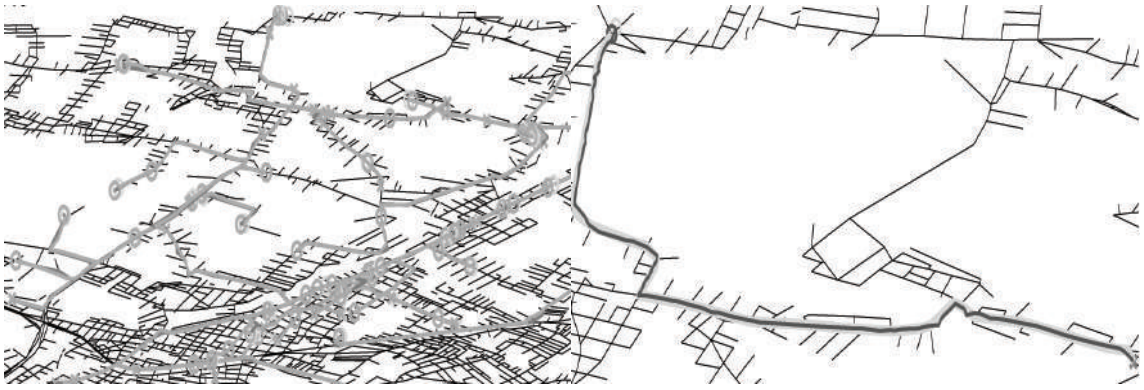


(f) Ge



(g) Karagiorgou

Figure 3.28: Shortest-Path Examples - *Chicago* Dataset.



(a) Network coverage (b) Links similarity

Figure 3.29: Assessing Inferred Road Network.

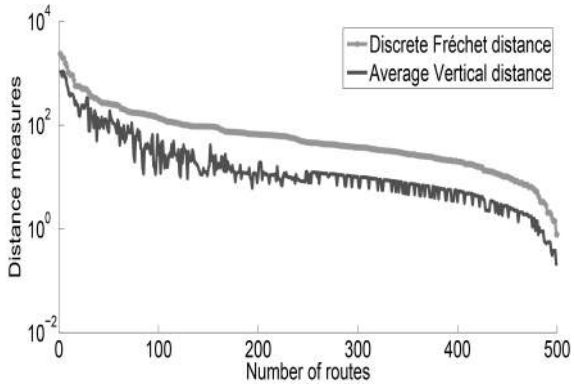


Figure 3.30: Similarity Results.

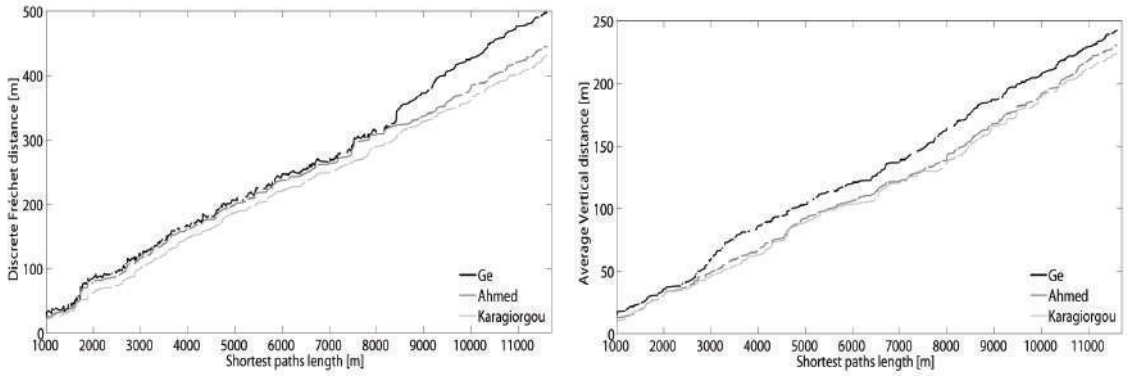
the destination nodes accounting for distance measures up to 300m.

Having a closer look at some of the 5% cases, we can observe two problems for the inferred network; (i) spatial accuracy and (ii) connectivity. Figure 3.31a shows an example in which due to the spatial accuracy, an alternative route was computed. In this case, the inferred network simply produced a slightly different geometry that made the shortest-path algorithm choose a partially different route. A more serious problem is that of connectivity. Figure 3.31b illustrates this problem, which results in different routes due to missing links. In both visualizations, the route of the actual road network is shown in black, while the route of the inferred road network is grey.

Assessment of Map Construction Algorithms We now give an example of how different constructed maps affect such paths in Figure 3.28 for all the compared algorithms using the *Chicago* dataset. Given a specific origin and destination for the *Chicago* map, the shortest path has length $3.66km$ in the ground-truth map (black dotted line). The computed shortest path for the map generated by each algorithm is shown in red line. In the map generated by Ahmed et al.'s algorithm the shortest path has length $4.67km$ (a Discrete Fréchet distance with respect to the ground-truth map of $65m$, and an Average Vertical distance of $21m$). The respective results for the other algorithms are Biagioni $3.71km$ ($36m$, $5m$), Cao $3.76km$ ($24m$, $6m$), Davies $3.39km$ ($35m$, $4m$), Edelkamp $3.64km$, ($26m$, $8m$), Ge $7.33km$, ($174m$, $98m$), and the

(a) Spatial accuracy (b) Links connectivity

Figure 3.31: Link Failures.



(a) Discrete Fréchet distance - *Athens large* (b) Average Vertical distance - *Athens large*

Figure 3.32: Map Comparison.

TraceBundle algorithm $3.73km$ ($21m$, $5m$). For most algorithms the resulting paths have small distance to the shortest path in the ground-truth map. However, in the case of Ahmed (Figure 3.28a) and Ge (Figure 3.28f), due to significant differences in the generated map, different shortest paths have been computed that have a larger distance when compared with the shortest path in the ground-truth map. This result is in line with the path-based measure of Section 3.2, where also Biagioni, Davies and the *TraceBundle* algorithm produced the best constructed maps.

Figures 3.32a and 3.32b show the Discrete Fréchet 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 connectivity 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 3.6. 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 (shaded light gray) in Table 3.6, this algorithm produces a map that in turn generates paths that have a min, max, and avg. Discrete Fréchet distance of $21m$, $469m$, and $192m$, 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* (shaded light gray in Table 3.6). 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, the *TraceBundle* algorithm produces maps that 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*.

Generated Map	Found (%)	Discrete Fréchet dist. (m)				Average Vertical dist. (m)				Shortest path dist. (km)			
Athens large		min	max	avg	stddev	min	max	avg	stddev	min	max	avg	stddev
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		min	max	avg	stddev	min	max	avg	stddev	min	max	avg	stddev
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		min	max	avg	stddev	min	max	avg	stddev	min	max	avg	stddev
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		min	max	avg	stddev	min	max	avg	stddev	min	max	avg	stddev
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

Table 3.6: Shortest-Path Based Measure Evaluation Summary.

3.4.3.4 Graph-Sampling Based Distance

For this measure we use the source code obtained from the authors of [13]. We modified the code to use Euclidean distance as our data uses projected coordinate system. The method that computes this measure has four parameters: 1. *sampling density*, how densely the map should be sampled (marbles for generated map and holes for ground-truth map), we use $5m$; 2. *matched distance*, the maximum distance between a matched marble-hole pair, we vary this distance from $10m$ to $120m$; 3. *maximum distance from root*, the maximum distance from randomly selected start location one will explore, we use $300m$; 4. *number of runs*, number of start locations to consider, we use $1,000$. 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 [13], but still the relative comparison of F-score values should provide an idea of whether an algorithm performs better than another.

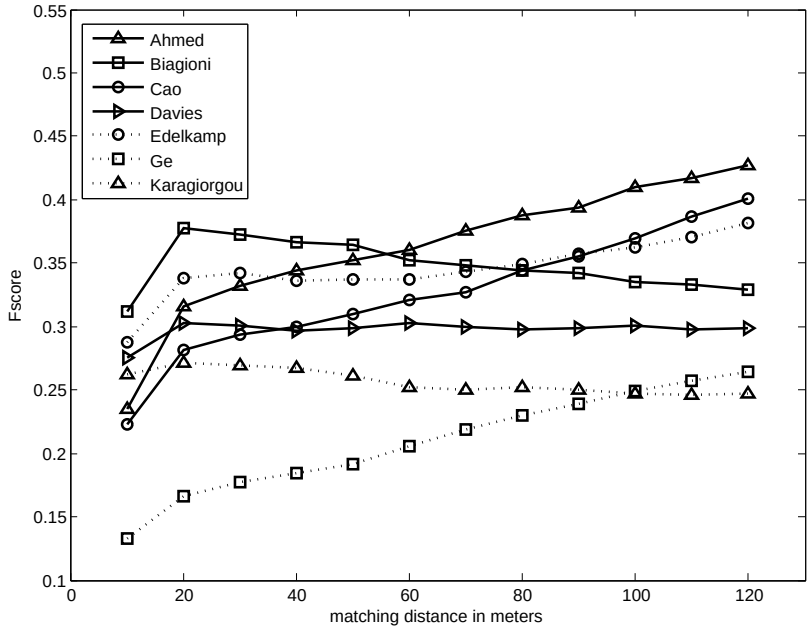


Figure 3.33: Comparison of F-scores - *Chicago*.

Figure 3.33 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, 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 3.7). 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. Figure 3.23 and Figure 3.24).

In Figure 3.33, we also see the performance based on F-score declines for Biagioni, Davies and the *TraceBundle* algorithm as the matching distance threshold increases. After investigating the reason of this unexpected behaviour we found, although precision increases with matching distances the recall declines for these three algorithms; and smaller recall indicates larger number of unmatched sample points on ground-truth (empty holes). Figure 3.23 and Table 3.4 show these three algorithms reconstruct less streets than others, which means they produce smaller

Generated Map	Precision Value (for *matched distance* 10, 40, 70, 100)			
Athens large	10	40	70	100
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	10	40	70	100
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	10	40	70	100
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	10	40	70	100
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

Table 3.7: Precisions for Varying *Matched Distance*.

number of marbles to match with larger number of holes.

Hence, in Table 3.7 we are ignoring F-score and recall values and *showcase only precision values*. According to precision values, the algorithms by Biagioni and Davies and as well as the *TraceBundle* algorithm perform best for dataset *Chicago*, which is consistent with our findings using the other three distance measures.

3.4.3.5 Summary of Map Comparison

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 *TraceBundle* algorithm and the maps reconstructed using the algorithms by Davies 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 time only limited coverage. In this evaluation, the *TraceBundle* algorithm produces maps that have both good coverage and high path similarity.

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

3.5 Summary

In this chapter, we have addressed the problem of automatic road network inference from tracking data and we have proposed a novel approach to road network inference from GPS traces. In an nutshell, the algorithm exploits changes in movement patterns by using turns as a means to identify intersection nodes. Intersection nodes are effectively used to bundle vehicle trajectories and links between derived intersections by merging them into a single geometry. In addition, we presented a method to assess the quality of the inferred road network based on comparing computed shortest-paths by means of distance measures and an extensive evaluation comparison of map construction algorithms.

Besides, this chapter has considered a variety of such map construction algorithms. In the past, the lack of benchmark data and quantitative evaluation methods has hindered a cross-comparison between algorithms. 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 `mapconstruction.org` web site.

Overall, the *TraceBundle* algorithm produces road networks that very closely resemble the actual road network provided sufficient tracking data are available. Redundancy in coverage increases the quality of the road network.

Chapter 4

Dealing with Noisy Tracking Data

The commoditization of tracking technology, e.g., smartphone applications involving check-ins, real-time navigation applications, fleet management, etc., provides us with a considerable tracking data source that enables us to derive not only road networks, but transportation networks in general.

In Chapter 3, we have addressed the problem of automatic road network inference using redundant tracking data. However, the global approach that the *TraceBundle* algorithm adopts can not cope with sparse, noisy and low sampled tracking data. Consider the case of a vehicles fleet where the position samples are not recorded at a regular time and a high frequency. As a consequence, an approach which uses global parameters for clustering and merging of geometries can not be efficient in tracking data with such characteristics.

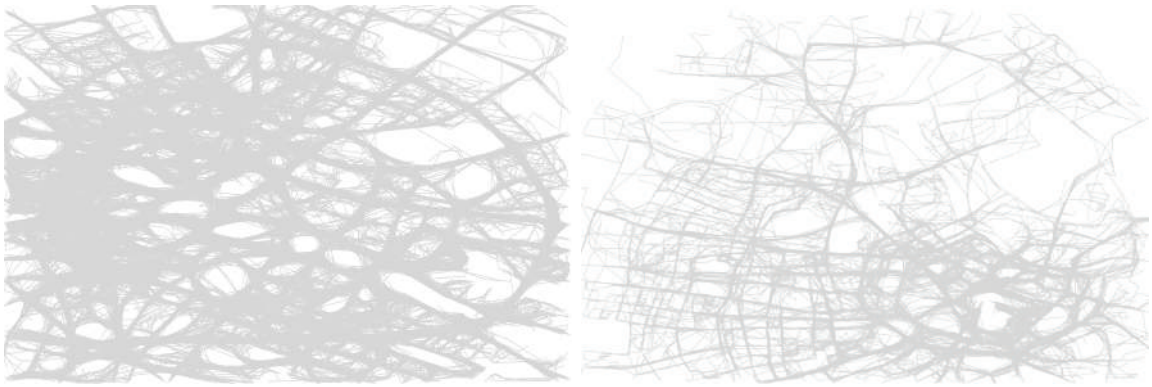
In this chapter, we deal with a novel methodology that converts movement trajectories into a hierarchical transportation network, in order to derive persistent knowledge from tracking data. We do so by utilizing an improved map inference algorithm on segmented input data based on types of movement. This produces hierarchical road network layers, which are then combined into a single network. The steps of this hierarchical approach for the automatic road network inference constitutes the *TraceConflation* algorithm.

Segmentation also addresses the challenges imposed by noisy, low-sampling rate tracking data and provides for a mechanism to accommodate automatic map maintenance on updates.

An experimental evaluation assesses the quality of the *TraceBundle* and *TraceConflation* algorithms by constructing the road networks for large parts of Berlin, Vienna and Athens. The data used are trajectories derived from GPS tracking taxi fleets and utility vehicles. Our results show significant improvements in terms of quality of the constructed road network over existing approaches.

The remainder of this chapter is organized as follows. Section 4.1 presents our algorithms for trajectories segmentation and re-association to build the layered network map, supporting updates. These algorithms constitute the *TraceConflation* algorithm. In Section 4.2, we present a quality evaluation of the segmentation-based road network inference method, comparing our results to the *TraceBundle* algorithm. Finally, Section 4.3 presents some conclusions regarding this chapter.

Our results in this chapter have been published in [59].



(a) Berlin (b) Vienna

Figure 4.1: Tracking Data.

4.1 The *TraceConflation* Algorithm

To efficiently solve the map construction problem in the case of sparse and low sampled data, we propose the *TraceConflation* algorithm. Here, the road network is constructed in a layered fashion based on segmenting the trajectory data using speed categories. The process involves three steps: (i) *segmentation of trajectories*, i.e., splitting the input dataset of trajectories into subsets of (sub-)trajectories according to their characteristics, (ii) *construction of the network layers*, i.e., processing each subset to identify nodes and edges of the network, and (iii) *conflation of the network layers*, i.e., merging the generated layers to produce the complete map of the road network. As it can be seen in Figure 4.1, inferring the road network from sparse tracking data is not a trivial task.

4.1.1 Segmentation of Trajectories

A main challenge when inferring a movement network from raw GPS traces is that these data are often noisy and heterogeneous (GPS errors, missing values, different sampling rates, different speeds, etc.). Thus, treating all the input data equally, inevitably introduces inaccuracies in the 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 refining the parameters of the map inference algorithm accordingly. The aim is to derive different (but probably overlapping) portions of the network with higher accuracy, which then need to be merged in order 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 classify trajectories accordingly. Notice that typically an object may have moved with different speeds across different parts of the trajectory, in which case the trajectory needs to be split into sub-trajectories, with each one 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 length of the time interval of its start and end points. Then, each segment is assigned to the corresponding speed category.

However, this often leads to a high degree of fragmentation, rendering the dataset unusable. Indeed, when a vehicle moves, it may often slow down, for example, due to an intersection, or a traffic light, or some other obstacle. To avoid excessive splitting of trajectories due to such abrupt changes of short duration, we apply a *sliding window* across the trajectory, replacing the speed value of each segment by the mean value computed over a series of consecutive line segments around it (Algorithm 4, Line 7). Then, splitting and classification of sub-trajectories is done according to these "smoothened" speed values.

The process is outlined in Algorithm 4. For each line segment L_j of each trajectory T, its mean speed is computed over a sliding window of width $2 \cdot w$ (Algorithm 4, Line 7), and the segment is then assigned to the corresponding class according to the min and max speed of each category. Lines 7-8 in Algorithm 4 capture this part of the algorithm.

Algorithm 4: Segmentation of Trajectories

Input: A set of trajectories T
Output: A set of segmented trajectories C

```
1  begin
2      /*Trajectories segmentation according to speed profiles*/
3      for (T_i ∈ T) do
4          for (L_j ∈ T_i) do
5              v̄(L_j) ← Mean(v(L_{j−w}), ..., v(L_{j+w}))
6              if v̄(L_j) ∈ C then
7                  if v̄(L_i) ∈ [C_min, C_max] then
8                      C ← L_i
9                  end
10             end
11         end
12     end
13 end
```

4.1.2 Construction of Network Layers

The next step is to use the trajectories classified in each category to infer a layer of the road network. This is done using an improved version of the *TraceBundle* algorithm we had previously introduced in Chapter 3. In the following, we first explain briefly the basics of this algorithm and then we describe some improvements introduced here.

4.1.2.1 The *TraceBundle* Algorithm

This trace-based map construction algorithm employs heuristics to identify intersection nodes and "bundle" trajectories around them. The basic idea of the *TraceBundle* algorithm is how it infers intersection nodes. This relies on detecting *changes in movement* and then clustering them. Such changes represent turns and are identified as changes in direction and speed. Clustering these turns based on (i) spatial proximity and (ii) turn type results in *turn clusters*. The centroid location of each of these turn clusters represents then an *intersection node*. Connecting the trajectories to intersection nodes and compacting them allows one to derive links and consequently the entire geometry of the road network.

The essential steps of the *TraceBundle* algorithm are outlined below:

- *Identification of turn samples*: Given a trajectory, each node (position sample) at which a significant change in direction and speed occurs becomes a turn sample;

- *Construction of turn clusters*: Turn samples are clustered based on proximity and a turn model;

- *Creation of intersection nodes*: The centroid of each turn cluster is computed and marked as an intersection node;

- *Linking of intersection nodes*: Based on the turn samples from which a turn cluster was derived, the respective trajectories are connected to intersection nodes;

- *Compacting links*: The portions of trajectories connecting two intersection nodes are compacted to create a single link.

4.1.2.2 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*. In *TraceBundle*, static parameters are used for both. However, since different types of roads and intersections exist in a road network, such a setting often results in erroneous clusters, e.g., generating multiple nodes for a single intersection or generating a single node for multiple nearby intersections.

To overcome this problem, we have replaced the clustering performed by *TraceBundle* with a *proximity-based expansion algorithm around turn samples based on turn similarity*. The essence of the algorithm is shown in Algorithm 5. A segmented set of trajectories, i.e., a set of trajectories (segments) that belongs to the same speed category, is the input to the intersection construction algorithm.

In a first step, all position samples are evaluated as to whether they represent turn samples based on a change of direction. Respective position samples are then added to the set of turn samples. The data recorded includes also the incoming and outgoing direction of the motion captured by the trajectory with respect to the specific turn sample. Lines 10 - 16 in Algorithm 5 capture this part of the algorithm.

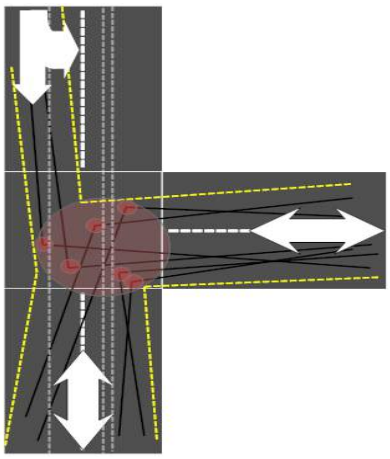
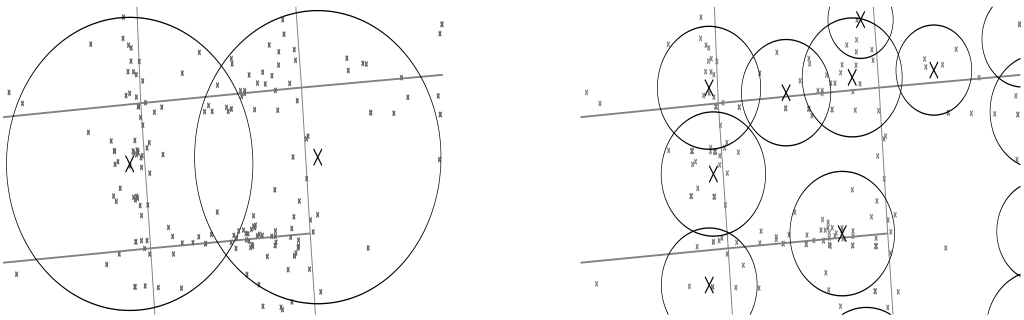


Figure 4.2: Turn Samples Clustering.

This information is in the next stage, the derivation of turn clusters, used to compute the directional similarity of turn samples. Samples that show a similar motion in terms of absolute direction and that are spatially close are grouped together into turn clusters. The turn clusters are constructed bottom up, i.e., by evaluating all recorded turn samples one at a time, for each the set of nearest-neighbor samples considering direction similarity is retrieved. The cardinality of each set depends on the number of turn samples existing within a threshold distance d_{max}, which was set to $25m$ in our experiments, of the turn sample in question. Experiments showed that many turn clusters effectively have a radius much less than d_{max}, since turn samples of similar direction are either clustered together or much further away (relating to different intersections). The turn clustering approach is captured in Lines 20 - 23 of the algorithm.

Turn clusters stemming from different movement directions (left turn vs. right turn), but relating to spatially the same intersection now need to be grouped together to produce one node in the road network graph rather than several nodes representing the various clusters. Our approach to group turn clusters now relies purely on spatial properties. Intersections are derived by scanning all turn clusters with respect to spatial coverage, i.e., smaller clusters if contained, will be absorbed by larger ones. Experimentation has shown this to be a very effective approach. The pseudocode of Lines 26 - 28 summarizes this approach.

Figure 4.2 illustrates the outcome of this approach by contrasting an output from the *TraceBundle* algorithm with the current approach. Figure 4.3a shows how the clustering using static parameters *TraceBundle* erroneously places nodes between actual intersections. Figure 4.3b shows the current approach with nodes being placed more accurately.

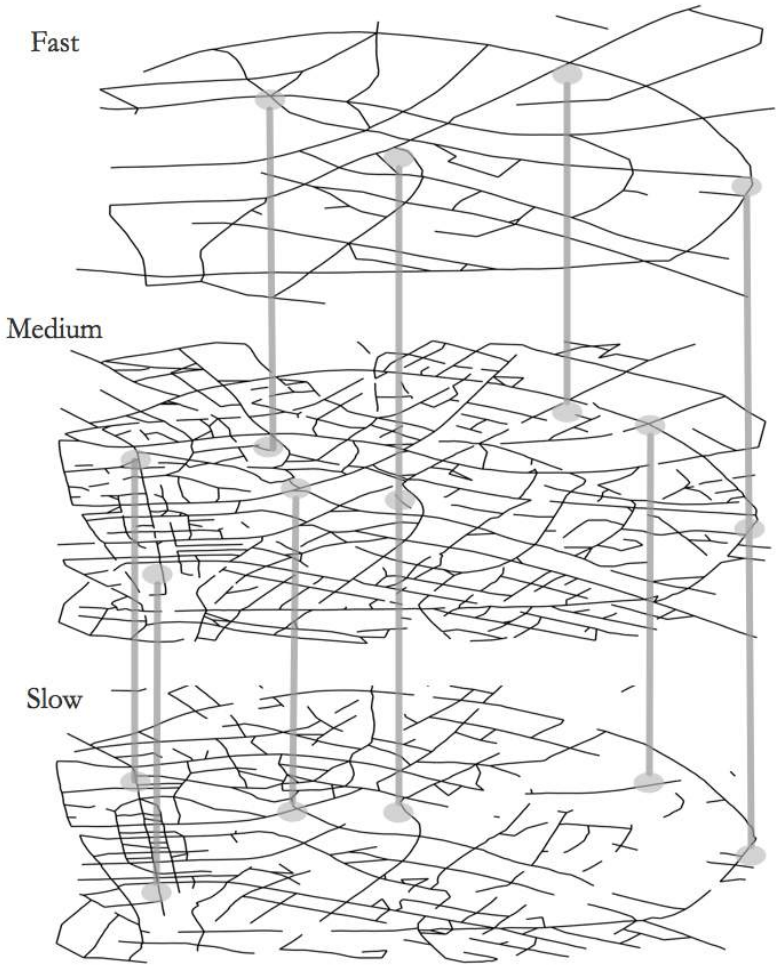


(a) Turn samples clustering in *TraceBundle* (b) Turn samples clustering in *TraceConflation*

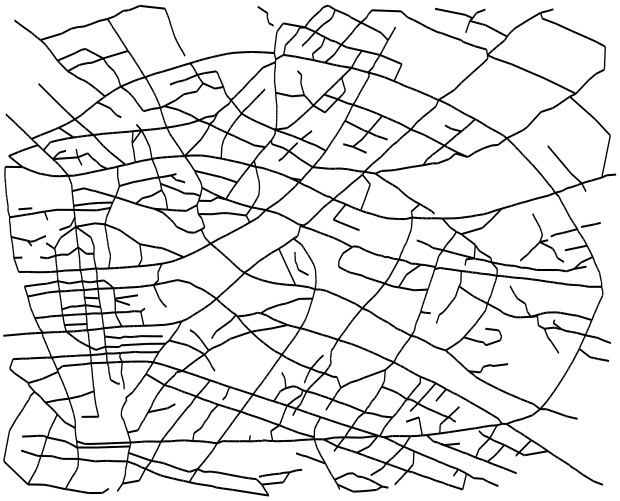
Figure 4.3: Intersections Inference.

4.1.3 Conflation of Network Layers

The final part of the process comprises the fusion of the generated network layers for the different speed categories to produce the overall road network. We build the road network incrementally starting from higher speed layers and progressing to lower speed layers. The intuition for this is that higher speed layers correspond to avenues and highways and they can be reproduced with more accuracy, since in



(a) Fast-medium-slow network - Berlin



(b) Complete constructed network - Berlin

Figure 4.4: Stitching Different Network Layers.

Algorithm 5: Finding Intersections.

Input: A set of trajectories T
Output: A set of Intersection nodes I

```
1  begin
2      /*Intersection nodes extraction based on trajectory data*/
3      P ← ∅ // Position samples set
4      P_S ← ∅ // Turn samples set
5      C_T ← ∅ // Turn clusters set
6      C_I ← ∅ // Intersection nodes set
7      α_max // angle difference threshold
8      d_max // proximity threshold
9      // Position Samples → Turn Samples
10     for (T[i] ≠ null) do
11         P ← T[i] // Positions samples of a single trajectory
12         α_d ← AngularDiff(P[i − 1], P[i], P[i + 1])
13         if (α_d ∈ Angle) then
14             α_in ← Angle(P[i − 1], P[i]) // incoming angle
15             α_out ← Angle(P[i], P[i + 1]) // outgoing angle
16             P_S.insert(P[i], α_in, α_out)
17         end
18     end
19     // Turn Samples → Turn Clusters
20     for (P_S[i] ∉ C_T do
21         // not yet considered
22         NN_P ← FindNN(P_S[i], d_max)
23         C_T ← ComputeTurnCluster(P_S[i], NN_P)
24     end
25     // Turn Clusters → Intersection Nodes
26     for (C_T[i] ∉ C_I) do
27         NN_C ← FindContained(C_T[i])
28         C_I ← ComputeIntersections(C_T[i], NN_C)
29     end
30 end
```

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 nodes 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 4.4 gives an example of this conflation process, experimented in Berlin dataset. Figure 4.4a shows the three road networks that were generated after segmenting the entire trajectory dataset of Figure 4.1. Gray lines link the various connection points between the constructed networks. The final result is shown in Figure 4.4b.

Starting with the fast and the medium network, we try to identify common nodes by spatial proximity, i.e., of their coordinates match. Experimenting with various tolerances, $10m$ was the best choice of a spatial distance threshold for two nodes in the respective networks to represent the same intersection.

The next step involves introducing new intersections onto existing links, e.g., in the fast network a link exists, but the medium network has additional intersection nodes (cf. Lines 10-11 in Algorithm 6). Using a buffer region around intersection nodes of lower layers (e.g., medium) we try to identify intersection nodes that are close to existing links. These new intersection nodes are then mapped onto the existing link and effectively split it (cf. Lines 12-21 in Algorithm 6).

Finally, new links for uncommon portions of the layered network are added, e.g., link of the medium network missing in the fast 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 (cf. Lines 24-29 in Algorithm 6).

Again, a result of applying this conflation algorithm to road network layers is shown in Figure 4.4.

Algorithm 6: Conflation of Network Layers

Input: A set of segmented trajectories H, L
Output: A conflated network graph $G = N_H, E_H$

```
1  begin
2      /*Conflation of segmented trajectories*/
3      // Networks to be conflated
4      E_H ← Edges(H)
5      N_H ← Nodes(H)
6      E_L ← Edges(L)
7      N_L ← Nodes(L)
8      N_HL // intersection pairs
9      // Node alignment
10     for N_L[i] do
11         N_HL ← (N_L[i], 1 − NN(N_L[i], N_H)) // Node insertion to higher layer
12         for (N_L[i] ∉ N_HL) do
13             E_i = On(E_H, N_L[i])
14             if E_i ≠ null then
15                 N_H.add(N_L[i])
16                 E_H.delete(E_i)
17                 E*_i ← E_i.split(N_L[i]) // produces two links
18                 E_H.delete(E_i)
19                 E_H.add(E*_i)
20             end
21         end
22     end
23     // Link insertion
24     for (N_L[i] ∉ N_H) do
25         N_H.add(N_L[i]) // remaining nodes
26     end
27     for (E_L[i] ∉ E_H) do
28         E_H.add(E_L[i]) // remaining links
29     end
30 end
```

4.2 Experimental Evaluation

In this section we present an extensive experimental study of the *TraceConflation* algorithm. To assess the effectiveness of the *TraceConflation* algorithm for inferring the road network in a layered and incremental fashion, we compare it to the *TraceBundle* algorithm presented in Chapter 3, which treats all the input data uniformly. It was shown that the *TraceBundle* algorithm largely outperforms existing algorithms for a variety of networks.

This experimental evaluation shows that the *TraceConflation* algorithm which has been presented in this chapter, performs even better. What follows is a description of the characteristics of the datasets used as well as our evaluation methodology, and then a presentation of the results of our experiments.

4.2.1 Datasets

We conduct experiments on three real-world datasets comprising vehicle tracking data from three European capital cities, namely Berlin, Vienna and Athens. In all three cases, we consider as ground-truth the corresponding road network obtained by OSM [74].

Necessary for our evaluation approach, we extract a ground-truth road network, which lines up with the tracking data. Using buffer regions around network edges, we derive the relevant part of the network by using the trajectories as a query set with the condition that they intersect the buffer regions (cf. [58]). As this process is by no means perfect, there might be redundant links which have been falsely identified. This mostly occurs in areas where the actual road network is very dense and thus neighboring road portions will be included in the extraction even though there is no tracking data coverage.

What follows is a brief description of the data (trajectories and networks) for all three cities.

4.2.1.1 Berlin

In Berlin, the actual road network consists of 6839 links (edges) and 5894 nodes. It covers an area of $6km \times 6km$. The edges have a length of $360km$. The tracking data, used for our purposes covering a great portion of this road network, comprises 15051 vehicles trajectories with a total length of $41116km$ (Figure 4.1). The total number of trajectories is 26831 with a mean length of $1.53km$. The mean sampling rate is $41.98s$, while the mean speed is $35.23km/h$.

4.2.1.2 Vienna

In Vienna, the actual road network consists of 9969 links (edges) and 8081 nodes. It covers an area of $5.5km \times 6km$. The edges have a length of 495km. The tracking data, used for our purposes comprises 7434 vehicles trajectories with a total length of $16106km$. The total number of trajectories is 12773 with a mean length of $1.26km$. The mean sampling rate is $38.59s$, while the mean speed is $33.68km/h$.

4.2.1.3 Athens

In Athens, the actual road network consists of 39699 links (edges) and 32212 nodes. It covers an area of $12km \times 14km$. The edges have a length of $2000km$. The tracking data comprises 120 vehicles trajectories with a total length of $6781km$. The total number of trajectories is 511 with a mean length of $13.27km$. The mean sampling rate is $30.14s$, while the mean speed is $20.16km/h$.

4.2.2 Evaluation Measures

A quick and easy way to get an overview of the quality of the inferred road network is by *visual inspection*, i.e., by overlaying it on the reference network and looking for similarities and differences. This way one can assess how well the constructed road network lines up with the actual road network.

To however obtain a systematic and *quantitative evaluation* of the approach, we follow the Shortest-Path Based Distance method introduced in Section 3.2. Given

the constructed and ground-truth networks, a common set of pairs of nodes (origin, destination) is selected in both. Then, the shortest paths between those pairs are computed in both networks. Performing a number of random shortest-path experiments, the geometric difference/similarity between the computed shortest paths can be used as a means to assess the quality of the constructed network.

In particular, two similarity measures are computed for each pair of shortest paths: (i) the Discrete Fréchet distance and (ii) the Average Vertical distance. The rationale for using this evaluation process is that measuring the similarity over entire paths instead of individual links allows to draw conclusions regarding more extensive portions of the road network and, especially, to take into account the connectivity of the inferred network. Essentially, the shortest paths sample the connectivity of the network. The "similar" the shortest paths in the constructed network are to the ground-truth network, the higher also the quality of the network.

4.2.3 Results

4.2.3.1 Visual Comparison

Comparing the created road networks visually is the simplest approach to assessing the map construction results. Figure 5.5 visualizes the inferred road networks for the three cities Berlin - Figures 4.5a, 4.5b, Vienna - Figures 4.5c, 4.5d and Athens - Figures 4.5e, 4.5f, showing results for the *TraceBundle* and the *TraceConflation* algorithms, respectively.

In each case, the inferred network is visualized using black lines, while the ground-truth network is shown using light gray lines. The network is derived from the OpenStreetMap dataset.

For better illustration, we have marked some areas on the maps where improvements of *TraceConflation* over *TraceBundle* can be observed. The overall observation is that the *TraceConflation* method produces results in which the core network is depicted more accurately. Especially, intersections are mapped more accurately.

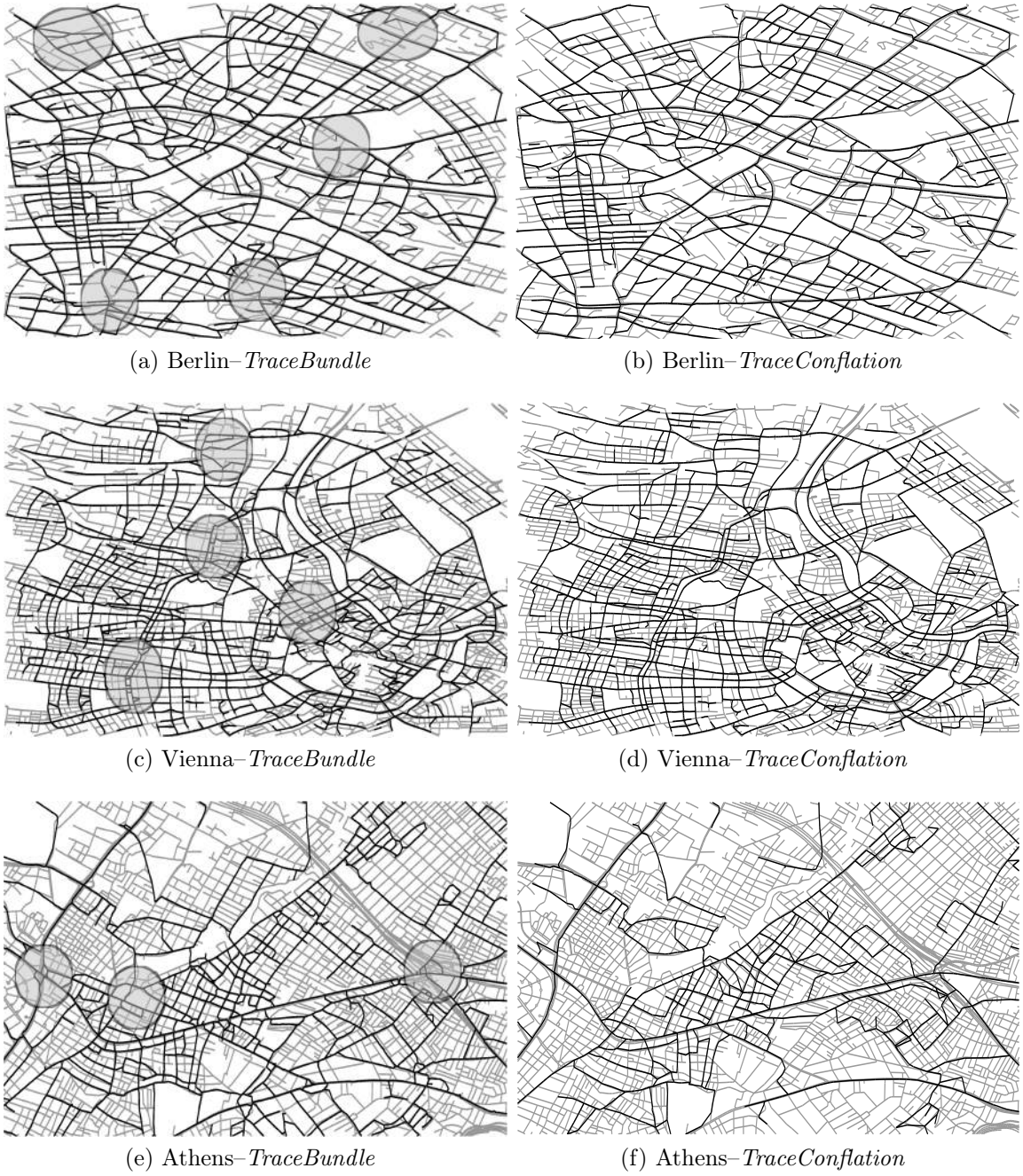
For the example of Athens, shown in Figures 4.5e and 4.5f, it can be clearly seen that due to the lack of redundancy, i.e., availability of tracking data, a smaller network is extracted by the *TraceConflation* method. However, what has been constructed is of increased accuracy.

4.2.3.2 Quantitative Evaluation

As a means to objectively and quantitatively evaluate and compare the constructed networks, we computed for each city case 500 shortest-paths problems. Origin and destination nodes are uniformly distributed over the networks.

The intuition is that if the constructed network closely matches the actual, ground-truth network, then the shortest paths found for each network should closely resemble each other. The more dissimilar the networks, the more the respective shortest paths will differ.

Dissimilarity is captured by computing two respective distance measures for the paths. Figure 4.6a shows the Discrete Fréchet distance between paths computed by the *TraceBundle* network (in light grey) and the *TraceConflation* method (in black), respectively.



(a) Berlin–*TraceBundle* (b) Berlin–*TraceConflation*

(c) Vienna–*TraceBundle* (d) Vienna–*TraceConflation*

(e) Athens–*TraceBundle* (f) Athens–*TraceConflation*

Figure 4.5: Inferred Road Networks - The *TraceConflation* Algorithm.

The figures plot the distance between the paths for each of the 500 paths in each case. The paths are ordered by decreasing distance of the *TraceConflation* result with respect to the ground-truth road network, i.e., the path that exhibits the greatest distance ($500m$ in Figure 4.6a) is plotted first. Figure 4.6b plots the second distance measure, the Average Vertical distance between the two paths and respective networks for the case of Berlin. Figures 4.6c and 4.6d depict the results for the city of Vienna, while Figures 4.6e and 4.6f show the evaluation results for Athens.

The evaluation for Berlin shows a significant improvement with respect to path similarity and, thus, constructed network. 93.8% of the paths showed increased

similarity.

To condense the results into single numbers, the distance plots are summarized in Table 4.1. For Berlin the shortest paths using Discrete Fréchet distance, for the case of the *TraceBundle* (old) method exhibit a minimum distance of $18m$ (last point in plot), a maximum distance of $499m$ (first point in plot), and when considering all 500 paths, resulting in an average distance of $253m$, with a standard deviation of $142m$. The numbers for the *TraceConflation* case are improved, i.e., a minimum distance of $14m$, a maximum distance of $477m$, an average distance of $237m$, with a standard deviation of $132m$.

The results are similar when considering the Vertical distance measure. The cases of Vienna and Athens shows similar results, with results for the latter showing a bigger improvement.

In terms of number of paths that were actually improved, for Vienna, experimental results show significant improvements for 96.4% of the paths. However, in Athens 8.4% of the results experience connectivity problems due to the limited data coverage and no shortest path being found to connect two nodes to produce a shortest-path. Measurements are missing for those paths in Figures 4.6e and 4.6f. This case was discussed before, in that the *TraceConflation* method is more demanding with respect to the trajectories that redundantly describe the network.

Segmenting the trajectory set into three partitions reduces the data that is available to generate each network layer. This might lead to a situation in which network links that were identified using the *TraceBundle* approach are not being found with the *TraceConflation* method.

		Discrete Fréchet distance				**Average Vertical distance**			
		min	**max**	**avg**	**stddev**	**min**	**max**	**avg**	**stddev**
Berlin	*TraceBundle*	18	499	253	142	8	360	126	101
	TraceConflation	14	477	237	132	6	346	124	87
Vienna	*TraceBundle*	15	416	208	117	4	212	108	66
	TraceConflation	12	410	203	115	2	198	97	63
Athens	*TraceBundle*	64	612	235	102	8	245	98	49
	TraceConflation	58	404	214	94	6	176	95	43

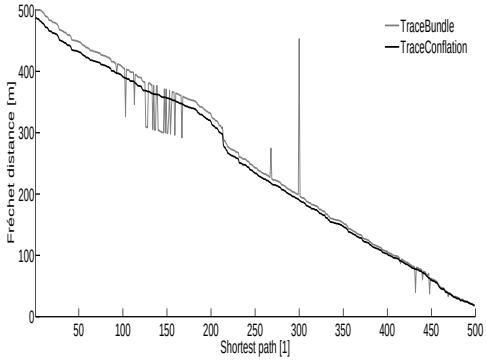
Table 4.1: Shortest-Path Comparison Summary

4.3 Summary

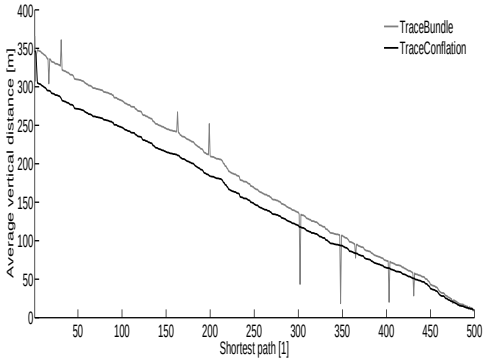
In this chapter, we have addressed the problem of automatic road network inference from sparse and noisy tracking data and we have proposed a novel approach to the map inference problem, i.e. the *TraceConflation* algorithm, which builds the map of a road network in an incremental and layered fashion.

In a nutshell, the *TraceConflation* algorithm is based on segmenting the trajectory datasets based on speed profiles, constructing separate map layers and conflating these results into a single road network. It also introduces an improved version of the *TraceBundle* algorithm to be used as the basic means for computing the separate road network layers.

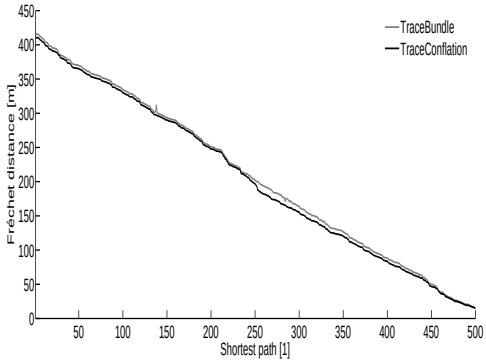
Experimentation have shown that the *TraceConflation* algorithm, by segmenting vehicle trajectories based on speed profiles produces navigable road networks that when compared to the *TraceBundle* algorithm, are of improved accuracy.



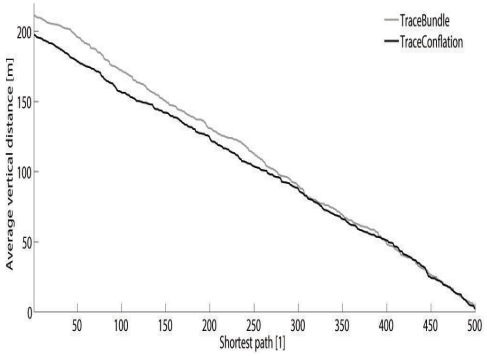
(a) Evaluation (Fréchet distance) - Berlin



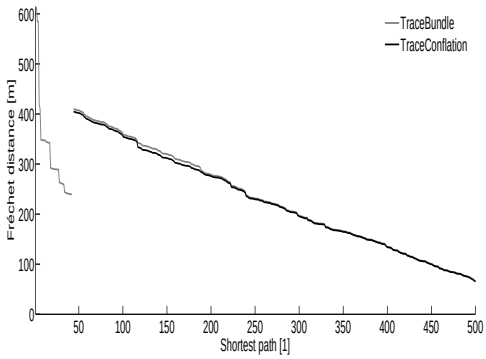
(b) Evaluation (Average vertical distance) - Berlin



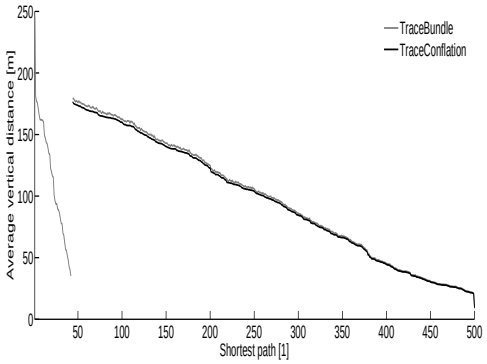
(c) Evaluation (Fréchet distance) - Vienna



(d) Evaluation (Average vertical distance) - Vienna



(e) Evaluation (Fréchet distance) - Athens



(f) Evaluation (Average vertical distance) - Athens

Figure 4.6: Road Networks Comparison.

Assessing the two respective inferred road networks using sets of shortest-path queries, found that *∼95% of paths computed exhibit significant improvement* similarity when compared to the ground-truth network results.

Visual inspection shows that the *TraceConflation* road networks better capture the actual road network if sufficient tracking data are available with redundant network coverage. Performing an experimental evaluation using three large-scale trajectory datasets, the *TraceConflation* algorithm produces road networks of improved accuracy. The resulting "maps" come very close to accurately capturing the ground-truth road network geometry, both in terms of topology and spatial accuracy.

Chapter 5

Inference of Transportation Networks from Multimodal Crowdsourced Data

An ever increasing amount of geospatial data generated by mobile devices and social media applications becomes available and presents us with applications and also research challenges. In previous Chapters 3 and 4, we have addressed the problem of automatic road network inference using redundant and sparse tracking data. This data has more geometric characteristics as the moving objects travel in a deterministic environment which is a road network. In this chapter, we go a little further the notion of networks by means of generalized transportation networks, which capture both the geometric and the semantics of people transportation in an urban area.

The scope of this approach is to discover persistent and meaningful knowledge from user-generated location-based "stories" as reported by Twitter data. We propose a novel methodology that converts geocoded tweets into a mixed geosemantic Network-of-Interest (NOI). It does so by introducing a novel network construction algorithm on segmented input data based on discovered mobility types. The generated network layers are then combined into a single network. This segmentation addresses also the challenges imposed by noisy, low-sampling rate "social media" trajectories. An experimental evaluation assesses the quality of the algorithms by constructing networks for London and New York. The results show that this method is robust and provides accurate and interesting results that allow us to discover transportation hubs and critical transportation infrastructure.

The remainder of this chapter is organized as follows. Sections 5.1 and 5.2 present our algorithms for trajectory segmentation and re-association to build the Network-of-Interest in a layered fashion. In Section 5.3, we evaluate the quality of the Network-of-Interest construction method. Finally, Section 5.4 presents some conclusions regarding this chapter.

Our results in this chapter have been published in [86].

5.1 Network-of-Interest Layer Construction

In this section, we present how we extract a Network-of-Interest that captures interesting information about user movement behaviors based on social media tracking data. User check-in data are tuples of the form $U = \langle u, x, y, t \rangle$, denoting that the

user u was at location (x, y) at time t. These data are organized into trajectories, which represent the sequence of locations a user has visited. Typically, multiple trajectories are produced for each user by splitting the whole sequence of check-ins, e.g., on a daily basis. Hence, each resulting trajectory is an ordered list of spatiotemporal points $T = \{p_0, \ldots, p_n\}$ with $p_i = \langle x_i, y_i, t_i \rangle$ and $x_i, y_i \in R, t_i \in R^+$ for $i = 0, 1, \ldots, n$ and $t_0 < t_1 < t_2 < \ldots < t_n$.

The goal is to construct a Network-of-Interest that reveals the *movement behavior* of users. This Network-of-Interest is a directed graph $G = (V, E)$, where the vertices V indicate important locations and the edges E important links between them according to observed user movements.

In particular, we are interested in two aspects of the Network-of-Interest. A *geometric NOI aspect* provides a representation of how users actually move across various locations, thus preserving the actual geometry of the movement, while a *semantic NOI aspect* represents the qualitative aspect of the network by identifying significant locations and links between them.

In our approach, we treat these two aspects as different layers of the same Network-of-Interest. In the following, we describe the steps for constructing these layers and fusing them to produce the final Network-of-Interest.

5.1.1 Segmentation of Trajectories

Behavioral trajectories, as in our case derived from geocoded tweets, contain data to construct both the geometric and the semantic layer of a Network-of-Interest. Conceptually, users tweet when they stroll around in the city as well as when they commute in the morning. While all these tweets will result in behavioral trajectories, *some of them depict actual movement paths*, while others simply are tweets sent throughout the day. In what follows, we try to separate our input data into two subsets and to extract the trajectories corresponding to the respective layer.

A main challenge when inferring a movement network from check-in data is that this data are very heterogeneous in terms of their sampling rate, i.e., often being very sparse. However, even the sparse subsets of the data are helpful in identifying significant locations, whereas the denser subsets can be used to capture more fine grained patterns of user movement.

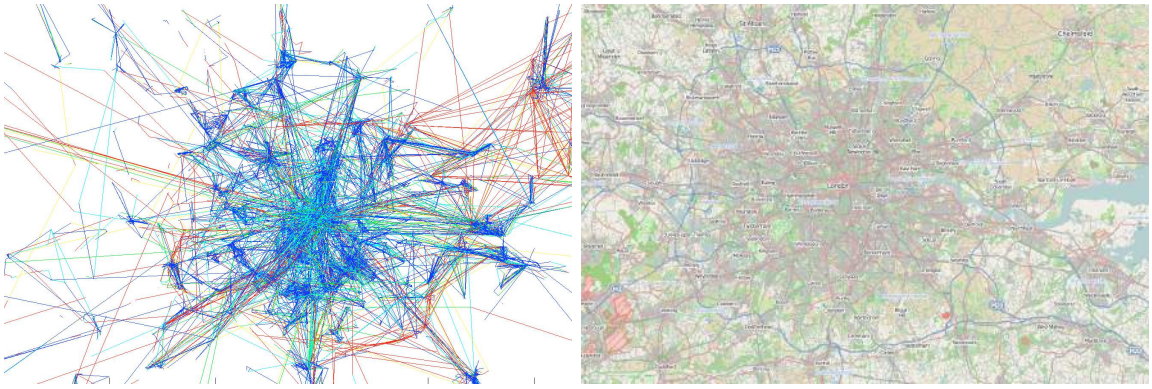
For this purpose, we analyze the trajectories and group them into subsets with different temporal characteristics. In our approach, we treat these two aspects by applying a (i) *mean speed* threshold to capture the user movement under an urban transportation mode and by applying (ii) a *sampling rate* threshold to identify "abstract" and "concrete" movement. This allows us to treat each subset separately later on in the network construction phase.

The "abstract" type of movement corresponds to the *semantic Network-of-Interest aspect* and the "concrete" corresponds to the *geometric Network-of-Interest aspect*. Users with frequent check-ins, i.e., a high sampling rate, provide us with the means to derive a geometric NOI layer, while low sampling rates only allow us to reason about abstract movement, i.e., derive a semantic NOI layer.

Notice that typically the same individual, within one daily trajectory may have recorded their data using different sampling rates. In this case, the trajectory needs to be segmented according to the frequency of user position samples. A simple process for achieving this separation is the following.

First, a duration and a speed (length divided by duration) is recorded for each segment of a trajectory. Each segment is assigned a corresponding duration type of movement. Focusing on urban transportation, we use a mean speed to filter out trajectories and then the duration between samples to determine "abstract" and "concrete" movement.

Figure 5.1 shows the trajectories classified to different sampling rates using the example of geocoded tweets for London. Using a heatmap colouring schema, concrete and abstract movements are shown in blue and red, respectively. The bounding box of the selected area is the following: ([51.18N, 0.85W],[51.80N, 0.86E]).



(a) Twitter trajectories("slow": blue, "fast": red) (b) Respective OSM network

Figure 5.1: Twitter Trajectories and OSM Network - London.

The process is outlined in Algorithm 7. For each line segment L_j of each trajectory T, we compute a duration and a mean speed value (Algorithm 7, Lines 6-7), and the segment is then assigned to the corresponding segmented set of trajectories T_G, T_S according to the min and max time interval (Lines 9-13). The algorithm produces segmented sets of trajectories (Lines 10 and 13) based on the corresponding time interval attributes.

Algorithm 7: Segmentation of Trajectories.

Input: A set of trajectories T
Output: Two sets of segmented trajectories T_G, T_S

```
1  begin
2      /*Trajectories segmentation according to time intervals*/
3      V_max // maximum mean speed
4      foreach (T_i ∈ T) do
5          foreach (L_j ∈ T_i) do
6              t̄(L_j) ← δt(P[i−1], P[i]) // Time interval
7              v̄(L_j) ← δx(P[i−1],P[i]) / δt(P[i−1],P[i]) // Mean speed
8              if v̄(L_j) ≤ V_max then
9                  if t̄(L_j) ≤ T_min then
10                     T_G ← L_j
11                 end
12                 else if t̄(L_j) ≥ T_min and t̄(L_j) ≤ T_max then
13                     T_S ← L_j
14                 end
15             end
16         end
17     end
18 end
```

5.1.2 Geometric Layer Construction

To construct the geometric NOI layer we use frequently sampled trajectories. The sampling rate threshold was established through experimentation. In the examples of Section 5.3, the sampling rate threshold was set to $5min$. I.e., for the construction of the geometric layer the duration in between position samples of trajectory dataset is less than $5min$ (cf. Table 5.1), approximately covering 57% of the original tweets collection.

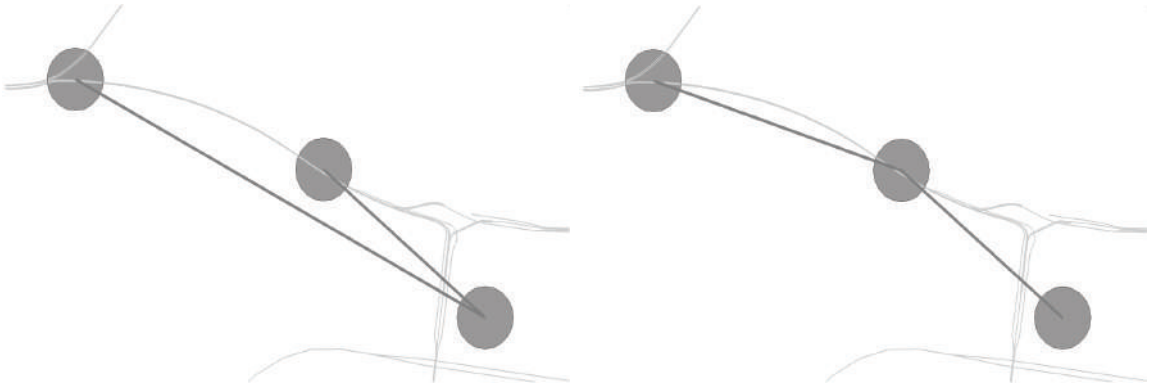
The geometric NOI layer construction approach follows a modified map construction approach of the *TraceBundle* and *TraceConflation* algorithms by (i) initially clustering position samples to derive network nodes, (ii) linking nodes by using the trajectory data and (iii) refining the link geometry.

To derive network nodes we employ the DBSCAN clustering algorithm [38] using a distance threshold and a minimum number of samples threshold parameter. We revisit the segmented trajectories to identify how the network nodes are connected by creating links. The links represent clustered trajectories as two nodes can be connected by different trajectories.

For each link (i) a *weight* is derived representing the number of the trajectories comprising the link and also (ii) a *length* representing the Euclidean distance between the nodes that constitute the link.

In addition to this, we apply a reduction step to simplify the constructed network. The intuition is that due to varying sampling rates, links between nodes might exhibit redundancy. This reduction step eliminates redundant links by substituting longer links with links of more detailed geometries. We reconstruct links of longer duration by using links of shorter duration if their geometries are similar.

We achieve this by using the degree of constructed nodes. Starting with nodes of a higher degree of incoming links, i.e., significant nodes, for such a node, we sort all incident links based on descending duration order. We then reconstruct those, which temporally and spatially cover other links that can be reached in less time. Figure 5.2a gives an example by showing in dark gray links before reduction, and in light gray a portion of the underlying OSM transportation network. Figure 5.2b shows then in dark gray the resulting links after applying the reduction step. Part of the larger geometry has been substituted with a more detailed geometry.



(a) before reduction (b) after reduction

Figure 5.2: Network Reduction Example.

5.1.3 Semantic Layer Construction

To construct the Semantic NOI layer, we rely on trajectories exhibiting low sampling rates (using approximately 19% of the original tweets collection), i.e., potentially cover large distances in between position samples making it difficult to reconstruct the actual movement (cf. Table 5.1).

By initially applying the DBSCAN clustering algorithm (see Table 5.1 for parameter details), we extract a set of nodes that correspond to the *hubs* of the semantic layer. Performing a linear scan of the trajectories reveals the respective portions that connect the sets of nodes. For each link sample (i) a *weight* is derived representing the number of the trajectories comprising a link. At this step, we do not apply any reduction method as the geometries of the semantic layer are less accurate. Overall, this layer allows us to extract a network with less spatial accuracy but of greater semantic value.

5.2 Network-of-Interest Layer Fusion

The final part of the Network-of-Interest construction process consists of (i) the extraction of hubs, i.e., significant locations that user frequently visits, and (ii) the fusion of the layers, i.e., the geometric and the semantic layer to produce the integrated network.

5.2.1 Network Hubs

Hubs are POIs that users frequently depart from and arrive at. In particular, specific indicators for hubs are (i) number of constituting position samples, (ii) stemming from many different users, (iii) over extended periods of time.

The Network Hubs Inference algorithm takes as input the *entire trajectory dataset* used in geometric and semantic layer construction (Algorithm 8, Line 10) and determines the k-NNs of each position sample (Line 13), which are subsequently filtered according to the number of users and the period of time covered (Lines 13-16). On these filtered position samples, we apply the DBSCAN clustering algorithm using a distance threshold and a minimum number of samples (Line 17). The centroids of the resulting clusters are the candidate hubs (Line 18).

Algorithm 8: Hub Inference.

Input: A set of segmented trajectories T_G, T_S
Output: Network Hubs

1 **begin**
2 /*Clustering position samples of segmented trajectories to compute network hubs*/
3 $H^* \leftarrow \emptyset$ // Candidate Hubs
4 $H \leftarrow \emptyset$ // Hubs
5 d_{max} // proximity threshold
6 u_{min} // min. number of users
7 h_{min} // min. number of time periods
8 $deg_{in}, deg_{out}, deg_{min}, \epsilon$
9 // position samples from combined trajectories
10 $P \leftarrow UNION(T_G, T_S)$
11 // Samples $\rightarrow$ Hubs
12 **foreach** $(P[i])$ **do**
13 $\nu_i \leftarrow FindNN(P[i], d_{max})$
14 $u_p \leftarrow CountUsers(\nu_i)$
15 $h_p \leftarrow CountHours(\nu_i)$
16 **if** $(u_p \geq u_{min})$ *and* $(h_p \geq h_{min})$ **then**
17 $C \leftarrow DBSCAN(\nu_i, d_{max})$ // Clusters
18 $H^* \leftarrow Centroid(C)$ // Hub candidates
19 **end**
20 **end**
21 **foreach** $H^*[i]$ **do**
22 $deg_{in} \leftarrow GetInDeg(H^*[i])$
23 $deg_{out} \leftarrow GetOutDeg(H^*[i])$
24 **if** $deg_{in} \geq deg_{min}$ *and* $deg_{out} \geq deg_{min}$ *and* $\left|\frac{deg_{in}}{deg_{out}} - 1\right| \leq \epsilon$ **then**
25 $H \leftarrow H^*[i]$
26 **end**
27 **end**
28 **end**

A final filtering step is applied as follows. For each candidate hub, we also record two properties. A *weight* for the hub is derived as the total number of nodes the hub was derived from, i.e., the size of the corresponding cluster. In addition, we record the *degree* of each hub, i.e., the number of incoming and outgoing edges of the cluster. A candidate hub is included in the output if both the following two conditions hold: (a) both the in-degree and out-degree are above a specified threshold and (b) the in-degree and out-degree do not differ significantly (threshold determined by experimentation). These conditions are used to ensure that the identified hubs correspond to places where a sufficiently large number of users frequently depart from and arrive at (Lines 22-24).

5.2.2 Layer Fusion

The final part of the process comprises the fusion of the geometric and semantic NOI layers. We construct the NOI by starting with the semantic layer and merging the geometric layer onto it. The intuition for this is that the semantic layer corresponds to a geometrically abstract but semantically richer user movement that contains relevant transportation hubs. The geometric layer corresponds to a less semantic but more accurate depiction of movement, i.e., fills in the gaps of the semantic layer. The fusion of these layers should result in a comprehensive movement network.

The fusion task involves (i) finding hub correspondences among the different network layers and (ii) introducing new links to the semantic layer for the uncommon portions of the NOI.

Algorithm 9: NOI Fusion.

Input: Networks to be conflated S, G
Output: Network of Interest

```
1  begin
2      /*Network layers fusion to extract the final map*/
3      // edges and nodes of Semantic and Geometric layers
4      E_S ← Edges(S), N_S ← Nodes(S)
5      E_G ← Edges(G), N_G ← Nodes(G)
6      H // Hubs
7      H_G // hubs ∩ geometric nodes
8      H_S // hubs ∩ semantic nodes
9      H_O // H − H_G − H_S
10     // Node alignment
11     foreach H[i] do
12         // finding Nearest Neighbors
13         H_G ← (H[i], NN(H[i], N_G))
14         H_S ← (H[i], NN(H[i], N_S))
15     end
16     // Node alignment
17     foreach H_G[i] do
18         H_O ← (H_G[i], 1 − NN(H_G[i], H_S))
19         // Node insertion to semantic layer
20         foreach (H_G[i] ∉ H_O) do
21             E_i = On(E_S, H_G[i])
22             if E_i ≠ null then
23                 H_S.add(H_G[i])
24                 E_S.delete(E_i)
25             end
26         end
27         // Link insertion
28         foreach (H_G[i] ∉ H_S) do
29             H_S.add(H_G[i]) // remaining nodes
30             foreach (E_G[i] ∉ E_S) do
31                 E_S.add(E_G[i]) // remaining links
32             end
33         end
34     end
35 end
```

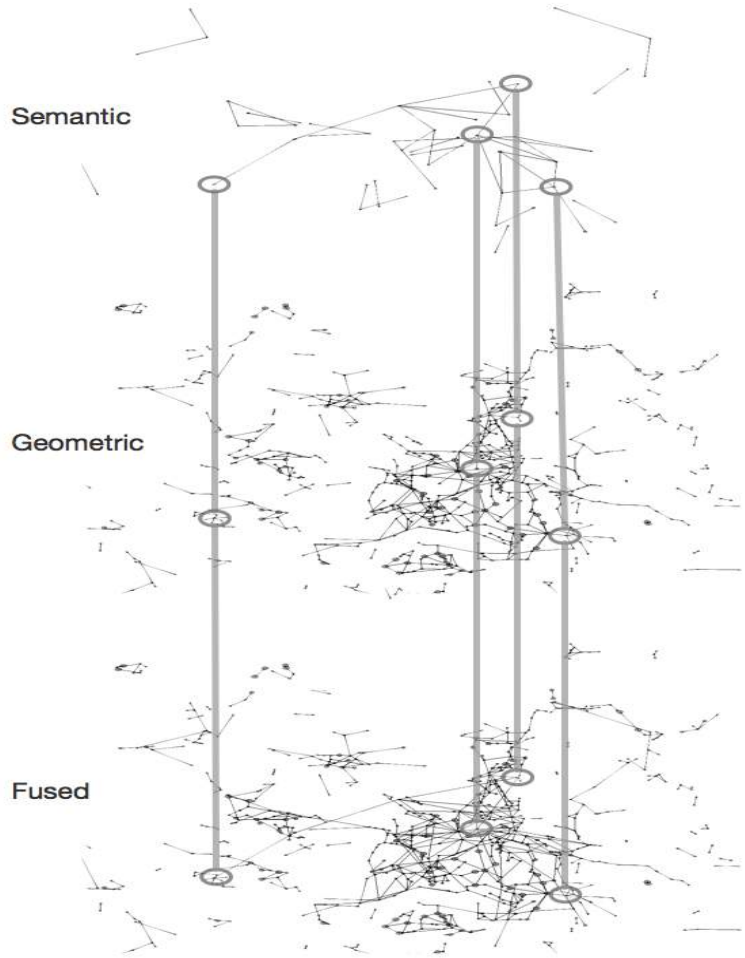
Using, both layers and the hubs, we try to identify common nodes by spatial proximity (Algorithm 9, Lines 12-14). Any node from the geometric layer that has not been introduced yet since it is not connected to the semantic layer will be added (Lines 23-24). The next step involves introducing new links for uncommon portions of the layered network. Here links of the geometric layer are introduced by adding them to the semantic layer (Lines 29-31). Typically this accounts for the cases of adding complete (local) network portions.

A result of applying this conflation algorithm to network layers is shown in Figure 5.3. Indicated are the circled hub correspondences between the semantic, the geometric layer, and the resulting fused Network-of-Interest.

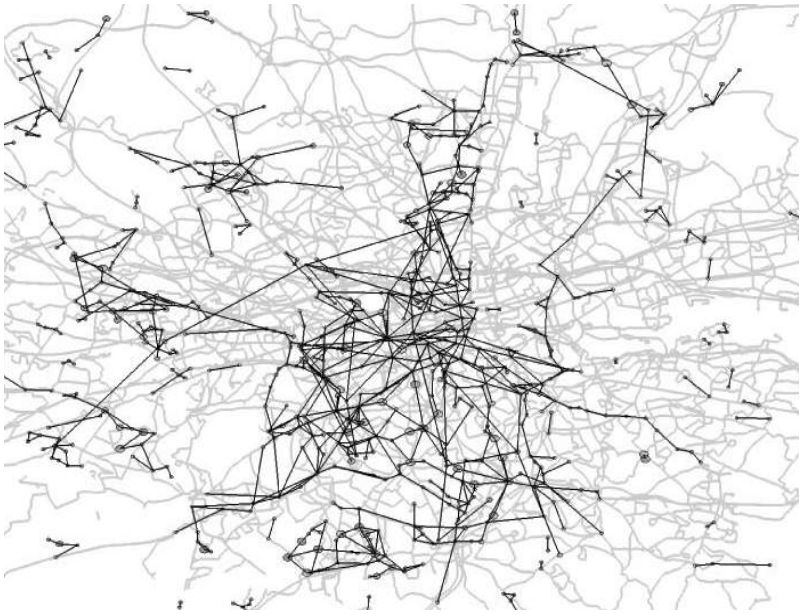
5.3 Experimental Evaluation

An assessment of the quality of a Network-of-Interest is a challenging task as there is no ground-truth data. In the case of map-construction algorithms, an existing road network can be used. However, a Network-of-Interest represents a geosemantic construction containing aspects of both regular transportation networks (roads, public transport, etc), but also the overall movement sentiment of users in a city.

For the following evaluation, we use a combination of existing POI datasets and (public) transportation networks to assess the constructed NOIs. Before giv-



(a) Geometric and Semantic networks - London



(b) Network-of-Interest - London

Figure 5.3: Stitching Network-of-Interest Layers.

Algorithm	Value
Segmentation of Trajectories	
Mean Speed	$10km/h$
Time Interval	5, $60min$
Geometric Network-of-Interest	
Distance Threshold	$100m$
Minimum Number of Samples	2
Semantic Network-of-Interest	
Distance Threshold	$300m$
Minimum Number of Samples	2
Extraction of Hubs	
Minimum Number of Samples	10
Minimum Number of Users	2
Minimum Number of Time Periods	10
Distance Threshold	$300m$
Layer Fusion	
Distance Threshold	$50m$

Table 5.1: Parameter Summary.

ing details of the experimental results and constructed NOIs, we first describe the characteristics of the datasets used and our overall evaluation methodology.

5.3.1 Datasets

We conduct experiments on two real-world datasets comprising geocoded tweets retrieved for London and New York city over a period of 60 days using the Twitter Public Stream API. Data from London cover the period of December 2012 to January 2013. The New York as collected from November 2013 to December 2013. To focus on trajectories of active users, we kept only the trajectories of the top 200 most active users with respect to geotweets for each city. Moreover, we only consider trajectories that consist of at least 5 geotweets.

Figure 5.1 visualizes the movements of 200 Twitter users during the course of a single day in London. Notice that some very prominent areas, such as highways, can be distinguished visually even before any processing of the data takes place. Through experimentation, we establish the parameters for the various steps of the algorithm as summarized in Table 5.1. To compare the generated network, we consider as ground-truth data the corresponding public transportation network obtained from OSM [74].

What follows is a brief description of the trajectories collected from the geocoded tweets, as well as the networks obtained from OSM.

In London, the actual public transportation network consists of 27,021 links (edges) and 47,575 nodes (vertices) and has a length of 21,287km. It covers an area of $420km \times 118km$ including the metropolitan area of London. The geocoded tweets cover a great portion of this network, specifically an area of $365km \times 104km$, and have a total combined length of 256,400km (Figure 5.1). The dataset consists of 463 trajectories with a mean length of $7.4km$. The mean sampling rate, i.e., rate at which a user geotweets, is $12min$, while the mean speed is $37km/h$.

For New York the actual public transportation network consists of 84,367 links and 75,070 nodes and has a length of 9,846km. It covers an area of $105km \times 85km$.

	Nearest Neighbor Statistics			Reverse Geocoding Statistics		
	Found	**Total**	**Ratio %**	**Found**	**Total**	**Ratio %**
London	1389	1562	89	964	1562	62
New York	1423	1649	86	873	1649	53

Table 5.2: Evaluation Summary.

The geocoded tweets consist of 37,962 trajectories, with a mean length of $2.9km$ and total length of $214{,}090km$, covering an area of $92km \times 74km$ largely overlapping with the public transportation network. The mean sampling rate is $8min$, while the mean speed is $22km/h$.

5.3.2 Evaluation Measures

For a more systematic and quantitative assessment of Networks-of-Interest we devise two means, (i) comparing the constructed Network-of-Interest to the geometry of a respective transportation network and (ii) comparing the nodes of our Network-of-Interest with a POI dataset to discover semantics in terms of their type. This approach allows us to assess the similarity with respect to the ground-truth network and to draw conclusions not only with respect to the spatial accuracy of the result, but also the semantics of the nodes.

To *compare networks* we select all the nodes of the constructed network and identify corresponding nodes in the ground-truth network by means of nearest-neighbor queries. Using the OSM public transport data, we select for every hub of the Network-of-Interest the nearest node in the OSM data. If the inferred nodes are close to the actual transportation network nodes, then the constructed Network-of-Interest closely relates to the transportation network.

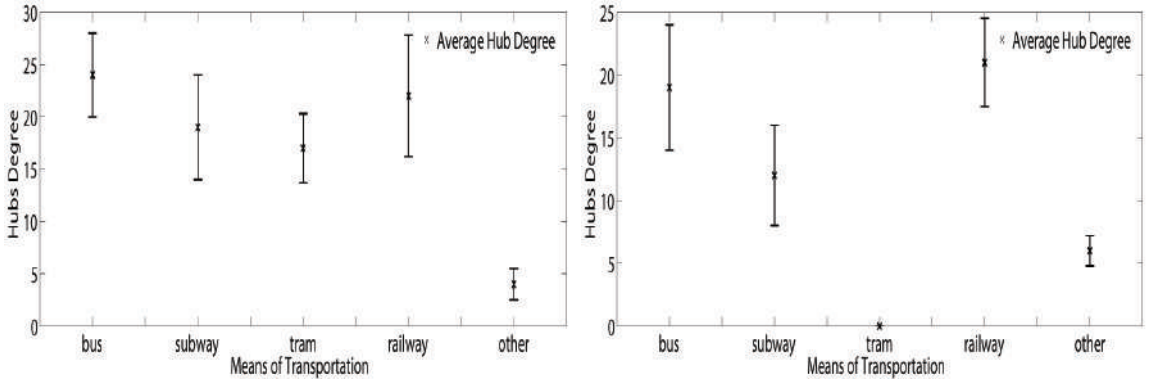
To discover the *type of transportation* a hub represents, e.g., bus, metro, tram and railway, we again use OSM data. We apply reverse geocoding (identify POIs based on coordinates) to relate OSM POIs to Network-of-Interest locations. This then allows us to identify public transportation nodes in our generated Network-of-Interest.

The results are summarized in Figure 5.4, which shows the degree of a node, i.e., the number of incoming and outgoing links. In this case, we use the degree as an indicator for the importance of the node and the fact that high-degree nodes were identified as transportation nodes allows us to reason about the type of network we constructed. Identified transportation nodes (i.e. bus, metro, etc) have higher degrees (> 20) when compared to *other* nodes with lower degree (< 5).

In this experimentation, (i) nearest-neighbor queries evaluate the spatial accuracy of the NOI, while (ii) the reverse geocoding assesses the semantics of the hubs. The higher the number of correctly constructed nodes, the higher also the quality of the network.

As shown in Table 5.2, transportation nodes are inferred with high accuracy. Indeed, 89% of the extracted hubs in London and 86% in New York are identified as transportation nodes in the OSM ground-truth network. In the case of the reverse geocoding test, the ratios are a bit lower due to the fact that the reverse geocoding service returns only POIs that are located exactly or very closely to specific coordinates.

An overall sentiment of our experimentation could be that the network construc-



(a) London (b) New York

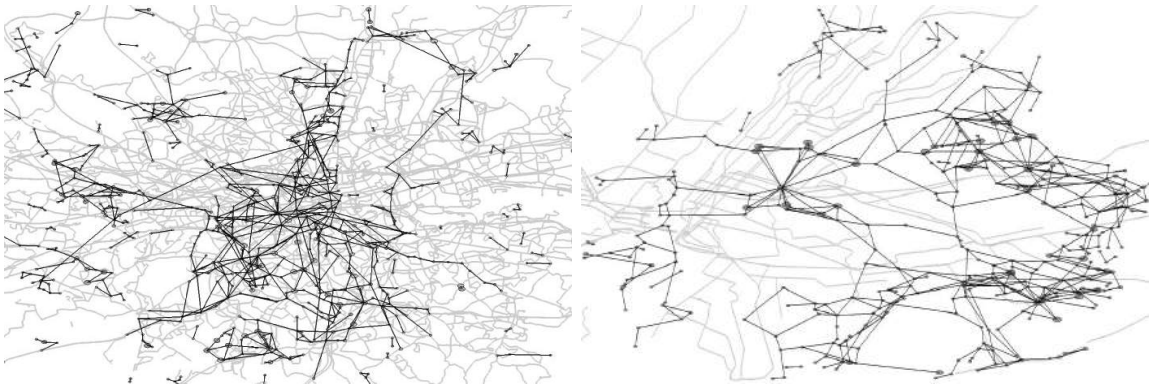
Figure 5.4: Hubs Statistics.

tion process results in a Network-of-Interest that *captures certain aspects of a public transportation network*. A core problem in such experimentation is that using social media as a tracking data source to construct a network has the inherent challenge that no actual ground-truth data is available to assess the quality of the result. Using in our case a public transportation network allows us to show some similarities, however, the constructed Network-of-Interest could not be completely mapped (explained) by it as it represents a more complex network whose characteristics cannot be captured by a single existing network dataset. These concerns are also issues we want to address in future work.

5.3.3 Results

An overview of the quality of the inferred Network-of-Interest can be also obtained by *visual inspection*, i.e., by comparing it to the ground-truth public transportation network and looking for similarities and differences.

Figure 5.5 visualizes the NOIs of the cities of London (Figure 5.5a) and New York (Figure 5.5b). In each case, the constructed network is visualized using black lines, while the ground-truth network is shown using light gray lines. As evident, especially for the case of New York, the constructed Network-of-Interest lines up with the transportation network and identifies major hubs.



(a) London (b) New York

Figure 5.5: Networks of Interest.

5.4 Summary

Social media applications and their data have been used in a wide range of data mining applications. However, to the best of our knowledge this approach is the first to construct a geosemantic Network-of-Interest using social media as a tracking data source.

The Network-of-Interest construction algorithm is based on segmenting geocoded tweets and constructing two separate network layers. A geometric and a semantic layer of a NOI are derived and using network hubs, these layers are then fused to generate a Network-of-Interest.

Performing an experimental evaluation using two large-scale datasets, the algorithm produces Networks-of-Interest of considerable accuracy, which identify important transportation hubs and capture portions of the respective public transport networks.

Chapter 6

Going Beyond Typical Map Inference Tasks

In this chapter, we present some preliminary results which are produced by applying a modified version of the *TraceBundle* algorithm on eye tracking datasets to extract polylines from observations of cartographic lines. In the following, we present the eye tracking process and the application domains of the latest years.

Eye tracking is the process of measuring either the point of gaze (where one is looking) or the motion of an eye relative to the head. An eye tracker is a device for measuring eye positions and eye movement. Eye trackers are used in research on the visual system, in psychology, in cognitive linguistics and in product design.

A great deal of research has gone into studies of the mechanisms and dynamics of eye rotation, but the goal of eye tracking is most often to estimate gaze direction. Users may be interested in features that the eye draws from an image, for example. It is important to realize that the eye tracker does not provide absolute gaze direction, but rather can only measure changes in gaze direction. In order to know precisely what a subject is looking at, some calibration procedure is required in which the subject looks at a point or series of points, while the eye tracker records the value that corresponds to each gaze position.

Each method of eye tracking has advantages and disadvantages, and the choice of an eye tracking system depends on considerations of cost and application. There are offline methods and online procedures like *Attention Tracking*. There is a trade-off between cost and sensitivity, with the most sensitive systems costing many tens of thousands of dollars and requiring considerable expertise to operate properly. Advances in computer and video technology have led to the development of relatively low cost systems that are useful for many applications and fairly easy to use. Interpretation of the results still requires some level of expertise, however, because a misaligned or poorly calibrated system can produce wildly erroneous data.

A wide variety of disciplines use eye tracking techniques, including cognitive science, psychology (notably psycholinguistics, the visual world paradigm), human-computer interaction (HCI), marketing research and medical research (neurological diagnosis). Specific applications include the tracking eye movement in language reading, music reading, human activity recognition, the perception of advertising, i.e. commercial eye tracking which includes web usability, marketing, automotive, etc., and the playing of sport.

In recent years, the increased sophistication and accessibility of eye tracking tech-

nologies have generated a great deal of interest in the commercial sector. Applications include web usability, advertising, sponsorship, package design and automotive engineering. In general, commercial eye tracking studies function by presenting a target stimulus to a sample of consumers while an eye tracker is used to record the activity of the eye. Examples of target stimuli may include websites, television programs, sporting events, films, commercials, magazines, newspapers, packages, shelf displays, consumer systems (ATMs, checkout systems, kiosks), and software. The resulting data can be statistically analyzed and graphically rendered to provide evidence of specific visual patterns. By examining fixations, saccades, pupil dilation, blinks and a variety of other behaviors researchers can determine a great deal about the effectiveness of a given medium or product.

The remainder of this chapter presents a description of the methodology that we apply on eye tracking data in order to infer the polylines from observations of cartographic lines. Section 6.1 presents use cases of eye tracking data, related approaches and motivation paradigms. In Section 6.2, we present a modified *TraceBundle* approach with respect to eye tracking data and inference of hubs for the identification of spatial fixation regions of people's interest. What follows is the hubs connection, the inference of a single polylines geometry and the experimental results. Finally, Section 6.3 presents our conclusions and directions for future work.

Our results in this chapter have been published in [57].

6.1 Visualization of Eye Tracking Data from Observations of Cartographic Lines

Several visualization methods for eye tracking data exist to help researchers from many disciplines depict data collected in eye tracking experiments. Focusing on eye tracking data from observations of cartographic lines, in this section we discuss early findings on a new visualization that uses inferred polylines instead of more traditional techniques such as heat maps to visualize eye tracking data. This visualization depicts the average line that is actually seen by subjects, which can be useful in the study of cartographic concepts such as the assessment of the effects of alternative cartographic lines presentations in maps, of distractions, abstraction levels and more.

Eye tracking is a widely used methodology in many scientific fields, as it reveals important findings about the human cognitive processes during the observation of a visual stimulus. In cartographic research, eye tracking is a valuable tool for the execution of experiments related to the study of map reading and cartographic design evaluation. An important element of eye movement analysis is the visualization of eye tracking data using techniques referred to the gaze behavior of either individuals or all the subjects in an experiment. Considering that the amount of data collected can blur the reference with the visual stimulus, visualization techniques are usually applied after clustering the gaze recordings in fixations and saccades. A typical visualization is the scan path graph, where fixations are depicted as circles with radical values related to their durations, while saccades are presented as connector line segments among fixations. Other techniques include heat maps and scan path graphs, using variables such as duration, number of fixations, participant percentage etc [15].

In this section we report early progress on the depiction of the gaze route history using a polyline, which is feasible, as the visual trace is generated from sequential raw eye tracking data [11]. The nodes of such a polyline contain information about the duration of fixations or other statistical values, which can also be attributed to line sections that represent saccadic movements. Generally, the reconstruction of gaze route history can be very useful in the study of several cartographic concepts as a gaze polyline depicts the line that is actually perceived from subjects.

The motivation for this work stems from methods used in the inference of graph geometries such as transportation networks, from GPS tracking data. Several such methods rely on trajectory clustering. Some of the algorithms in the literature [98, 38] operate on point data and do not take the temporal aspect into consideration. Others infer curved paths using k-means clustering of raw tracking data along with distance measures [35]; others transform tracking data to discretized images using Kernel Density Estimation (KDE). They function well for frequently sampled and redundant tracking data [14], but are sensitive to noise.

Other approaches relying on computational geometry techniques [7] operate on tracks of high-resolution and accuracy. The final category involves trace-clustering approaches that derive a connected network graph from vehicle trajectories [58]. This approach applies such a technique in eye tracking data to automatically extract "hubs" and construct a polyline that corresponds to the observed geometry of cartographic lines.

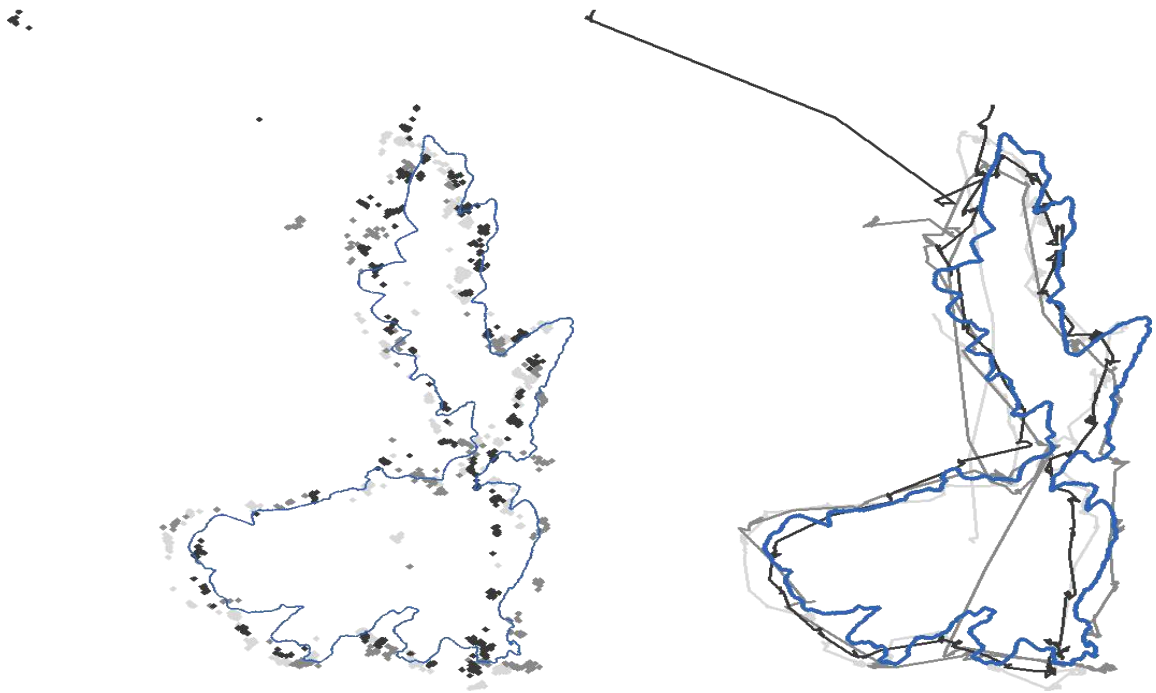


Figure 6.1: Eye Tracking Data from 3 Subjects.

6.2 Inference of Polylines from Eye Tracking Data

The aim of this approach is to derive a single polyline geometry from sampled eye tracking data from multiple users. Figure 6.1 plots tracking data used in our

experiment in blue colour with the actual cartographic line that the subjects have been asked to follow, shown in black.

6.2.1 The Proposed Algorithm

The proposed algorithm to derive the polylines from eye tracking data involves three steps; (i) identifying hubs, (ii) connecting hubs, and (iii) reducing the links into a single geometry, which are discussed in the sequel.

In the first step, we infer hubs from spatial fixation on eye tracking data. A hub represents the spatial fixation that the eye creates near an area of interest. Indicators for hub recognition are the number of tracking samples, the number of different users and the coverage of an extended area of focus. The algorithm takes as input the eye tracking data and determines the k-NNs of each tracking sample, which are subsequently filtered according to the number of users. On these filtered tracking samples, we apply the DBSCAN clustering algorithm using a distance threshold and a minimum number of samples, which depend on the specifics of the experiment. The centroids of the resulting clusters are the hubs. Figure 6.2 shows the hubs derived after applying the hubs inference algorithm in our experimental dataset.

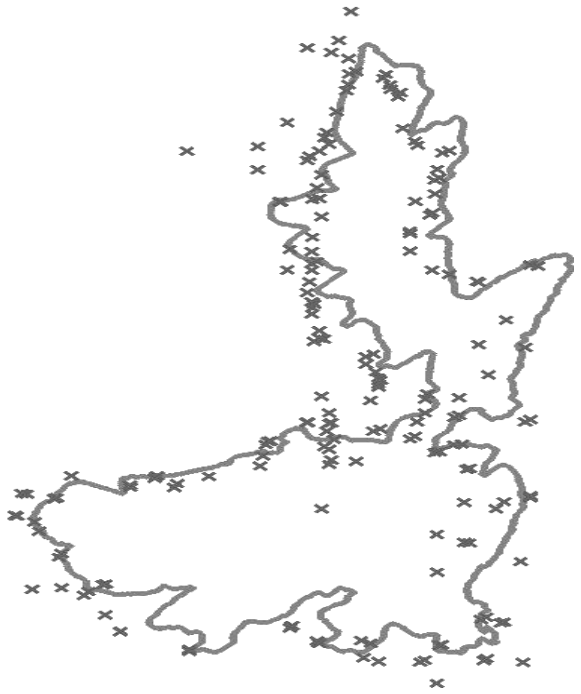


Figure 6.2: Inference of Hubs.

In the following we connect hubs by links. A fringe benefit of the hubs computation based on spatial fixation is that for all data we know which samples helped in identifying hubs. To derive links we exploit this knowledge: for each hub we record the outgoing and/or incoming tracking portions connecting this hub to others by scanning all eye tracking data to discover sequences of hubs. The result of this step is the creation of a sample polyline set that connects hubs with links. In our representation of eye tracking data, all tracking samples that are also hubs are marked as such. Hence, performing a linear scan of all eye tracking data reveals the respective tracking portions that connect hubs.

In the last step of the algorithm we compact links to a reduced geometry expressed by polylines. To this point, we have hubs connected by links derived from eye tracking data that exhibit spatial fixation at these hubs. In a nutshell, the algorithm identifies tracking portions that are close to existing links by means of a buffer region and merges their geometry into the existing link geometry. The size of the buffer region depends on the specifics of the data; in our case we used 15 pixels as buffer region. In this step, we only adjust the geometry of existing links using a three-step algorithm: (i) sort existing link samples in a descending order according to their length, (ii) determine relevant tracking portions using a buffer region around link samples, and (iii) adjust the geometry of links based on the tracking data geometry.

In our experimentation so far we first sort all links according to their length so as to process longer links first as they may be more significant for polyline construction, which remains to be further tested future work. In step (ii) the algorithm uses a buffer region around the examined link sample and retrieves all intersecting portions of other links. New links are created by interpolating link samples and introducing hubs. New links are assigned a weight that is the sum of the weights of the merged links. Link samples are updated several times during this phase. While the examined links are reconstructed, new link samples are created to replace links in previous iterations.

6.2.2 Results

The cartographic line that we try to infer consists of 6595 links (edges) and 6607 nodes. The edges have a length of 4041 pixels, as the reference system is in pixels. Sampling of eye tracking data was at 60 Hz (0.017 sec). Data comes from 3 different users with a total length of 89880 pixels (Figure 6.1). Following the various stages of the polylines inference algorithm, the following output is produced. During the first phase, i.e., hubs extraction and connection, 109 hubs and 300 link samples are generated. The second polylines inference phase, i.e. compacting links, produces 119 hubs, 79 links and a length of 2990 pixels.

This result shows that during the second phase of the algorithm, the number of hubs remains largely constant but only the length of the links connecting them is significantly reduced since we radically merge links during this phase. Figure 6.3 visualizes the inferred polylines in blue and the actual cartographic data in grey colour.

6.3 Summary

In this chapter, we have addressed the problem of automatic polylines inference on the visualization of eye tracking data from observations of cartographic lines. We adopted a modified *TreceBundle* algorithm approach to identify hubs, links between derived hubs and to infer a single polylines geometry from observations of cartographic lines.

This polyline-based visualization of eye tracking data depicts the *average* cartographic line observed by subjects. Clearly, such a visualization is of little use in cases where the context of eye tracking experiments has no lines of some kind that subjects are required to follow. It is, however, quite interesting in cases that such

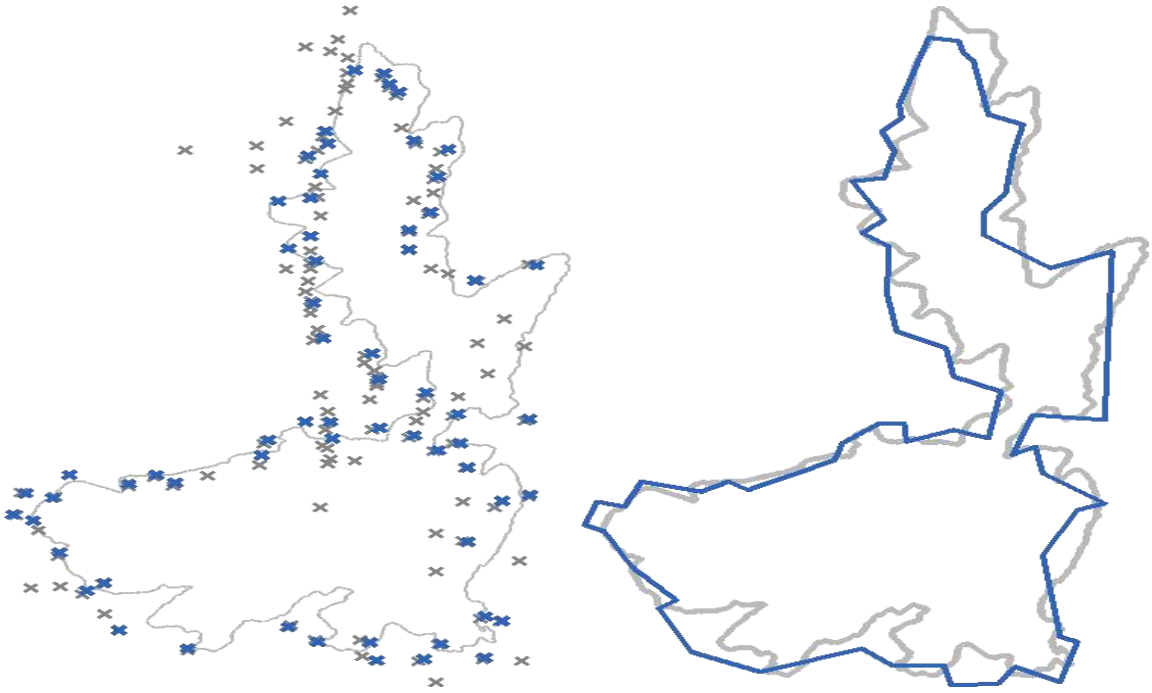


Figure 6.3: Compacting Links and Final Inferred Polyline.

a line really exists, as is the case in cartography where borders, navigation routes and all kinds of curves, are used to represent useful information on a map. Studying the effects of different visualization attributes of cartographic lines in the concentration of the eye's attention to a central linear entity can benefit from using the representation of eye tracking data introduced in this section.

This visualization can be further improved by adding colour attributes to the inferred polyline using calculations such as eye tracking samples data density near the line, or other statistical metrics. Considering that it is the mind that actually does the cognitive interpretation of lines observed, it is rather impossible to infer a polyline that very closely matches the initial cartographic line. However, studying the deviations of individual observers' tracks from the average polyline, and combining the results with semantics from the experiment and subject context may produce some interesting results, too.

Application of the proposed visualization in other kinds of lines whose eye tracking makes sense, as is the case with some medical images, is another area that is definitively worth exploring. Last but not least, the production algorithm of the polyline needs further experimentation on bigger data sets and possibly improvement in few operational aspects.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

This thesis has focused on the provision of evolving and updated transportation maps from vast amounts of trajectory data. Over the last years, vast amounts of geo-referenced trajectory data have been collected due to the ubiquitous availability of positioning technologies such as the Global Positioning System (GPS). In this direction, this thesis has addressed the very timely challenge of analyzing this data. It also proposed novel algorithms for construction and maintenance of digital street maps, which are among the most valuable digital data resource in today's society.

The results of the thesis will benefit a wealth of applications ranging from a variety of location-based services on street maps to the analysis of tracking data for hiking trail map generation or for studying social behavior in animals. The presented work identified and focused on four major issues: a) the inference of road networks from tracking data and the evaluation comparison of various map inference algorithms, b) the inference and maintenance of road networks from sparse tracking data c) the inference of transportation networks from tracking data involving high uncertainty (social media and check-in data), and d) the generalized applications that utilize our approach of bundling sets of traces into geometries.

First, we have dealt with the problem of road network inference and the evaluation comparison of various map inference algorithms. Previous research efforts have focused on map inference from frequently sampled tracking data or have posed strict conditions regarding the geometry of the road networks. Our work elaborates on these approaches and extends them providing more advanced capabilities. In contrast to previous approaches which have been presented for vertically aligned road networks or for sampling rate of $1s$ to $10s$, we have proposed an algorithm which is efficient towards high sampling rates and arbitrary road networks. The outcome was the *TraceBundle* algorithm which exploits the ubiquitous vehicle tracking data in order to analyze, reconstruct and extract road network geometries enriched by attributes. Our approach "bundles" the trajectories around intersection nodes to derive links and consequently the entire geometry of the road network. The *TraceBundle* algorithm addresses the challenges of evolving map data sets, by working towards *automatic map and attribute generation* from massive amounts of vehicle tracking data. The proposed method has been validated through experimental analysis and comparison to existing approaches, showing that it achieves a significant improvement of the precision of the retrieved results. Also, the *mapconstruction.org*

site has been established by making available map construction algorithms, datasets and constructed maps, to motivate other researchers towards the contribution in the area of road network construction algorithms.

Following that, we have focused on sparse tracking data and methods which convert movement trajectories into a hierarchical transportation network. Motivated by the results in the precision of the *TraceBundle* algorithm towards existing approaches, as well as the need for map maintenance and updates, we have emphasized on building a road map hierarchically. For this purpose, we have exploited the types of vehicles movement in an urban context to segment the input data accordingly. The outcome was the *TraceConflation* algorithm which addresses the challenges imposed by noisy, low-sampling rate and sparse trajectories and provides for a mechanism to accommodate automatic map maintenance on updates. It does so by segmenting the input dataset into different groups of trajectories based on their characteristics and then by inferring the network in a layered fashion using dynamically determined values for the parameters. The proposed method has been validated through experimental analysis which shows significant improvements in terms of quality of the constructed road network over existing approaches. In contrast to traditional approaches which build the road network globally and can not cope with large scaled and noisy data, our approach considered the exploitation of segmentation and fusion of the data, is more capable to support updates and to deliver road maps of high spatial accuracy.

Finally, we have considered the problem of inferring and connecting POIs, which encode both geometric and semantic information, from geocoded social media data. Motivated by the fact that the connectivity of such POIs has been overlooked and that it is not obvious how to create meaningful links and networks between them, we proposed the *Network-of-Interest* algorithm. The *Network-of-Interest* algorithm encodes different types of connectivity between POIs and captures peoples type of movement and behavior while visiting these POIs. In contrast to existing approaches which mostly encode geometric information to produce street maps, at this stage we use *behavioral trajectories* as a datasource to also discover transportation infrastructure such as subway maps, bus maps, and hiking trail maps. The proposed method has been validated through experimental analysis and shows that critical transportation infrastructure can be inferred from spatial check-in data obtained by social media applications.

Last but not least, we experimented on generalized applications that utilize the approach of bundling sets of traces into geometries. Towards this direction, we used eye tracking datasets to extract polylines from observations of cartographic lines. Motivated by the great interest in the commercial sector and the increased sophistication and accessibility of eye tracking technologies delivering applications including web usability, advertising, sponsorship and package design, we proposed a method to infer polylines instead of more traditional techniques such as heat maps to visualize eye tracking data. Exploiting the efficiency of the *TraceBundle* algorithm, we presented a polyline-based visualization of eye tracking data that depicts the *average* cartographic line observed by subjects, along with the algorithm that is used to infer this polyline. Application of the proposed visualization in other kinds of lines whose eye tracking makes sense, as is the case with some medical images, is another area that is definitively worth exploring.

All the aforementioned techniques proposed in this thesis, combine, address and

facilitate several major tasks necessary to achieve the overall goal which is to provide evolving and updated maps from vast amounts of trajectory data.

7.2 Future Work

In the previous chapters we have presented in detail the outcomes of the work conducted in the context of this thesis. Still, several research issues remain open. We conclude by identifying and outlining the most prominent ones:

- The problems considered in this thesis are very relevant to trends and requirements that can be identified in other emerging paradigms as well, most notably geomarketing. Geomarketing and in general the the analysis of location-related issues illuminates the regional factors that drive or inhibit growth and optimizes business so it can respond more quickly to market changes and exploit untapped potential before the competition. Geomarketing solutions answer the "where" questions that impact performance and determine who will survive in a competitive marketplace, such as where are your ideal customers located, where are optimal conditions for new business sites, and why or where are areas of unexploited potential in the markets. This work, will facilitate the discovery and analysis of these geo-referenced data, and, consequently, will allow to better target groups and customers in Networks-of-Interest and to adjust/expand the sales territory structures into the most promising areas. Thus, it would be a very interesting challenge to study how the techniques proposed in this thesis can be adapted or extended to provide solutions to such activities.

- Another important issue when inferring tracking data from different sources and with different characteristics is related to the transportation networks. Transportation networks are commonly represented using networks as an analogy for their structure and flows. They belong to the wider category of spatial networks because their design and evolution are physically constrained as opposed with non-spatial networks such as social interactions, corporate organization, and biological systems. The territorial structure of any region corresponds to a network of all its economic interactions. The implementation of networks, however, is rarely premeditated but the consequence of continuous improvements as opportunities arise, investments are made and as conditions change. The setting of networks is the outcome of various strategies, such as providing access and mobility to a region, reinforcing a specific trade corridor or technological developments making a specific mode and its network more advantageous over others. It can be extended to cover various types of links between points along which movements can take place. Thus, it would be very interesting challenge to study how the techniques proposed in this thesis can be applied to integrate noticeable interdependencies among the different nodes and networks over time, based on spatial and functional proximity. It would be also interesting to study means of multimodal transportation from heterogeneous tracking data sources.

- When tracking data are gathered from several heterogeneous sources, potentially of different levels of quality, it may be uncertain, incomplete or inconsis-

tent. This requires advanced techniques in order to manage and reason with such data. Hence, it would be a challenging issue to study how the techniques proposed in this thesis can be adapted to take into account this additional aspect.

- Our work on polylines inference from cartographic data includes a preliminary study on eye tracking data sources. The visualization can be further improved by adding colour attributes to the inferred polyline using calculations such as eye tracking samples data density near the line, or other statistical metrics. Application of the proposed visualization in other kinds of lines whose eye tracking makes sense, as is the case with some medical images, is another area that is definitively worth exploring. Last but not least, the polyline inference algorithm needs further experimentation on bigger data sets and possibly improvement in few operational aspects. Considering that it is the mind that actually does the cognitive interpretation of lines observed, it is rather impossible to infer a polyline that very closely matches the initial cartographic line. However, studying the deviations of individual observers' tracks from the average polyline, and combining the results with semantics from the experiment and subject context may produced some interesting results, too.

Bibliography

[1] *http://geodata.gov.gr/geodata/*.

[2] Mridul Aanjaneya, Frederic Chazal, Daniel Chen, Marc Glisse, Leonidas J. Guibas, and Dmitriy Morozov, *Metric graph reconstruction from noisy data*, Proceedings of the 27th Annual ACM Symposium on Computational geometry, 2011, pp. 37–46.

[3] Gabriel Agamennoni, Juan I. Nieto, and Eduardo M. Nebot, *Robust inference of principal road paths for intelligent transportation systems*, IEEE Transactions on Intelligent Transportation Systems **12** (2011), no. 1, 298–308.

[4] National Geospatial-Intelligence Agency, *Geodesy and geophysics*, 2014.

[5] Mahmuda Ahmed, Kyle S. Hickmann, and Carola Wenk, *Path-based distance for street map comparison*, Computing Research Repository (2014).

[6] Mahmuda Ahmed, Sophia Karagiorgou, Dieter Pfoser, and Carola Wenk, *A comparison and evaluation of map construction algorithms*, GeoInformatica (2014).

[7] Mahmuda Ahmed and Carola Wenk, *Constructing street networks from GPS trajectories*, Proceedings of the 20th Annual European Symposium on Algorithms, 2012, pp. 60–71.

[8] Helmut Alt, Alon Efrat, Günter Rote, and Carola Wenk, *Matching planar maps*, Journal of Algorithms **49** (2003), no. 2, 262–283.

[9] Helmut Alt and Leonidas Guibas, *Discrete geometric shapes: Matching, interpolation, and approximation - a survey*, Handbook of Computational Geometry (1999), 121–154.

[10] Yasuo Asakura and Takamasa Iryo, *Analysis of tourist behaviour based on the tracking data collected using a mobile communication instrument*, Transportation Research Part A: Policy and Practice **41** (2007), no. 7, 684 – 690.

[11] Theodora Bargiota, Vasilis Mitropoulos, Vassilios Krassanakis, and Byron Nakos, *Measuring locations of critical points along cartographic lines with eye movements*, Proceedings of the 26th International Cartographic Conference, 2013.

[12] Albert Baumgartner, Stefan Hinz, and Christian Wiedemann, *Efficient methods and interfaces for road tracking*, International Archives of Photogrammetry and Remote Sensing **34** (2002), 28–31.

[13] James Biagioni and Jakob Eriksson, *Inferring road maps from global positioning system traces: Survey and comparative evaluation*, Transportation Research Record: Journal of the Transportation Research Board **2291** (2012), 61–71.

[14] James Biagioni and Jakob Eriksson, *Map inference in the face of noise and disparity*, Proceedings of the 20th ACM SIGSPATIAL GIS Conference, 2012, pp. 79–88.

[15] Agnieszka (Aga) Bojko, *Informative or misleading - heatmaps deconstructed*, Human and Computer Interaction, New Trends **5610** (2009), 30–39.

[16] Béla Bollobás, Gautam Das, Dimitrios Gunopulos, and Heikki Mannila, *Time-series similarity problems and well-separated geometric sets*, Proceedings of the 13th Annual Symposium on Computational Geometry, 1997, pp. 454–456.

[17] Claude Boucher and Zuheir Altamimi, *Itrs, pz-90 and wgs-84: current realizations and the related transformation parameters*, Journal of Geodesy **75** (2001), 613–619.

[18] Sotiris Brakatsoulas, Dieter Pfoser, Randall Salas, and Carola Wenk, *On map-matching vehicle tracking data*, Proceedings of the 31st International Conference on VLDB, 2005, pp. 853–864.

[19] René Brüntrup, Stefan Edelkamp, Shabid Jabbar, and Björn Scholz, *Incremental map generation with gps traces*, Proceedings of the IEEE Intelligent Transportation Systems, 2005, pp. 574–579.

[20] Chris Brunsdon, *Path estimation from gps tracks*, Proceedings of the 9th International Conference on GeoComputation, 2007.

[21] Lili Cao and John Krumm, *From gps traces to a routable road map*, Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, 2009, pp. 3–12.

[22] Chen Chen and Yinhang Cheng, *Roads digital map generation with multi-track gps data*, Proceedings of the 2008 International Workshops on Education Technology and Training, and on Geoscience and Remote Sensing, IEEE Computer Society, 2008, pp. 508–511.

[23] Daniel Chen, Leonidas J. Guibas, John Hershberger, and Jian Sun, *Road network reconstruction for organizing paths*, Proceedings of the 21st Annual ACM-SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics, 2010, pp. 1309–1320.

[24] Zaiben Chen, Heng Tao Shen, and Xiaofang Zhou, *Discovering popular routes from trajectories*, Proceedings of the 27th International Conference on Data Engineering, 2011, pp. 900–911.

[25] Otfried Cheong, Joachim Gudmundsson, Hyo-Sil Kim, Daria Schymura, and Fabian Stehn, *Measuring the similarity of geometric graphs*, Proceedings of the 8th International Symposium on Experimental Algorithms, 2009, pp. 101–112.

[26] Donatello Conte, Pasquale Foggia, Carlo Sansone, and Mario Vento, *Thirty years of graph matching in pattern recognition*, International Journal of Pattern Recognition and Artificial Intelligence **18** (2004), no. 3, 265–298.

[27] David J. Crandall, Lars Backstrom, Daniel Huttenlocher, and Jon Kleinberg, *Mapping the world's photos*, Proceedings of the 18th International Conference on World Wide Web, 2009, pp. 761–770.

[28] Andrew Crooks, Dieter Pfoser, Andrew Jenkins, Arie Croitoru, Sophia Karagiorgou, Alexandros Efentakis, George Lamprianidis, Duncan Smith, and Anthony Stefanidis, *Crowdsourcing urban form and function*, International Journal of Geographical Information Science (2014).

[29] Peter H. Dana, *The global positioning system overview*, 2000.

[30] Jonathan J. Davies, Alastair R. Beresford, and Andy Hopper, *Scalable, distributed, real-time map generation*, IEEE Pervasive Computing **5** (2006), no. 4, 47–54.

[31] Victor Teixeira de Almeida and Ralf Hartmut Güting, *Supporting uncertainty in moving objects in network databases*, Proceedings of the 13th Annual ACM International Workshop on Geographic Information Systems, 2005, pp. 31–40.

[32] Munmun De Choudhury, Moran Feldman, Sihem Amer-Yahia, Nadav Golbandi, Ronny Lempel, and Cong Yu, *Constructing travel itineraries from tagged geo-temporal breadcrumbs*, Proceedings of the 19th International Conference on World Wide Web, 2010, pp. 1083–1084.

[33] Demitris Delikaraoglou, *The hellenic positioning system (hepos) and its foreseeable implications on the spatial data infrastructure in greece*, Technical Chamber of Greece **28** (2008), 95–103.

[34] Edsger Wybe Dijkstra, *A note on two problems in connection with graphs*, Numerische Mathematik **1** (1959), 269–271.

[35] Stefan Edelkamp and Stefan Schrödl, *Route planning and map inference with global positioning traces*, Computer Science in Perspective (2003), 128–151.

[36] Alexandros Efentakis, Sotiris Brakatsoulas, Nikos Grivas, Giorgos Lamprianidis, Kostas Patroumpas, and Dieter Pfoser, *Towards a flexible and scalable fleet management service*, Proceedings of the 6th ACM SIGSPATIAL International Workshop on Computational Transportation Science, 2013, pp. 79–84.

[37] Brigitte Englisch, *Erhard ertzlaub's projection and methods of mapping*, Imago Mundi **48** (1996), 103–123.

[38] Martin Ester, Hans-Peter Kriegel, Jörg S, and Xiaowei Xu, *A density-based algorithm for discovering clusters in large spatial databases with noise*, Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, 1996, pp. 226–231.

[39] Alireza Fathi and John Krumm, *Detecting road intersections from gps traces*, Proceedings of the 6th International Conference on Geographic information science, Springer-Verlag, 2010, pp. 56–69.

[40] Alireza Fathi and John Krumm, *Inferring the road network from gps data*, Geographic Information Science **6292** (2010), 56 – 69.

[41] OpenStreetMap Foundation, May 2014.

[42] Xinbo Gao, Bing Xiao, Dacheng Tao, and Xuelong Li, *A survey of graph edit distance*, Pattern Analysis and Applications **13** (2010), 113–129.

[43] Georg Gati, *Further annotated bibliography on the isomorphism disease*, Journal of Graph Theory **3** (1979), no. 2, 95–109.

[44] Xiaoyin Ge, Issam Safa, Mikhail Belkin, and Yusu Wang, *Data skeletonization via Reeb graphs*, Proceedings of the 25th Annual Conference on Neural Information Processing Systems, 2011, pp. 837–845.

[45] Fabien Girardin, Francesco Calabrese, Filippo D. Fiore, Carlo Ratti, and Josep Blat, *Digital footprinting: Uncovering tourists with user-generated content*, IEEE Pervasive Computing Magazine **7** (2008), 36–43.

[46] Michael F. Goodchild, *Citizens as voluntary sensors: spatial data infrastructure in the world of web 2.0*, International Journal of Spatial Data Infrastructures Research **2** (2007), 24–32.

[47] Kazuaki Iwamura Guo, Tao and Masashi Koga, *Towards high accuracy road maps generation from massive gps traces data*, Proceedings of the IEEE International Geoscience and Remote Sensing Symposium, IEEE, 2007, pp. 667–670.

[48] Andreas Hackeloeer, Klaas Klasing, Jukka Matthias Krisp, and Liqiu Meng, *Georeferencing: a review of methods and applications*, Annals of GIS (2014), 61–69.

[49] Mordechai Muki Haklay and Patrick Weber, *Openstreetmap: User-generated street maps*, IEEE Pervasive Computing **7** (2008), no. 4, 12–18.

[50] A. James Harrell and V. Max Brown, *The oldest surviving topographical map from ancient egypt (turin papyri 1879, 1899 and 1969)*, Journal of the American Research Center in Egypt **29** (1992), 81–105.

[51] J.A. Harrell and V.M. Brown, *The world's oldest surviving geological map - the 1150 bc turin papyrus from egypt*, Journal of Geology **100** (1992), 3–18.

[52] Jon S. Horne, Edward O. Garton, Stephen M. Krone, and Jesse S. Lewis, *Analyzing animal movements uing brownian bridges*, Ecology **88** (2007), 2354–2363.

[53] Kathleen Hornsby and Max J. Egenhofer, *Modeling moving objects over multiple granularities*, Annals of Mathematics and Artificial Intelligence **36** (2002), 1–2.

[54] Jiuxiang Hu, Anshuman Razdan, John Femiani, Ming Cui, and Peter Wonka, *Road network extraction and intersection detection from aerial images by tracking road footprints*, IEEE T. Geoscience and Remote Sensing **45** (2007), no. 12-2, 4144–4157.

[55] Sera Jang, Taehwan Kim, and Eunseok Lee, *Map generation system with lightweight gps trace data*, Proceedings of the 12th International Conference on Advanced communication technology, 2010, pp. 1489–1493.

[56] Evangelos Kalogerakis, Olga Vesselova, James Hays, Alexei A. Efros, and Aaron Hertzmann, *Image sequence geolocation with human travel priors*, Proceedings of the 11th International Conference on Computer Vision, 2009, pp. 253–260.

[57] Sophia Karagiorgou, Vassilios Krassanakis, Vassilios Vescoukis, and Byron Nakos, *Experimenting with polylines on the visualization of eye tracking data from observations of cartographic lines*, Proceedings of the 2nd International Workshop on Eye Tracking for Spatial Research, 2014.

[58] Sophia Karagiorgou and Dieter Pfoser, *On vehicle tracking data-based road network generation*, Proceedings of the 20th ACM SIGSPATIAL GIS Conference, 2012, pp. 89–98.

[59] Sophia Karagiorgou, Dieter Pfoser, and Dimitrios Skoutas, *Segmentation-based road network construction*, Proceedings of the 21th ACM SIGSPATIAL GIS Conference, 2013, pp. 470–473.

[60] Balázs Kégl, Adam Krzyzak, Tamás Linder, and Kenneth Zeger, *Learning and design of principal curves*, IEEE Trans. Pattern Anal. Mach. Intell. **22** (2000), no. 3, 281–297.

[61] Slava Kisilevich, Daniel A. Keim, and Lokach Rokach, *A novel approach to mining travel sequences using collections of geo-tagged photos*, Proceedings of the 13th AGILE International Conference on Geographic Information Science, 2010, pp. 163–182.

[62] Slava Kisilevich, Florian Mansmann, and Daniel Keim, *P-dbscan: A density based clustering algorithm for exploration and analysis of attractive areas using collections of geo-tagged photos*, Proceedings of the 1st International Conference and Exhibition on Computing for Geospatial Research and Application, 2010, pp. 38:1–38:4.

[63] Felix Kling and Alexei Pozdnoukhov, *When a city tells a story: Urban topic analysis*, Proceedings of the 20th International Conference on Advances in Geographic Information Systems, 2012, pp. 482–485.

[64] Bart Kuijpers, Bart Moelans, Walied Othman, and Alejandro A. Vaisman, *Analyzing trajectories using uncertainty and background information.*, 11th International Symposium on Spatial and Temporal Databases, vol. 5644, 2009, pp. 135–152.

[65] Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang, *Trajectory clustering: a partition-and-group framework*, Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data, 2007, pp. 593–604.

[66] Xuemei Liu, James Biagioni, Jakob Eriksson, Yin Wang, George Forman, and Yanmin Zhu, *Mining large-scale, sparse gps traces for map inference: comparison of approaches*, Proceedings of the 18th ACM SIGKDD Conference, 2012, pp. 669–677.

[67] Yuncai Liu, *An automation system: generation of digital map data from pictorial map resources*, Pattern Recognition **35** (2002), no. 9, 1973–1987.

[68] Stuart Lloyd, *Least squares quantization in pcm*, IEEE Transactions on Information Theory **28** (2006), no. 2, 129–137.

[69] Bob Mckercher and Gigi Lau, *Movement patterns of tourists within a destination*, Tourism Geographies **10** (2008), no. 3, 355–374.

[70] Juliane Mondzech and Monika Sester, *Quality analysis of openstreetmap data based on application needs*, Cartographica **46** (2011), 115–125.

[71] Scott Morris and Kobus Barnard, *Finding trails*, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2008.

[72] Brian Niehofer, Ralf Burda, Christian Wietfeld, Franziskus Bauer, and Oliver Lueert, *Gps community map generation for enhanced routing methods based on trace-collection by mobile phones*, Proceedings of the 1st International Conference on Advances in Satellite and Space Communications, 2009, pp. 156–161.

[73] Jennifer Ogle, Randall Guensler, William Bachman, Maxim Koutsak, and Jean Wolf, *Accuracy of global positioning system for determining driver performance parameters*, Transportation Research Record **1818** (2002), 12–24.

[74] OpenStreetMap Foundation, *Openstreetmap: User-generated street maps*, 2013.

[75] Nikos Pelekis, Ioannis Kopanakis, Gerasimos Marketos, Irene Ntoutsi, Gennady Andrienko, and Yannis Theodoridis, *Similarity search in trajectory databases*, Proceedings of the 14th International Symposium on Temporal Representation and Reasoning, 2007, pp. 129–140.

[76] Dieter Pfoser and Christian S. Jensen, *Capturing the uncertainty of moving-object representations*, Proceedings of the 6th International Symposium on Advances in Spatial Databases, 1999, pp. 111–132.

[77] Dieter Pfoser and Christian S. Jensen, *Trajectory indexing using movement constraints*, GeoInformatica **9** (2005), no. 2, 93–115.

[78] Mohammed A. Quddusa, Washington Y. Ochiengb, and Robert B. Nolandb, *Current map-matching algorithms for transport applications: State-of-the art and future research directions*, Transportation Research Part C: Emerging Technologies (2007), 312–328.

[79] Tye Rattenbury, Nathaniel Good, and Mor Naaman, *Towards automatic extraction of event and place semantics from flickr tags*, Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 2007, pp. 103–110.

[80] Ronald C. Read and Derek G. Corneil, *The graph isomorphism disease*, Journal of Graph Theory **1** (1977), no. 4, 339–363.

[81] Injong Rhee, Minsu Shin, Seongik Hong, Kyunghan Lee, Seong Joon Kim, and Song Chong, *On the levy-walk nature of human mobility*, IEEE/ACM Transactions on Networking **19** (2011), no. 3, 630–643.

[82] Seth Rogers, Pat Langley, and Christopher Wilson, *Mining gps data to augment road models*, Proceedings of the 5th International Conference on Knowledge Discovery and Data Mining, ACM Press, 1999, pp. 104–113.

[83] Seth Rogers and Stefan Schroedl, *Creating and evaluating highly accurate maps with probe vehicles*, IEEE Conference on Intelligent Transportation Systems, 2000.

[84] Stefan Schrödl, Kiri Wagstaff, Seth Rogers, Pat Langley, and Christopher Wilson, *Mining gps traces for map refinement*, Data mining and knowledge Discovery **9** (2004), 59–87.

[85] Wenhuan Shi, Shuhan Shen, and Yuncai Liu, *Automatic generation of road network map from massive gps vehicle trajectories*, Proceedings of the 12th International Conference on Intelligent Transportation Systems, 2009, pp. 48–53.

[86] Karagiorgou Sophia, Pfoser Dieter, and Skoutas Dimitrios, *Geosemantic network-of-interest construction using social media data*, Proceedings of the 8th International Conference on Geographic Information Science, 2014.

[87] Albert Steiner and Axel Leonhardt, *Map generation algorithm using low frequency vehicle position data*, Proceedings of the 90th Annual Meeting of the Transportation Research Board, January 2011, pp. 1–17.

[88] Mohamad Tavakoli and Azriel Rosenfeld, *Building and road extraction from aerial photographs*, IEEE Transactions on Systems, Man and Cybernetics **12** (1982), no. 1, 84–91.

[89] Chalko J. Tom, *High accuracy speed measurement using gps (global positioning system)*, NU Journal of Discovery (2007).

[90] Goce Trajcevski, *Uncertainty in spatial trajectories*, Computing with Spatial Trajectories, 2011, pp. 63–107.

[91] Goce Trajcevski, Ouri Wolfson, Klaus Hinrichs, and Sam Chamberlain, *Managing uncertainty in moving objects databases*, ACM Transactions on Database Systems **29** (2004), no. 3, 463–507.

[92] Yin Wang, Xuemei Liu, Hong Wei, George Forman, Chao Chen, and Yanmin Zhu, *Crowdatlas: Self updating maps for cloud and personal use*, Proceedings of the 11th MobiSys Conference, 2013.

[93] Andrew Wilson, *Galileo: The european program for global navigation*, 2003.

[94] Stewart Worrall and Eduardo Nebot, *Automated process for generating digitised maps through gps data compression*, GPS Data Compression, in Australasian Conference on Robotics and Automation, 2007.

[95] Keiji Yanai, Hidetoshi Kawakubo, and Bingyu Qiu, *A visual analysis of the relationship between word concepts and geographical locations*, Proceedings of the ACM International Conference on Image and Video Retrieval, 2009, pp. 13:1–13:8.

[96] Zhiping Zeng, Anthony K. H. Tung, Jianyong Wang, Jianhua Feng, and Lizhu Zhou, *Comparing stars: on approximating graph edit distance*, Proceedings of the 35th VLDB Conference, 2009, pp. 25–36.

[97] Lijuan Zhang, Frank Thiemann, and Monika Sester, *Integration of gps traces with road map*, Proceedings of the 3rd ACM SIGSPATIAL International Workshop on Computational Transportation Science, 2010, pp. 17–22.

[98] Tian Zhang, Raghu Ramakrishnan, and Miron Livny, *Birch: An efficient data clustering method for very large databases*, Proceedings of the 25th ACM SIGMOD International Conference on Management of Data, 1996, pp. 103–114.

[99] Huijing Zhao, Jun Kumagai, Masafumi Nakagawa, and Ryosuke Shibasaki, *Semi automatic road extraction from high resolution satellite image*, Journal of Applied Sciences (2008), 3431–3438.

[100] Yan-Tao Zheng, Zheng-Jun Zha, and Tat-Seng Chua, *Mining travel patterns from geotagged photos*, ACM Transactions on Intelligent Systems and Technology **3** (2012), no. 3, 56:1–56:18.

[101] Yu Zheng, Xing Xie, and Wei-Ying Ma, *Proceedings of the 10th international conference on mobile data management: Systems, services and middleware*, vol. 33, 2010, pp. 32–39.

Chapter 8

Curriculum Vitae

Contact Information

Research Center "Athena"
Institute for the Management of Information Systems
Artemidos 6 and Epidavrou
Marousi 15125, Greece

Telephone: (+30) 210 6875430
Fax: (+30) 210 6856804
E-mail: karagior@gmail.com

Education

- **National Technical University of Athens**, Greece (2009–2014)
  PhD, School of Rural and Surveying Engineering
  Title: Inference of Transportation Networks from Sparse Tracking Data
  Supervisor: Assist. Prof. Vassilios Vescoukis

- **University of Thessaly**, Greece (2006–2009)
  MSc, Department of Electrical and Computer Engineering
  Grade: 8.75/10
  Title: Supporting Service Differentiation in Wireless Sensor Networks
  Supervisor: Prof. Georgios Stamoulis

- **University of Crete**, Greece (2001–2006)
  BSc, Computer Science Department
  Grade: 7.02/10
  Thesis: Throughput Differentiation in IEEE 802.11e Wireless LANs
  Supervisor: Assist. Prof. Vassilios Siris

Research Interests

- Spatio-temporal databases

- Transportation networks
- Map inference
- Knowledge extraction

Academic Experience

- **National Technical University of Athens**, Greece (2010-2013)
  Teaching Assistant
  - Database Systems (Fall 2013)
  - Database Systems (Fall 2012)
  - Database Systems (Fall 2011)
  - Database Systems (Fall 2010)

Technical Skills

- **Programming:**: C/C++, Java, C Sharp, Python
- **Operating Systems**: Linux, Unix, MacOS, iOS, Windows
- **Database Systems**: MySQL, PostgreSQL, MS SQL Server, Oracle

Foreign Languages

- English (FCE Cambridge)
- French (Delf I, Delf II)

Hobbies

Traveling, cinema, music, volleyball, reading