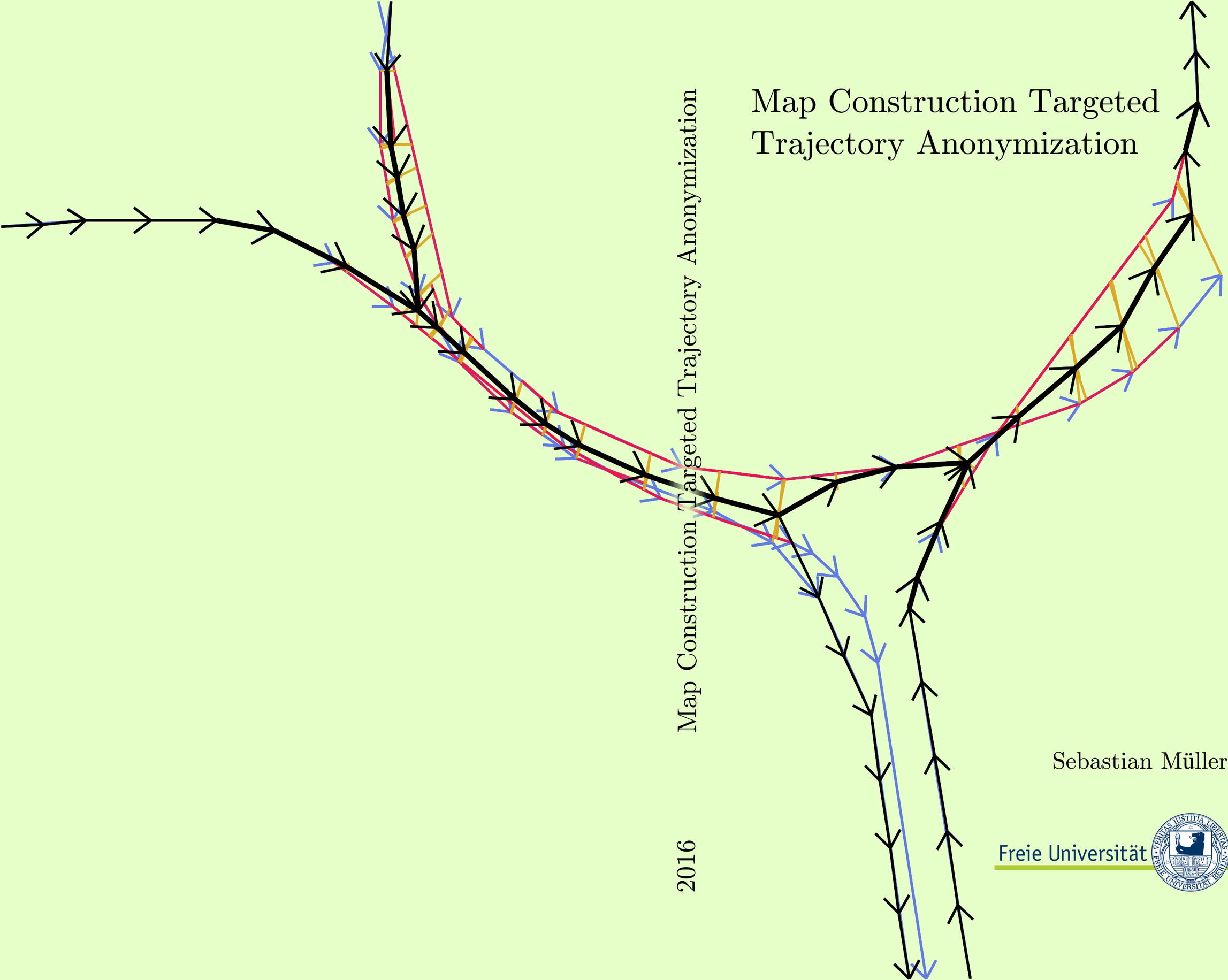


Map Construction Targeted Trajectory Anonymization

Map Construction Targeted Trajectory Anonymization

2016

Sebastian Müller, 2016

Freie Universität Berlin

Map Construction Targeted Trajectory Anonymization

Dissertation zur Erlangung des Grades
eines Doktors der Naturwissenschaften (Dr. rer. nat.)
am Fachbereich Mathematik und Informatik
der Freien Universität Berlin

von

Sebastian Müller

Berlin
eingereichtes Exemplar vom 29.06.2015
mit Änderungen vom 16.03.2016

- Die Betreuerin der Arbeit ist Professorin Agnès Voisard[1].
- Die Gutachter der Arbeit sind Professorin Agnès Voisard und Professor Dieter Pfoser[2].
- Die Disputation fand am 21. Oktober 2015 am Institut für Informatik der Freien Universität statt.

[1]Agnès Voisard, Ph.D.
Institut für Informatik
Takustr. 9
14195 Berlin
Germany

[2]Dieter Pfoser, Ph.D.
Department of Geography and Geoinformation Science
George Mason University
Exploratory Hall, Room 2203
4400 University Drive
Fairfax, VA, 22032
United States of America

Acknowledgements

While declaring this thesis as solely my own work, I have to admit, it wasn't done alone in a basement abandoned from the rest of the world. Without the help of others this thesis would have been very different from what it is at present. As I am proud of its current status, I feel I have to thank the following people for their involvement during the time of writing this thesis.

First, I would like to thank my advisor Prof. Agnès Voisard. She gave me the honor and opportunity of being her first employee at the working group for database systems and information systems. With regards to this thesis, she always ensured that I had enough time and freedom to work on my Ph.D. I appreciate her contribution in terms of constructive feedback and ideas very much. I also want to thank her for being my 1st reviewer and part of the committee of my Ph.D. defense.

My 2nd reviewer is Prof. Dieter Pfoser who generously accepted my request for reviewing. I very much appreciate his contributions to geographical information systems and more particularly, I very much liked his moderation of the Geocrowd workshop at ACM GIS 2013.

I would like to thank the members of the defense committee, including Prof. Heinz Schweppe, Prof. Wolfgang Mulzer, and Dr. Frank Hoffmann. Prof. Heinz Schweppe, predecessor of Prof. Agnès Voisard as the working group leader, showed interest in my Ph.D. research and gave me very helpful advices for teaching. Prof. Wolfgang Mulzer has conducted important work on the computation of the Fréchet distance. I am using implementations of his co-author Prof. Wouter Meulemans in my work.

Prof. Oliver Günther offered me my first job in academia which I very much appreciate. I consider accepting this job offer as one of my best decisions. I would also like to thank Dr. habil. Benjamin Fabian for being my supervisor during that time and for sharing his insights and helping me at the start of my research career.

I would like to thank my colleagues at Freie Universität Berlin and Humboldt-Universität zu Berlin. In Particularl, I would like to mention Steffen Kunz,

Franziska Brecht and Paras Mehta with whom I successfully worked and published my first research papers.

I would like to thank all students who contributed to the Agg2Graph project. They are in chronological order Jens Fischer, Franz Gatzke, Johannes Mitlmeier, Marc Simons, Manuel Kotlarski, Sebastian Barthel, Ferhat Beyaz, Damla Durmaz, Martinus Dipobagio, Serdar Tosun, Christian Windolf, Martin Liesenberg, and Daniel Neumann. Out of all these wonderful contributions, a special thanks goes to Johannes Mitlmeier whose components have had the strongest impact.

Lastly I wish to thank my parents, Renate and Jürgen Müller, my girlfriend, Paula Schmieder, and close friends, whose enthusiasm, interest and support in this project have given me the motivation to realize this achievement.

Contents

Acknowledgements **5**

List of Figures **11**

List of Tables **17**

1 Introduction **19**
1.1 Aim . . . 19
1.2 Motivation . . . 20
1.2.1 Environment . . . 20
1.2.2 Demand . . . 22
1.3 Definitions . . . 24
1.4 Organization . . . 26

2 Background **27**
2.1 Related Work . . . 27
2.1.1 Work on Trajectory Anonymization . . . 27
2.1.2 Work on Map Construction . . . 29
2.1.3 Applied Methods . . . 29
2.2 Aggregation Process . . . 35
2.2.1 Network Architecture: Client-Server and Peer to Peer . . . 35
2.2.2 Iterative Process . . . 37
2.2.3 Privacy Threats . . . 37
2.2.4 Privacy Measures . . . 38
2.2.5 Security Measures . . . 38
2.2.6 Data Cleaning . . . 39
2.2.7 Functional Architecture . . . 40

3 Matching of Trajectories **43**
3.1 Point-Based Distances . . . 43
3.2 Edge-Based Distances . . . 44

3.3 Trajectory-Based Distances . . . 48
3.4 Angle Components . . . 51
3.5 Iterative Matching . . . 55
3.6 Global Matching . . . 57
3.7 Conclusion . . . 58

4 Merging of Trajectories 63
4.1 Point-based Merging . . . 64
4.2 Edge-based Merging . . . 66
4.3 Trajectory-based Merging . . . 70
4.3.1 Merging with Normalization . . . 70
4.3.2 Sweep Algorithm . . . 72
4.4 Merging: Finalization Steps . . . 74
4.5 Conclusion . . . 76

5 Privacy Preservation 77
5.1 k-Anonymity for Trajectories . . . 78
5.2 Integration in Matching and Merging . . . 80
5.2.1 k-Anonymity and weight . . . 80
5.2.2 Finding Subtrajectories . . . 82
5.3 Differential Privacy for Trajectories . . . 84
5.4 Conclusion . . . 84

6 Evaluation 87
6.1 Synthetic Cases . . . 87
6.1.1 Introduction to the Cases . . . 87
6.1.2 Results with Edge-Based Algorithm . . . 89
6.1.3 Results with Subtrajectory-Based Algorithm . . . 95
6.2 Demo Scenarios . . . 98
6.2.1 Introduction to the Cases . . . 98
6.2.2 Results with Edge-Based Algorithm . . . 101
6.2.3 Results with Subtrajectory-Based Algorithm . . . 105
6.3 Conclusion . . . 107

7 Conclusion 111
7.1 Limitations . . . 112
7.2 Lessons Learned . . . 112
7.3 Future Work . . . 114

8 Bibliography 117

A Graphical User Interface **125**
A.1 Installation and Start 125
A.2 Import 128
A.3 Filter 128
A.4 Cleaning 128
A.5 Aggregation 131
A.6 Road Generation 135
A.7 Export 136

B Zusammenfassung **137**

C Abstract **139**

D Selbstständigkeitserklärung **141**

List of Figures

2.1 Trajectory generalization by Voronoi tessellation [54] 28
2.2 Trajectory generalization via point matching [8] 29
2.3 Road map construction from trajectories [17] 29
2.4 Comparing Hausdorff distance with Fréchet distance [67] 30
2.5 TRACLUS distance function elements [43] 32
2.6 TRACLUS sweep function for representative trajectories [43] . . 32
2.7 Client-Server Network Structure 36
2.8 Peer-To-Peer Network Structure 37
2.9 Functional Architecture of Agg2Graph 41

3.1 Two directed edges with extension and perpendiculars marking the crossings of perpendicular and edges or their extension including the angle between the two edges 46
3.2 Comparison between measuring the distance between start and end points or between points and crossings of the perpendicular[55] 47
(a) Difference in length 47
(b) Difference in angle 47
(c) Low difference in start points 47
(d) High difference in start points 47
3.3 Two parallel edges with extension and perpendiculars marking the crossings of perpendicular and edges or their extension with a high shift to each other 49
3.4 Two GPS traces and their free-space diagram [27] 51
(a) GPS traces 51
(b) Free-space diagram 51
3.5 A difference in angle at a highway departure 52
3.6 The aggregation represents a highway and the individual trace represents a departure from the highway 53
3.7 Iterative matching of trajectories 55
3.8 Matching of two edges to one edge 56

3.9 Partial matching of edges to the aggregation 57
3.10 Marking the affected area by using a bounding box with threshold 58
3.11 Global edge-based matching process 59

4.1 Point-based merging example with two trajectories 64
4.2 Point-based merging example with one new point for sets of points 65
4.3 Point-based merging example with one new point for each iteration of the aggregation 66
4.4 Edge-based merging example with iteration of the aggregation and averaging of start and end nodes 67
4.5 Edge-based merging example with iteration of the aggregation and averaging of start and end nodes and additional merging of displaced nodes 67
4.6 Edge-based merging example with iteration of the aggregation and finding new nodes along the perpendicular 68
4.7 Increasing the granularity of the trajectory aggregation by creating new points at the crossings of the perpendiculars of the edge of the single trajectory with the trajectory aggregation 69
4.8 Merging trajectories by merging points found by distance thresholds 72
4.9 Merging trajectories by generating points when the sweep-line detects changes 73
4.10 Adjusting the sweep-line backwards to prevent leaving out nodes 74
4.11 Increasing the granularity of the trajectory aggregation by creating new points at the crossings of the perpendiculars of the edge of the single trajectory with the trajectory aggregation 75

5.1 Obfuscation methods for location-based data[13] 79
(a) Deletion 79
(b) Randomizing 79
(c) Discretizing 79
(d) Subsampling 79
(e) Mixing 79
5.2 Using the edge weight to match edges according to k as threshold 81
5.3 Iterative matching of trajectories within a filter step 83
5.4 The identification of characteristic points of the single trajectory next to the trajectory aggregation 84
(a) Single trajectory with GPS errors inducing characteristic points 84
(b) Single trajectory leaving trajectory aggregation 84

6.1 Synthetic cases of GPS traces for aggregation 90

(a) Converging and diverging traces 90
(b) Converging two times 90
(c) Shorter and longer traces 90
(d) Traces with imprecision 90
(e) Converging, cCrossing and diverging 90
(f) Crossing 90
(g) Converging, crossing and diverging in opposite direction . . 90
(h) Crossing with a curve 90
6.2 Synthetic cases of GPS trace aggregation with edge-based matching and merging 94
(a) Converging and diverging traces 94
(b) Converging two times 94
(c) Shorter and longer traces 94
(d) Traces with imprecision 94
(e) Converging, crossing and diverging 94
(f) Crossing 94
(g) Converging, crossing and diverging in opposite direction . . 94
(h) Crossing with a curve 94
6.3 Two possible crossings with the perpendicular 95
6.4 Possible results with iterative matching and a subtrajectory distance 97
(a) Iterative matching result 1 97
(b) Iterative matching result 2 97
(c) Iterative matching result 3 97
(d) Iterative matching result 4 97
6.5 Synthetic cases of GPS trace aggregation with subtrajectory-based matching and merging 99
(a) Converging and diverging traces 99
(b) Converging two times 99
(c) Shorter and longer traces 99
(d) Traces with imprecision 99
(e) Converging, crossing and diverging 99
(f) Crossing 99
(g) Converging, crossing and diverging in opposite direction . . 99
(h) Crossing with a curve 99
6.6 Real cases (in Berlin) of GPS trace aggregation 102
(a) Street crossing 102
(b) T-crossing 102
(c) Highway drive up 102
(d) Highway departure 102

(e) Entering and leaving of a circle . . . 102
(f) Highway below street . . . 102
(g) Circle traffic . . . 102
(h) A place amid lanes . . . 102
6.7 Real cases (in Berlin) of GPS trace aggregation with edge-based matching and merging . . . 106
(a) Street crossing . . . 106
(b) T-crossing . . . 106
(c) Highway drive up . . . 106
(d) Highway departure . . . 106
(e) Entering and leaving of a circle . . . 106
(f) Highway below street . . . 106
(g) Circle traffic . . . 106
(h) A place amid lanes . . . 106
6.8 Real cases (in Berlin) of GPS trace aggregation with subtrajectory-based matching and merging . . . 108
(a) Street crossing . . . 108
(b) T-crossing . . . 108
(c) Highway drive up . . . 108
(d) Highway departure . . . 108
(e) Entering and leaving of a circle . . . 108
(f) Highway below street . . . 108
(g) Circle traffic . . . 108
(h) A place amid lanes . . . 108
6.9 Expansion at connections . . . 109

7.1 Street crossing (in Berlin) GPS trace aggregation with edge-based matching and merging . . . 113
(a) All improvements included . . . 113
(b) Without edge length limitation . . . 113
(c) Without merging of near connections . . . 113
(d) Without checking for existing connections . . . 113
7.2 Place (in Berlin) GPS trace aggregation with subtrajectory-based matching and merging . . . 114
(a) All improvements included . . . 114
(b) Without perpendicular angle and length restriction . . . 114

A.1 GUI start screen . . . 127
A.2 GUI screen after input . . . 129
A.3 GUI screen after filter . . . 130

A.4 GUI screen after cleaning 132
A.5 GUI screen after aggregation 134
A.6 GUI screen after road generation 136

List of Tables

2.1 Patient records, raw data 33
2.2 Patient records, 2-anonymized data 33

3.1 Execution times of point-based distances 45
3.2 Distance execution times 61

6.1 Location information for real-world scenarios 100

Chapter 1

Introduction

This thesis reports on research conducted in the field of applied computer science. More precisely, it can be categorized to data privacy on the one hand and to computational geometry on the other hand. The kind of science falls into the category design science [39]. I write about a design and a prototypical implementation which solve a defined problem in a certain way. The degree to which the problem is solved and the effectiveness of the solution can only be estimated and not proven. I evaluate my work with qualitative and quantitative measures that indicate the quality of the solution. Whenever possible, there is a comparison to a state-of-the-art solution. In the following sections I describe the aim of the thesis, its motivation and necessary definitions which are repeatedly used throughout the thesis.

This chapter is organized as follows: Section 1.1 describes the aim of this thesis and Section 1.2 the motivation, including the environment and the demand. Section 1.3 gives defintions which are used throughout the thesis and the last section of this chapter deals with the organization of the following chapters.

1.1 Aim

The aim of this thesis is to anonymize trajectory data with the goal of constructing a map in a consecutive step. Trajectory data, even without direct identifiers, can directly be linked to a person if it is comprehensive enough [48, 3, 41]. Hence, trajectory data is personal data and should be treated accordingly. Therefore, to distribute and analyze the data it should be anonymized in a way that no personal link is possible afterwards. This is also a claim of the European data protection directive [36]. Specializing anonymization for map construction can provide many benefits in providing both privacy (anonymity) and information utility. Map construction from trajectory data can be used to

create maps for territories where no manually created maps are available, e.g., company campuses[17] and rural areas. It also can be used to create maps for special transportation modes, e.g., inline skating and sailing.

1.2 Motivation

To automatically construct a road network, the amount of available data is important. If everything else is constant, the higher the amount of data, the better the expected result. For example Karagiorgou and Pfoser[42] state in their summary "Visual inspection shows that the generated road network closely resembles the actual road network if sufficient tracking data that provides redundant coverage of the road network is available". According to Biagioni and Eriksson[5], Niehofer et al.[59] show that the relative position error of a road segment rapidly decreases with increasing amounts of data. Internet users tend to prefer anonymized communication and are willing to accept disadvantages, including latency, to a certain degree [10, 11]. This is also true for trajectory data: people are more willing to contribute their individual trajectories if they are anonymized or obfuscated [13]. Conclusively, more anonymity will attract more users, more users generate more data, and more data will lead to better results in map construction. Nevertheless, anonymization methods lead to an information loss, which increases with stricter measures [12]. One argument is that the loss of information might outweigh the information gain by having a higher amount of data available. However, anonymization does not necessarily need to be applied to all the data, it can be an option for users who are more privacy-aware and would not contribute their data unanonymized.

Automatic map construction from GPS traces (trajectory data) is feasible and inexpensive [17]. It is more up-to-date than other approaches and contributors do not need to have expert knowledge. OpenStreetMap[37], for example, uses a manual approach where the input are GPS traces and satellite images and often a volunteer contributes by editing the map. Automatic map construction can help navigation systems to detect road changes over time and creating maps where manual map construction is not cost-effective, e.g., for sports-map creation and for having maps available for special purposes or special campi.

1.2.1 Environment

This section concerns the environment that on the one hand enables the proposed work and on the other hand shows a demand for this work. Particularly important are advances in technologies to retrieve one's location and to share data in a timely manner.

Satellite navigation

Satellite navigation provides geo-spatial positioning. Nowadays, the main systems for satellite navigation available for private use are GPS (Global Positioning System, United States of America) and GLONASS (Globalnaya navigatsionnaya sputnikovaya sistema, int.: Global Navigation Satellite System, Russia). GPS and GLONASS use two different reference ellipsoids, WGS-84 and PZ-90 [9]. For combined usage the GPS/GLONASS receiver has to calculate transformations to use a higher coverage of satellites offered by both of the systems. Exemplary practical tests show the advantage of the combined use of both systems, particularly for areas with obstacles [61]. Nevertheless, as for now, the data of concern for this work are practically data from GPS (Global Positioning System) which are positional data calculated by a GPS receiver with the help of time delays of satellite data. The result of the calculation is a point in 3-dimensional space and time described by latitude, longitude, altitude and a timestamp. These data can be collected over time by the GPS receiver. For the purpose of this work, it is not relevant by which satellites the data was retrieved. Therefore, future use of more satellites would only enhance precision and results. Particularly interesting, for the improvement of the precision, might be the launch of the Galileo (Europe) with a claimed error of less than one meter for civilian use [20].

Mobile Phones

For the purpose of this work we are interested in smartphones, which provide mobile Internet access and global positioning. According to Ericsson[26] there is an estimation of 1.9 billion smartphone subscriptions in 2013 and an estimation of 5.6 billion smartphone subscriptions in 2019. The estimation is based on a current level of mobile phone subscriptions and a replacement rate of mobile phones with smartphones. This indicates a high amount of users and potential use of this work. Furthermore, not only the availability is increasing, but also the actual usage. This is indicated by an estimated increase of high-bandwidth subscriptions. Ericsson[26] estimates 2.6 billion LTE (Long Term Evaluation) subscriptions in 2019.

Users

There is no typical smartphone user and the use of smartphones is highly individual. For our purposes it is important to know, how many users use applications based on maps, how many use their smartphone for supporting their sports-activities, how many contribute in which way to map providers. Additionally,

important is, how privacy-aware users are and how much their willingness to contribute depends on privacy-measures.

According to GlobalWebIndex[30] Google Maps[33] is at the time of writing the most common used mobile application with a percentage share of 54 of all mobile users. Ovi Maps (now Here[60]) is a popular map based application, ranking 11th with a 9% share. BBBike[63], a more specialized map application, for bicyclists in Berlin and Brandenburg, has 10.000 to 50.000 installations on Android[31] according to Rezic[32]. Runtastic[64], a popular sports application has 5.000.000 to 10.000.000 installations on Android[34]. Conclusively, there is a high interest in mobile map applications in general, but also for sport applications which use maps or navigation applications for bikes or other sporty vehicles.

1.2.2 Demand

This section discusses the demand for applications which make use of automatic map construction. We can categorize these demands in applications based on maps which can be automatically created, the use of additional information in existing maps which can automatically be added, and the creation and update of existing maps.

Applications based on Maps

Maps that can be automatically constructed can be used for any application where manual map construction is too expensive. Special purpose maps for which, for example, one person collects all the traces for one campus can be used for company purposes in navigation applications. Many navigation applications are thinkable, including campus navigation for employees, routing for supply robots or process optimization by optimal route planning. Navigation is also an application for bike maps or maps for inline skates which can be constructed from bike and inline skate traces. Besides navigation, applications for planning are also important to look at. Automatically constructed maps can be used in planning, e.g., when choosing the location for a store for inline skates or a café. Also, for city or traffic planners these maps might be useful to detect weaknesses in the traffic system, e.g., when average speed on a certain connection road drops for bicycles.

Use of Additional Information

Based on GPS traces additional information can be added to existing maps. Additional information can be the:

- average velocity,
- average velocity for separate transportation modes, e.g., with a bike 15 km/h,
- usage split per transportation mode, e.g., this road has a usage share of cars of 50%,
- usage of transportation modes, e.g., if bikes use this road at all,
- velocity variation. This might be a quality indicator for a road. Roads with low velocity variation should be preferred.

The additional information can be used in applications. Based on this information, more demand-oriented routing is possible. It is also important for city planning to detect weaknesses in the road network and possibilities to reroute traffic to less occupied streets. Additionally, this information can be used by trip planners, who are creating a trip based on this information, e.g., one is able to spot routes which are especially suitable for bikes.

Creation and Update of Maps

Existing maps may need updates due to road changes. Road changes not only influence the course of the road, but also properties, like the aforementioned additional information, of the road. For example, a change of traffic light behavior does not change the course of the road, but it can change average speeds or velocity variation due to better traffic flow. Additionally, a new company in a certain area can influence the occupancy of nearby roads which would also affect road properties. The preferred way to detect these changes is an automatic processing of GPS traces, which can be done in a cost-effective and timely manner.

Another important aspect is the creation of maps which would not be created because there is not enough financial interest to create them manually. This is of concern for rural areas where map creation is only useful for a few people and these people are not willing to pay particularly for the coverage of the missing area. Another important idea is to create specialized sports maps for different transportation modes, e.g., an inline skating map. In comparison to the bike community, inline skating is interesting for fewer people, so there does not exist a good coverage of maps for inline skating. GPS traces can be automatically

filtered for transportation mode and a road map can be constructed only taking the filtered traces into account.

Privacy

Privacy can also be seen as a demand of individuals. Obfuscation methods which ensure a certain level of privacy can increase the willingness to contribute trajectory data [13]. Also, monetary incentives increase the willingness to contribute trajectory data [13, 19, 2]. Privacy is a strong concern in modern societies and an important personal right that people are willing to protest for. Conclusively, ensuring privacy can replace monetary incentives and create higher acceptance for contribution.

1.3 Definitions

This section covers essential definitions regarding the objects used in this work. The objects in this work are mainly data gathered from mobile positional data and spatial objects to represent these data. Please note that the definitions are not generic and might only be valid in the context of this work.

- *Point*: A point is a unique location in the Euclidean space, described by an ordered set of numbers. In this work we will only need two-dimensional points which are described by an ordered pair of numbers:[1]
  $p \coloneqq (x, y), x \in \mathbb{R}, y \in \mathbb{R}$.

- *Coordinate*: A coordinate is a unique location in WGS84[57] (World Geodetic System Revision 84) described by an ordered triple with latitude, longitude and altitude:
  $c \coloneqq (lat, lon, alt), lat \in \mathbb{R}, lon \in \mathbb{R}, alt \in \mathbb{R}$
  or, as in this thesis, an ordered pair with latitude and longitude:
  $c \coloneqq (lat, lon), lat \in \mathbb{R}, lon \in \mathbb{R}$.

- *Coordinate Reference System*: the coordinate reference system used for this work is the WGS84[57] which defines the a reference ellipsoid and uses the EGM96 geoid[44].

- *Trajectory*: A trajectory is the path that a moving object follows through space and time. As a more general term a trajectory can be described by a function or a log. We define a trajectory as an ordered sequence of points:
  $t \coloneqq (p_1, p_2, ..., p_n), n \in \mathbb{N}$.

[1] We use the () notation to represent ordered sets, as it is common in mathematics.

- *Subtrajectory*: A subtrajectory is a part of an original trajectory which was splitted for a certain purpose, e.g., matching of trajectories:
  $s \coloneqq (p_1, p_2, ..., p_n), n \in \mathbb{N}, s \subset t$.

- *Trace*: A trace or GPS trace is a more general description of a point or coordinate log over time.

- *Track*: A track or GPS track is an ordered set of coordinates over time. In this work the term track is mainly used in line with the GPX (GPS eXchange) format.

- *Node*: A node is a point or coordinate which represents a connection between edges or the start or final point or coordinate of a trace. A node has one or more connections. We can define it as an ordered set:
  $n \coloneqq (p, c, e_1, e_2, ..., e_n), n \in \mathbb{N}$.

- *Edge*: An edge is a connection between nodes or formally an ordered pair of nodes:
  $e \coloneqq (n_1, n_2)$.

- *Angle*: An angle is a synonym for the term course in navigation. Since in this work the context is not navigation we avoid the term course and substitute it with angle. An angle is measured in degrees, ranging from 0°to 360°where 0°is north, 90°is east, 180°is south and 270°is west. An angle is a (derived) property of an edge.

- *Slope*: A slope describes the direction and steepness of a line. In two-dimensional Euclidean space it is calculated by
  $m = \frac{y_2 - y_1}{x_2 - x_1}$.

- *Microdata*: Microdata is data at the level of individuals. For example for patient records it is the detail level per patient. For navigation systems it is, for example, one trajectory recorded by an individual.

For performance analysis it is important to know about the hardware of the test system. If not specified in the test directly, all performance tests were executed on a terminal server with 4 cores (2 x Intel(R) Xeon(R) CPU X5450 @ 3.00GHz) and 16 GB RAM. Since, it is a multi user system, tests were repeated to exclude load variance due to high usage by other users.

If the complexity of an algorithm is stated, the computational model Real RAM [65] is assumed. In Real RAM, arithmetic operations, comparisons, and trigonometric functions are available at unit cost.

1.4 Organization

The rest of this thesis is organized as follows: Chapter 2 discusses the related work and the implemented aggregation process. Chapter 3 discusses the matching methods and Chapter 4 the merging methods which are used for the aggregation process. Means for privacy preservation are discussed in Chapter 5. Chapter 6 shows the results of the evaluation of the proposed methods. The work is concluded in Chapter 7.

Chapter 2

Background

The background for this thesis concerns the related work and the conceptual design. The related work includes a differentiation of this work from other works, but also related work which was used within the extent of this thesis as a part or a module. Another important background for this thesis is the conceptual view for the components which were developed for this thesis.

This chapter is organized as follows: Section 2.1 describes realated work on trajectory anonymization, map construction, and applied methods. Section 2.2 reports on new work which was carried out within the extent of this thesis, including an achitectural and process view and additional measures taken into account.

2.1 Related Work

Relevant work can be divided trajectory anonymization, map construction, and applied methods.

2.1.1 Work on Trajectory Anonymization

[58] describes an aggregation approach to anonymize trajectory data based on k-anonymity. The approach includes basically 3 steps: a grouping of trajectory, the anonymization (basically finding a trajectory's representation) and a randomized reconstruction. The approach is straight-forward and efficient. For map construction purposes this approach would need an extension on how to split trajectories beforehand to represent streets or parts of streets, but not complete trajectories. Also, it remains unclear how trajectories with different time or distance intervals are handled.

[54] has a similar approach as in this work. Basically both approaches want

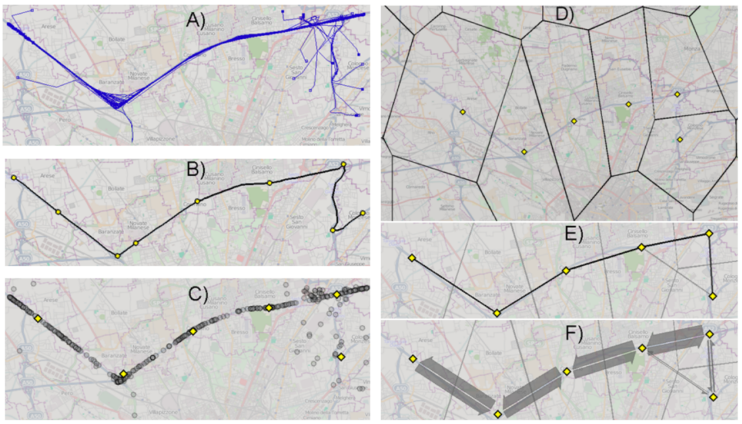


Figure 2.1: Trajectory generalization by Voronoi tessellation [54]

to achieve anonymity of GPS traces by generalization. Nevertheless, [54] has a different application in mind. More important for the clustering later on are here start and destination of a trajectory. Subtrajectories themselves are not considered, but are important for map construction, which this work has as its application. Figure 2.1 shows the process of the trajectory generalization. (A) shows the set of trajectories which should be generalized. First, for each trajectory characteristic points are extracted. This is shown for a single trajectory (B) and (C) shows the characteristic points for all trajectories. These characteristic points are clustered. The cluster centers are shown as yellow rhombi in (C) and (D). Based on these cluster centers a Voronoi tessellation is done. And based on this tessellation the generalized traces are represented as movements within these Voronoi areas. You can see that the focus are movements from one area to another. Nevertheless, in this work the focus are the movements on a more fine-grained level which can represent the characteristics of streets.

[8] discusses several possibilities to anonymize trajectory data. One anonymization it discusses is also based on generalization. Figure 2.2 shows the generalization approach via point matching. Points are matched to points on other trajectories. Around the matched points a bounding box is created. This bounding box is being stored and the individual trajectories are deleted. After that, the paper shows a reconstruction approach to generate synthetic individual traces based on these bounding boxes. Nevertheless, the approach doesn't cover a step to split trajectories and create subtrajectories which have similar properties. Avoiding this step it is most likely that the result of applying it to a whole set of trajectories covering a road network would not lead to a satisfying result.

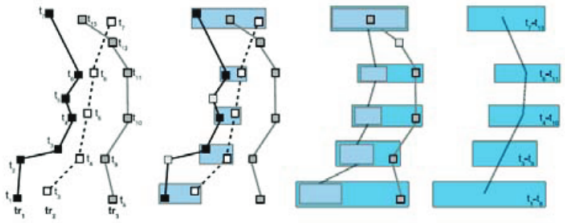


Figure 2.2: Trajectory generalization via point matching [8]

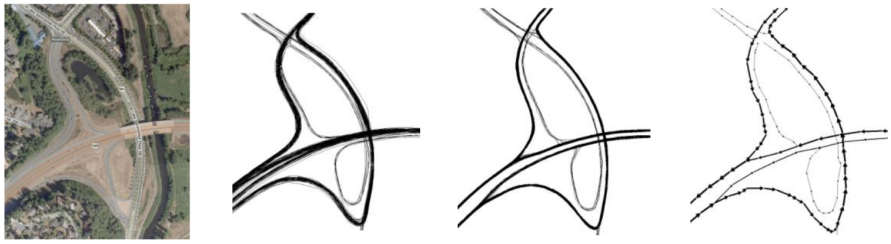


Figure 2.3: Road map construction from trajectories [17]

2.1.2 Work on Map Construction

[17] shows how to create a routable road map from a set of trajectories. Simulated potential energy is used to group trajectories together. Figure 2.3 depicts a satellite image of an area, the belonging raw GPS traces, the result after clarification, and the result after road map creation. The main steps are clarification and road map creation. In the clarification step, attraction forces are used to move single nodes of a GPS trace. The nodes are moved towards other points on traces which are calculated via an orthogonal cut with nearer points having a higher attraction force. The result shows traces which were pulled to each other. The direction of the traces was taken into account. Hence, traces in opposite directions had a negative attraction force. This is the reason that one road is usually represented as two thick lines of GPS traces after the clarification step. In the graph generation step the traces are considered iteratively. The graph first includes the first trace and after that the second trace is merged with the graph. After merging all traces into the graph, refinement is considered, e.g., at crossroads.

2.1.3 Applied Methods

This section is about methods which are used within this work or related work of the methods applied, e.g., a newer improvement or an application example. The applied methods can be categorized in distance measures, methods for subtrajectory clustering (often applying distance measures), and anonymity

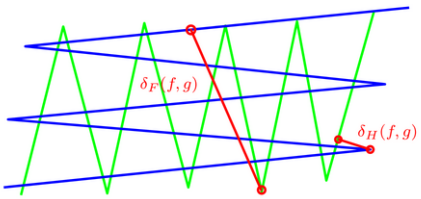


Figure 2.4: Comparing Hausdorff distance with Fréchet distance [67]

measures.

Distance Measures

For spatial objects, in particular, points, the Euclidean distance[21] is considered. In a two-dimensional metric system the calculation is $d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2}$. Nevertheless, to calculate a metric distance in Geodesic space, e.g., using the WGS84[57] ellipsoid, you need to take into account the shape of the reference ellipsoid. One way to calculate the distance is shown in [76]. This method is also used in the Google API for Android[73]. The direct distance can be useful to find similar trajectories based on point-to-point or edge-to-edge comparisons. Nevertheless, for trajectories exist special measures which can cope with the fact that a trajectory is not limited in the number of points which describe the trajectory. The Hausdorff distance[78] which is defined over sets can be applied for geometric objects and also trajectories. For trajectories it will search for every point on one trajectory the closest point on the other trajectory and vice versa. The upper bound then is the Hausdorff distance. Nevertheless, for trajectories which are near together but follow completely other directions and do not resemble each other, the Hausdorff distance seems too close. For that reason, when comparing trajectories, the Fréchet distance[1] is often used. Figure 2.4 displays two curves (trajectories) which are used for the illustration of the difference between the Hausdorff distance and the Fréchet distance. While for parallel curves the Hausdorff distance and the Fréchet distance have the same value, for curves like the ones depicted in Figure 2.4 the difference is high. To cope with geodesic data, [40] presents a calculation of the Fréchet distance in logarithmic time with geodesic distances.

Subtrajectory Clustering

[75] contains definitions of trajectory similarity and their computation. It takes into account a difference in time of the trajectories. Distance measures like the Fréchet distance have no time component. A time component might be helpful

to filter according to travel times or speed difference. Nevertheless, I suppose in my work it has no direct use and a direct filtering according to time might be better placed as a preprocessing step than in the actual comparison of GPS traces. The three case distinctions (fixed duration and no time shift, non-fixed duration and no time shift, and fixed duration with time shift) are reasonable, but I wonder about which application might fall in which case, except for the hurricane scenario. Altogether, the paper focuses more on computational complexity than on applications.

[35] proposes a clustering approach for trajectories. A major difference to my approach is the classification of trajectories in direction categories like *short right turn* or *long straight segment*. The following steps are based on this classification and therefore not directly comparable. The next step is motif discovery which asks for often occurring substrings where the direction categories are represented as characters.

In [43] a method for subtrajectory clustering is proposed. It is called TRACLUS and uses a minimum description length (MDL) principle. It is designed to find common sub-trajectories which is also what I try to accomplish. The used distance function consists also of multiple elements. These are namely 3 distances: the perpendicular distance, the parallel distance and the angle distance. Figure 2.5 shows the three distances comparing two edges. The first phase of the TRACLUS algorithm is to find characteristic points. Thereby, some preciseness is traded for conciseness, by weighing according to the MDL principle. After that line segments (or edges, not a complete trajectory) are clustered according to their distance function with DBSCAN. The representative trajectory is found by a sweeping algorithm which sweeps according to the main direction and sets average points each time the number of sweep crossings changes and MinLns is met. Figure 2.6 shows an illustration of the sweeping algorithm. This algorithm can also be used for map construction even though the evaluation was done with hurricane data. Maybe only parts of the algorithm like the distance function or the sweeping algorithm will turn out to be useful. There is a python implementation available at `http://code.google.com/p/zhouyunlib/source/browse/trunk/UOM/VR-Preprocess/src/Traclus.py?r=266`. The functionality is also used in MoveMine[47] which is a Moving Object Mining Tool. A demo is available at `http://dm.cs.uiuc.edu/movemine/`. An implementation is available at `http://web.engr.illinois.edu/~klei2/movemine/`. [46] gives a more comprehensive overview of MoveMine.

[14] is also a work which describes a subtrajectory clustering to detect commuting patterns. As distance the Fréchet distance is used. The main contribution of the work is an optimized approach in using the Fréchet distance for subtrajectory clustering. Proofs of hardness and simplification are shown, which

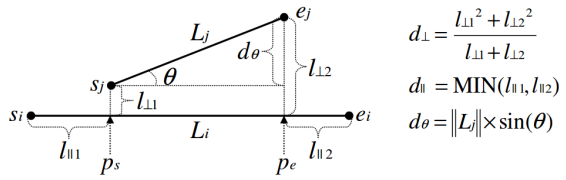


Figure 2.5: TRACLUS distance function elements [43]

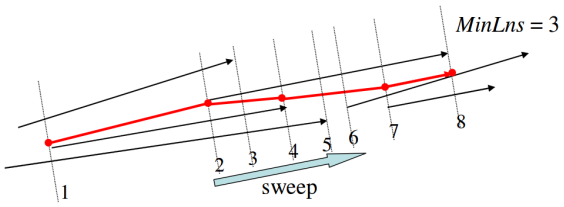


Figure 2.6: TRACLUS sweep function for representative trajectories [43]

give a strong insight in possible calculation efforts.

[80] describes an adaptive clipping algorithm based on the Fréchet distance to match curves which addresses just the matching phase and has to be extended for clustering or merging purposes. Nevertheless, it also provides deep analysis of performance improvements in matching algorithms.

[83] proposes a subtrajectory clustering based on meshing grids and the Fréchet distance. The method uses a fixed grid structures which is defined in advance. A hierarchical clustering is applied. After the clustering, an inter-grid concatenation is done.

Another preprocessing step that might be promising for this work is trajectory calibration [69]. It can enhance the data quality of trajectories and might therefore used as a preprocessing step.

Anonymity Measures

In microdata anonymization k-Anonymity[72] is often displayed as introductory example of anonymity measures. The most exhaustive and fundamental research in microdata anonymization is about relational data though it is not limited to relational data[56]. In relational data microdata anonymization looks at attribute which can be categorized into quasi-identifiers and sensitive attributes. Quasi-identifiers can be used in combination to identify a person and sensitive attributes are the attributes which should not be linkable to a person. The approach of k-Anonymity is to generalize within the quasi-identifiers until there are at least k entities of a unique combination of quasi-identifiers. An often referred example are patient records. Table 2.1 shows a simplified example of

age	gender	postal code	disease
41	male	10178	lung cancer
73	female	10117	arthritis
23	male	10245	Pfeiffer's disease
21	male	10243	influenza
30	female	13357	breast cancer
39	female	10243	breast cancer

Table 2.1: Patient records, raw data

age	gender	postal code	disease
40-79	*	101**	lung cancer
40-79	*	101**	arthritis
20-29	male	102**	Pfeiffer's disease
20-29	male	102**	influenza
30-39	*	1****	breast cancer
30-39	*	1****	breast cancer

Table 2.2: Patient records, 2-anonymized data

patient records. The attributes age, gender, and postal code are quasi identifiers. In this example already the age or the postal code are sufficient to identify a person. Nevertheless, in a real world example those quasi-identifiers can be used in combination for identification. The sensitive attribute in this example is the disease. For illustration we choose $k = 2$ as privacy requirement. Next, theory provides several algorithms to generalize up to the security requirement[70, 71, 4]. In our example it is easy to create 2-anonymized data manually. In table 2.2 the raw data was anonymized. The different ages were grouped together in age ranges, the gender was either kept or completely removed and the digits of the postal code were partly suppressed. Now, it should be impossible to directly link one person to a disease. Nevertheless, this example was also chosen to show one shortcoming of the k-anonymity approach. The problem with this data is the distribution of sensitive attributes within one block of the same quasi-identifiers. In this example you can link any 30-39 year old female in Berlin (1****) to breast cancer if you know she is part of the data set because there is no distribution at all in the sensitive attributes block. Therefore, extension to k-anonymity evolved which also take the distribution of sensitive attributes into account, namely l-diversity[50] and t-closeness[45] for example. Nevertheless, for trajectory anonymization these extensions are not considered because the trajectories are categorized as quasi-identifiers without a sensitive attribute.

Another approach or measure is differential privacy[24]. The method to

anonymize here is randomization. Differential privacy is a guarantee of the strength of the anonymization by randomization. The randomization is chosen as strong as it is necessary to fulfill a certain differential privacy level, e.g., using ε-differential privacy, ε is the guarantee level. A certain ε ensures a certain level of indistinguishability whether a data set is generated with or without a certain person in it. The claim is that it shouldn't be able to distinguish between a data set where one individual participated or not. To randomize a LaPlace distribution is used because it can create a more randomized outcome in the same range than , e.g., a Gaussian distribution can. Using differential privacy for trajectory anonymization might be promising in some applications [18]. In particular, if the trajectory data can be represented as a tree or a graph, this can be a strong approach.

Transportation Mode Detection

In this work transportation mode detection is used as a method to categorize GPS traces. Transportation mode detection is mainly based on information retrieved on smartphones. The main source of information is the GPS receiver. Based on these data important features like velocity, acceleration, but also GPS error are calculated. Works in transportation mode detection can be categorized via the transportation modes they take into account. There are works concentrating on basic mobility states, like standing still, walking, bicycle riding, and motorized travel [62]. Other publications extend this classification with several motorized transportation modes, e.g., car, bus, train, and underground [68, 7]. The feature set also varies between different works. A standard feature set contains the velocity and acceleration [62]. [7] uses additionally features of time difference between points and the distance between the available points which can be derived from the same raw data. GIS data like the average distance to bus stops can also be used as background information which improves the overall precision[68]. Furthermore, the related research can can be distinguished according to the classifiers used. [68] and [62] use multiple classifiers, including Naive Bayes from WEKA[38] as their classification algorithms. [7] uses Support Vector Machines which is also used in [68]. Apart from these, other promising techniques for identification of transportation modes also exist [74]. For the purpose of this work an approach which is easy to implement and does not need very accurate data from multiple sensors is preferred. Nevertheless, for this work background information about bus stops and more is available. The implementation in this work is comparable to the one in [68].

2.2 Aggregation Process

Following a K-anonymity[72] paradigm, a major issue is the aggregation of trajectory data. The aggregation can be divided into a matching phase and a merging phase. Trajectories are matched to belong together and then merged into a representative trajectory. In the literature this is also discussed as sub-trajectory clustering.

An important aspect of the aggregation is the architecture which describes the communication between the participating stakeholders. Important issues are the type of distribution of data and responsibilities, the classification in trustworthy and non-trustworthy actors, and the organization of this distribution. The data to be stored is the aggregation of the GPS traces which were contributed in the past. This data can be distributed among self-coordinating partners or centrally organized. The main task to fulfill is the integration of new traces to the existing aggregation, respectively the update of the aggregation. This task can be done on a client or on a trustworthy actor in the network. The distribution of tasks is influenced by the selection of which actors are or are not trustworthy. This is important to choose beforehand because it influences the design of the network in a high degree. If the data is distributed there needs to be a concept of organization which defines how to retrieve and process the data of concern.

2.2.1 Network Architecture: Client-Server and Peer to Peer

From the communication perspective there are two different approaches to distribute data and responsibilities, namely a client-server approach and a peer-to-peer approach. These are two different approaches for distribution, a centralized and a decentralized approach. It is important that peer-to-peer networks are more prone to attacks because of their openness and autonomous nature [79]. It is more likely that peers are regarded as non-trustworthy participants than a central server. Nevertheless, the amount of data that can be misused is higher in a centralized approach. In a peer-to-peer network methods to organize reputation and trust can be implemented [79]. Overall, the decision for a specific network structure is highly influenced by the assumptions of the participants.

Figure 2.7 displays the network structure in a client-server approach. The server is the centralization point and participates in all network traffic. It serves a central trajectory aggregation and is the only participant which makes direct changes to this data set. The process of adding an individual trajectory to the aggregation can include the following exemplary steps:

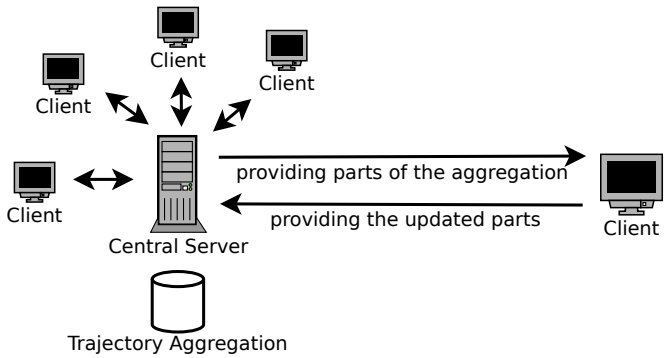


Figure 2.7: Client-Server Network Structure

- The client asks the server for a certain part of the aggregation which it wishes to add the individual trajectory to.
- The server provides the requested part.
- The client adds he trajectory to the aggregation. In this step the client first matches the trajectory to the aggregation, decides within privacy restrictions which trajectories or subtrajectories to add, and finally adds them to the aggregation (merging).
- The client provides the modified parts to the server.
- The server checks for plausibility and if possible for reputation and trust of the client and updates the aggregation based on the received modified parts.

Generally, the client-server approach causes less network traffic and is easier to maintain. Nevertheless, the central server has to deal with high load and is the bottleneck of the system. This is another reason why the calculation of the new aggregation should be on the client side. A central server can also be replaced by a distributed system which can overcome shortcomings. A natural distribution for this system can be geographical one [23].

Figure 2.8 depicts the network structure in a peer-to-peer approach. Multiple peers replace the central server. The detailed communication is not depicted here. Nevertheless, this should be the same as in the client-server approach. A peer-to-peer approach has more aspects to take into account. If the data (aggregation) is not strictly separated, e.g. geographically, then a revision control and a data exchange between peers is needed. Nevertheless, if the data is strictly separated an important advantage of a peer-to-peer network is lost, namely the fault-tolerance against the breakdown of a single peer. Therefore, there is a strong advantage which comes along using revision control and data

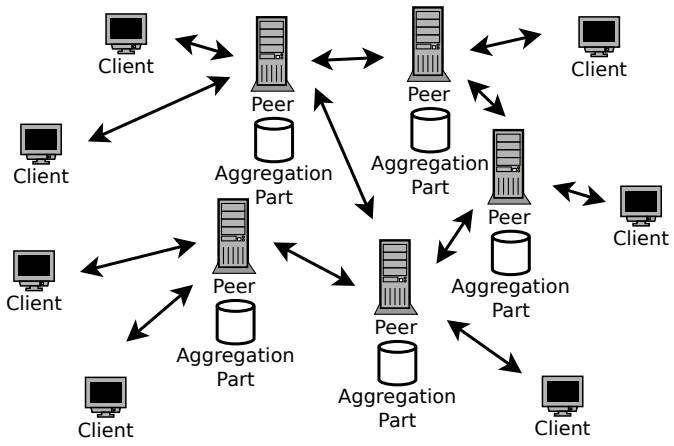


Figure 2.8: Peer-To-Peer Network Structure

exchange between peers. Nevertheless, these considerations are not application specific and thus there are already solutions proposed and implemented, e.g., Mercurial[51], Git[29], BitKeeper[6] and Bazaar[16].

2.2.2 Iterative Process

One important design decision for this work is that the collection of GPS traces is iterative. This means that the aggregation of the trajectories has to be adjusted every time a new single trajectory is added to the collection. This is necessary because the aggregation should always be able to represent the current number of trajectories which were added to a specific node or subtrajectory. Additionally, it should always be possible to anonymize an individual trajectory via the generalization process of adding the trajectory to the aggregated trajectories. This implies that depending on the aggregation method the adding sequence can influence the result. It is important to distinguish between algorithms which can ensure the same result independent from the adding sequence and algorithms which do not ensure the same result.

From the architectural point of view, having an iterative process means that the calculation (aggregation) has to be done as a transaction in real-time and cannot be processed in batch mode which is possible if the data (trajectories) can be collected beforehand.

2.2.3 Privacy Threats

Privacy threats can evolve through non-trustworthy participants in the network. In a client-server network the server might reveal information about clients which are not supposed to be revealed. Generally, a direct link to a client

should be avoided. Clients or users in the network can be identified by their IP (Internet Protocol) address. This fact can be misused by a server to combine information from different requests and merge data which the client intentionally separated to preserve privacy. Another measure to combine different requests is the possibility to match geographically close data. If the server receives requests which can be merged usefully from only one certain area then there is an assumption with high probability that all requests are from one client. Additionally, security measures for trust and reputation can be misused for unique identification.

2.2.4 Privacy Measures

It can be assumed that anonymous access is possible to a certain degree, e.g., with onion routing[22]. The main concern is the identification via the contributed trajectories. To ensure privacy according to the user's preference, it has to be decided which trajectories or subtrajectories will be added to the global trajectory aggregation. Seen from a networking privacy perspective, this decision should be made on the client side. Otherwise, many attacks are possible if the server is not trustworthy. Furthermore, it is preferable to send changes based on individual trajectories distributed over multiple requests, to avoid re-sampling of the complete trajectories. These multiple requests should be randomized in order and time. Following these recommendations does not ensure absolute privacy if the server is not trustworthy, but these recommendations are necessary to be able to ensure privacy if other circumstances cannot be exploited by the server. An example might be that someone is the only contributor and it does not matter is he or she sends multiple randomized requests, the server will still be able to reconstruct the original trajectory. Nevertheless, if there is a critical mass of users and they share common area in their trajectories, a simple reconstruction will fail.

2.2.5 Security Measures

Besides secure Internet connection also the data provider needs protection from attacks from clients. The most important case to look at from the perspective of this work is the protection against receiving a critical amount of wrong or synthetic trajectories which do not represent natural movements. In areas where already a certain amount of data exists, implausible trajectories might be able to detect and can be used to create a system of reputation. This can be done using anonymous credentials [49] which can also be used in a decentralized network [28].

2.2.6 Data Cleaning

Data cleaning can be useful for GPS trajectories to prevent mistakes in the following aggregation of the trajectories. When recording GPS trajectories, effects can occur which can have an unwanted effect on the aggregation of the data. The GPS signal has different qualities depending on multiple factors. For example, one has to expect a lower quality inside a car than on top of the car and in an area with many mountains the signal should also be worse than on a flat landscape. This variation of the signal quality can induce errors. A demonstrative example is a car stopping at a traffic light. Before stopping the movement can be detected well because the GPS imprecision was marginal compared to the movement. Nevertheless, when the car stands still the imprecision is not marginal and determines completely the movement of the recorded trajectory. Due to the GPS error the recorded movement has no relation to the street network the car is on and includes any direction. There a multiple measures which can be taken to filter out GPS errors and ease the aggregation process.

One measure is to filter for a heading change. A strong heading change can, for example, be induced by the aforementioned stop at a traffic light. When filtering for the heading change one has to set an upper limit for the difference between the angle of the edge before and the angle of the edge to be filtered. If the difference is too high then the current point (end point of the current edge) will be omitted and only the next point will be added. This filter can help with GPS imprecision while standing still. Nevertheless, one has to be aware that pedestrians can actually have a fast heading change. The filter should be used if transportation modes with a high possible heading change can be excluded.

Another measure is to normalize the length of edges. To normalize the length of edges one can set a lower and an upper value for the length of an edge. If the current point creates an edge which is shorter than the lower value then the point is omitted. If the current point creates an edge which is longer than the upper value then there are inserted as many points in-between as the factor by which the edge is longer (minus 1). This measure can also help with the imprecision of GPS. Furthermore, the calculation time can be reduced if there are a many points near together which have no significance for a road network. Nevertheless, also the calculation time can be increased when there are long episodes of steady driving like on a highway. Another important issue is that a normalization of the edge length can help to adjust to a fixed distance threshold. Depending on the used distance the distance can vary according to the length of the edges within the trajectory. This is in most cases an unwanted effect and can be handled either in adjusting the distance or in a normalization step beforehand.

2.2.7 Functional Architecture

In this section the functional architecture of the Agg2Graph software is introduced. Agg2Graph is a prototypical implementation of methods for trajectory aggregation with additional functionality for data cleaning and road map construction. The software is organized in process steps: import, cleaning, aggregation and road network generation.

Figure 2.9 shows the functional architecture of Agg2Graph. The main process can be controlled via a command line interface or a graphical user interface. The functions of the command line interface can additionally be automated via test configuration files. In the import step the GPX files are read and cached as trajectories in main memory. In the cleaning step cleaning methods are applied to the trajectories. The cleaning methods can be distinguished as normalization and simplification. All the methods described in Section 2.2.6 can be used. However, this is not a necessary step. The aggregation can be either based on the imported or the cleaned trajectories. If the cleaning step was used then the aggregation is based on the cleaned data. The aggregation has two man components: the matching component and the merging component. The distance is a subcomponent of the matching component. These components can be exchanged, e.g., the Fréchet distance can be replaced by the Hausdorff distance. However, this is limited to components of the same type and one has to distinguish between components for points, edges, and subtrajectories. The result of the aggregation step is one trajectory aggregation. Based on this trajectory aggregation the road network construction step can produce an OpenStreetMap file which describes the road network.

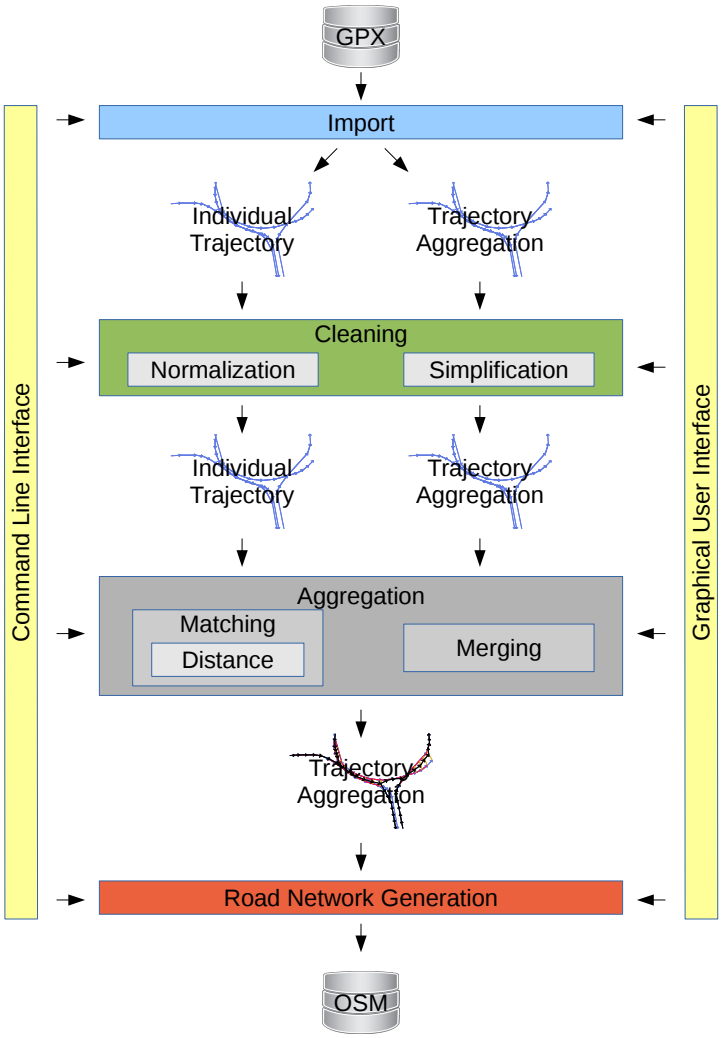


Figure 2.9: Functional Architecture of Agg2Graph

Chapter 3

Matching of Trajectories

This chapter gives an overview of different methods to match trajectories. Matching trajectories is the prerequisite of merging trajectories. Before trajectories are merged, it has to be ensured that they match together. The matching can also include a splitting of trajectories into subtrajectories. We will first take a look at distances which can express a match, e.g., fulfillment of a certain distance. Next, matching and splitting procedures are on display. Conclusively, the last part is the global view on the matching process.

This chapter is organized as follows: Section 3.1 describes point-based distances, Section 3.2 edge-based distances, and Section 3.3 trajectory-based distances. The extension with an angle component is described in Section 3.4. Section 3.5 discusses an iterative matching approach and Section 3.6 discusses the global matching perspective. Section 3.7 concludes this chapter.

3.1 Point-Based Distances

Point-based Distances compare two points to each other. The most important geometric distance is the Euclidean Distance. The Euclidean Distance can be calculated for multiple dimensions. Relevant for this work is the two-dimensional calculation:
$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$

The Euclidean distance can be calculated in linear time. For this work the execution time on standard hardware (see Section 1.3) was measured. The measurement is based on 5 trajectories. Each point on every trajectory was compared to every other point of every other trajectory. In total the execution time for 184212 calculations was 178 ms. In geometry there are many more distances which cover specific aspects, such as the Manhattan distance, the Chebyshev distance, the Mahalanobis distance, the Minkowski distance and more. Since

we are interested in spatial distances, all these distances seem less appropriate than the Euclidean distance. For example the Manhattan distance seems useful when streets are organized block-wise and the direction of the streets are represented by the axes represent the direction of the streets. Nevertheless, we are not able to ensure that we can take advantage of the specific advantage of the Manhattan distance. For the other distances the argument is the same: they all cover specific circumstances which cannot be ensured within this work. The execution time for the Manhattan distance was measured under the same conditions as the Euclidean distance with 87 ms, which is a plausible result.

To calculate a metric distance in with coordinates (latitude, longitude) one has to take the earth ellipsoid into account. For the purpose of this work it is feasible to use the WGS84[57] ellipsoid via the Android Location[73] implementation which calculate the distance based on Vincenty's formulae[76]. By using a jar build provided by Robolectric[81] it is possible to directly access Android functionality in normal Java code. The execution time for the calculation (same conditions as for the other point based distances tests) was 3090 ms which is more than 17 times higher than for the Euclidean distance. If execution time is critical it might be feasible to refrain from using a precise spherical calculation and calculate distances only geometrically, ignoring the spherical property of the earth. This is particularly the case when we take a look at the matching of near trajectories which have distances of less than 100 meters. Another implementation[53] of Vincenty's formulae had an execution time of 2163 ms. Applying the Euclidean Distance to coordinates implies the conversion from coordinates in distances in meters. The distance between two circles of latitude is 111.3 km and the distance between two circles of longitude is $111.3 * cos(lat)$ km where lat is the angle which is in the middle between both latitude values. Taking this into account the execution time insignificantly increases to 230 ms which is less than $1frac13$ of the time the Euclidean distance itself can be calculated. The Haversine[66] distance approximates a great circle and has more precise results than the adjusted Euclidean distance, but less precise results than the calculation based on Vincenty's formulae[76] and the WGS84[57] ellipsoid. The execution time is 598 ms.

Table 3.1 summarizes the execution times of the evaluated point-based distances.

3.2 Edge-Based Distances

An edge of a trajectory includes much more information than a point or coordinate. Additionally to the two locations given by the points, an edge has information about direction, slope, distance, and if time is given, speed. This

distance	reference	execution time	factor
Manhattan Distance		87 ms	0.49
Euclidean Distance		178 ms	1
coordinate adjusted		230 ms	1.29
Haversine distance	[66]	598 ms	3.36
Vincenty's formulae	[53]	2163 ms	12.15
Vincenty's formulae	[73]	3090 ms	17.36

Table 3.1: Execution times of point-based distances

information can all be taken into account for an edge-based distance. The TRACLUS[43] distance function shown in Section 2.1.3 and Figure 2.5 is an example for an edge-based distance. This distance function was developed to cover important aspects when comparing two edges. Nevertheless, based on the application and the favorable outcome of a clustering or aggregation process, much more edge-based distance can be developed. Figure 3.1 depicts the most important aspects which can be taken into account for edge-based distances. To formalize we set the following:

- e_t is the edge in comparison of the single trajectory, depicted as *trace* in Figure 3.1
- e_a is the edge in comparison of the trajectory of the aggregation, depicted as *agg* in Figure 3.1
- p_{a1} is the first point of the edge of the aggregation
- p_{a2} is the second point of the edge of the aggregation
- p_{t1} is the first point of the edge of the single trajectory
- p_{t2} is the second point of the edge of the single trajectory
- x_{a1} is the crossing of the perpendicular of the first point of the edge of the aggregation with the edge of the single trajectory or its extension
- x_{a2} is the crossing of the perpendicular of the second point of the edge of the aggregation with the edge of the single trajectory or its extension
- x_{t1} is the crossing of the perpendicular of the first point of the edge of the single trajectory with the edge of the aggregation or its extension
- x_{t2} is the crossing of the perpendicular of the second point of the edge of the single trajectory with the edge of the aggregation or its extension

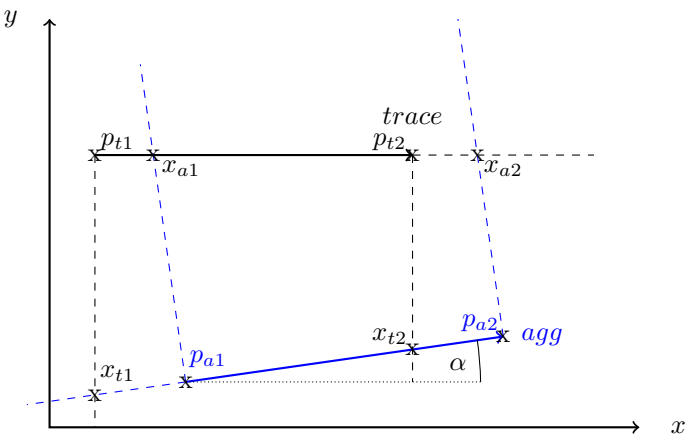
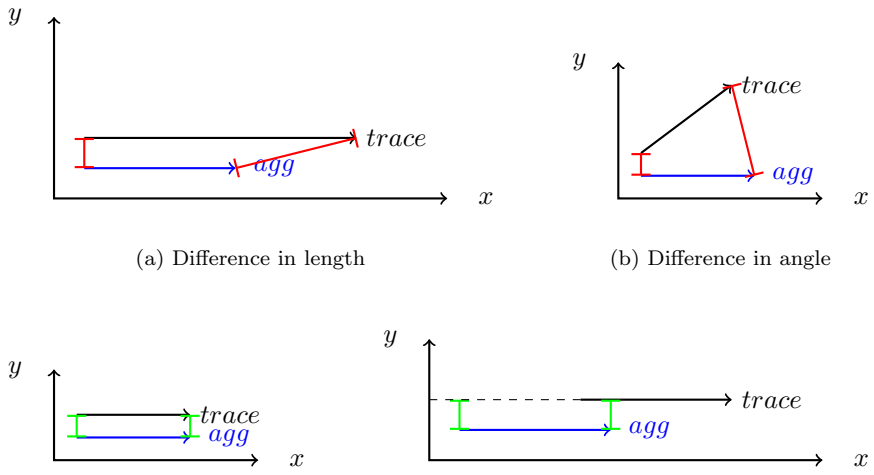


Figure 3.1: Two directed edges with extension and perpendiculars marking the crossings of perpendicular and edges or their extension including the angle between the two edges

- α is angle between the parallel edge of the single trajectory which is moved so that both first points have the same position and the edge of the aggregation or vice versa. It is given as absolute positive value and the smaller of the both possible values is chosen, e.g., if 350° and 10° are possible, 10° is chosen.

Figure 3.1 displays start and end points of an edge, their perpendiculars and the crossing point of the perpendicular and the other edge. Using the perpendiculars to measure the distance between the two edges is favorable to just measuring the distance between start and end points. Figure 3.2 shows the deficiency of just comparing the start and end points. The two cases depicted in Figure 3.2a and Figure 3.2b show the same measured distance if only the distance between start and end points is measured. Nevertheless, in Figure 3.2a the difference resulted out of a difference in length of the two edges which means a high probability that both edges were recorded following the same course of the road. Contrary, in Figure 3.2b the same measured distance is caused by a difference in the angle of the edges which means a significant probability that the edges were recorded following another course of the road, e.g., turning at a crossing. On the other hand, the two cases depicted in Figure 3.2c and Figure 3.2d display the same distance that was measured using the perpendicular and the crossings between the perpendicular and the other edge. In Figure 3.2c both edges are likely to follow the same course of the road, there is just a small gap between the two edges (in this case the distance measured with start and



(a) Difference in length

(b) Difference in angle

(c) Low difference in start points

(d) High difference in start points

Figure 3.2: Comparison between measuring the distance between start and end points or between points and crossings of the perpendicular[55]

end points is equal). Nevertheless, Figure 3.2d shows two edges which also follow the same, but have quite different start and end points because these are shifted. Conclusively, in this case, the distance via the perpendicular is the same as in the case where the edges are not shifted and the distance measured with start and end points is much higher which does not represent the favorable difference which should be small.

As we have seen, the distance measured via the crossings of the perpendiculars is more significant for similarity measurement than the distance measured directly between both start and end points. There are in total 4 distances that can be measured via the crossings of the perpendiculars. If the two edges to be compared are parallel, all these 4 distances are the same. If these are not the same, they can point out a difference in the angle of the two edges. Nevertheless, this difference has to be set in relation to the length of the edges, and furthermore if the edges point in opposite directions then the difference of the 4 distances shrinks again and the distances are the same if they point to opposite directions and are parallel. As final distance the higher one of the measured distances should be taken for comparison because it includes the highest difference and a high difference at one point is already enough to exclude edges from matching.

The execution time for the distances using the crossings of the perpendiculars was measured for this work. The test environment was the same as for the point-based distances (see Section 1.3 and Section 3.1). The test data set is also the

same as for the point-based distances. Nevertheless, within each GPS trace file the first point cannot be used because there is no point pair for comparison. This means that instead of 184212 comparisons for the edge-based test there were 66528 comparisons. The execution time for all comparisons was 413 ms. Setting this in relation with the lower amount of comparisons this means a factor of 6.4. This factor is reasonable since the calculation includes 4 calculations of the Euclidean distance plus 4 calculations of slopes plus 4 calculations of interceptions of lines.

Nevertheless, including only the distances of the crossings via the perpendicular can also lead to non-significant matches when edges are parallel but have a high shift to each other. Figure 3.3 depicts two edges which are parallel and have a short distance when the distance is calculated via the crossings of the perpendiculars. This is a realistic scenario on straight lanes. The problem is that there is a high probability that a better match with approximately the same distance exists. Comparing this scenario to the edges depicted in Figure 3.2c and Figure 3.2d the probability for a better match with the same distance increases the higher the shift is. If the shift is half the length of one edge, then the probability increases rapidly. The solution is to include the shift, but to weigh it according to its length relative to the length of the edge. For each crossing a shift can be calculated. To reduce complexity only the shift for the crossing with the highest distance can be calculated. The following formula combines the previously calculated distance with the shift:

$$d_w = d_h + w \times (\tfrac{2 \times s}{l})^4$$

where d_w is the weighted distance, d_h is the highest distance that was measured beforehand via the crossing of the perpendiculars, w is the weight, s is the shift of the crossing which was used to calculate d_h and l the length of the edge which was used to calculate d_h and s.

The complexity does not increase significantly when the shift is combined with the distance via the crossings of the perpendiculars. The execution time for 66528 calculations is 471 ms which is a factor of 1.14 compared to the calculation without the shift.

3.3 Trajectory-Based Distances

Trajectory-based distances measure distances between trajectories. This means, contrary to the point-based and edge-based distances, that the comparison is based on multiple points or sets of points. Furthermore, this influences the complexity of the calculation which is based on the number of points in the sets.

Informally, the Hausdorff distance or Hausdorff metric is the highest distance

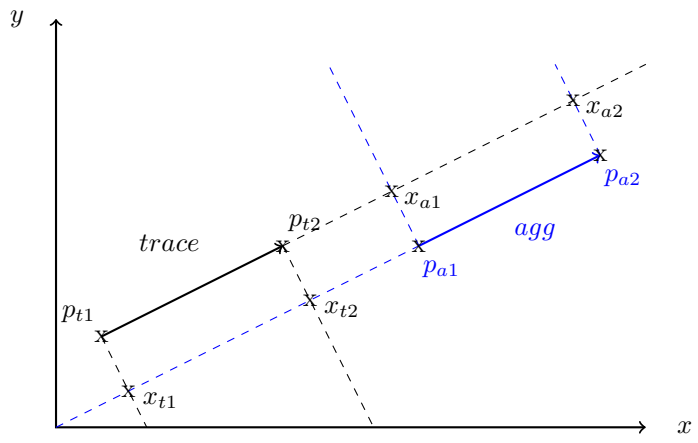


Figure 3.3: Two parallel edges with extension and perpendiculars marking the crossings of perpendicular and edges or their extension with a high shift to each other

from a set to another set where the distance of one element to the other set is the distance to the nearest element of this set. It is helpful to first define the distance between a point and a set of points (the lower bound):

$D(x, K) := min\{d(x, k)|k \in K\}$

where x is an element, K is a set and $d(x, k$ is a distance function which for the purpose of this work is a point-based distance (any distance, including edge-based distances, is possible). Next, we can define the Hausdorff distance (the upper bound of all the possible before mentioned distances):

$\delta(A, B) := max\{max\{D(a, B)|a \in A\}, max\{D(b, A)|b \in B\}\}$

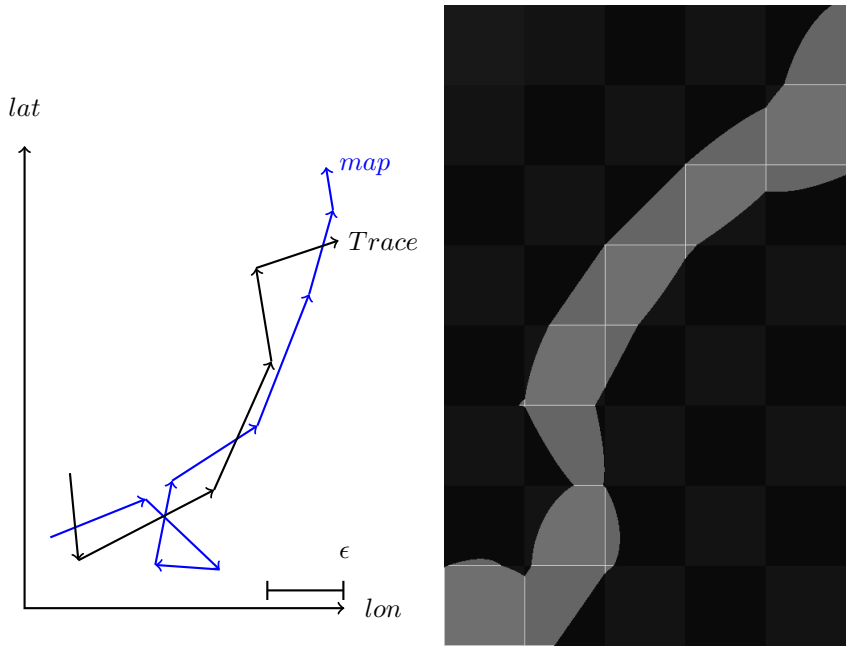
where A and B are the input sets for the calculation and a and b points within the particular set. The complexity of a simple calculation of this formula (requiring the distance between each point of the set with each point of the other set) is $O(n(A) \times n(B))$ where $n(A)$ is the number of elements in set A and $n(B)$ is the number of elements in set B. The test environment for the trajectory-based distances is the same as for the point- and edge-based distances, but for trajectory-based distances the number of points in the trajectory varies. For the purpose of this work 2 scenarios were used for runtime testing. The first scenario ensured calculations with trajectories with the size of at most 5 points while the second scenario was open and the size of points varies within the size of each trajectory. For the first scenario there were 17550 calculations possible with an execution time of 159 ms. For the second scenario 184212 calculations were possible with an execution time of 1728 ms. 5 trajectories with 34,2,9,43

and 7 points were evaluated. In the first scenario sizes do not vary, they were 5,2,5,5 and 5 and in the second scenario sizes vary between 1-34,1-2,1-9,1-43 and 1-7. Since the complexity of one distance measurement varies with the number of points of the trajectory, the most expensive distance is between point size 34 and point size 43 which includes 1462 calculations of the point-based distance (Euclidean distance).

Contrary to the Hausdorff distance which compares sets in metric space and can be used to compare trajectories, the Fréchet distance has the comparison of trajectories as main purpose. As Figure 2.4 points out the Fréchet distance better describes the distance of curves which are close to each other, but follow different directions and the minimal distance cannot be kept when the curves are followed sequentially. This is often illustrated with a dog-owner, a dog and a leash. The Fréchet distance is the length of the minimum required leash between the dog-owner and the dog when the dog-owner follows one curve and the dog the other from start to end. This is not yet a precise definition. The calculated distance can still vary depending of restrictions like the dog-owner and the dog are allowed to move back and forth or if they have to walk the curves without being able to step back. If they can move back and forth this is called the weak Fréchet distance. The weak Fréchet distance can be calculated in polynomial time ($O(p \times qlog(p \times q))$ where p and q are the number of edges of the curves P and Q) [1]. Another variation of the Fréchet distance is the discrete Fréchet distance where only the vertices of the curves are taken into account for calculation. The calculation of the discrete Fréchet distance is also possible in polynomial time [25].

For the calculation of the Fréchet distance in [1] a free-space diagram is used. This is not only important for calculation, but it can also serve for illustrative purposes. Figure 3.4 shows an example of two GPS traces and their respective free-space diagram. The free-space diagram is calculated by comparing each point on one trace with every point on the other trace. If the distance between those two points is below ϵ then this combination is marked as free, if not it is marked dark shaded. If there is a possibility to draw a line from bottom-left to top-right in the free-space diagram while monotonously moving up and right, then the two traces fulfill a Fréchet distance equal to or below ϵ.

For performance measurements multiple implementations for the Fréchet distance were considered. The first implementation[82] is an extension for the JTS Topology Suite[77]. To use this implementation minor bug fixes had to be done within this source code. For all Fréchet distance implementations all trajectories with a length of only 1 point had to be rejected. Therefore, only the first test scenario, like for the Hausdorff distance, was taken into account. For evaluation purposes a break was used if the calculation exceeded 5000 ms



(a) GPS traces (b) Free-space diagram

Figure 3.4: Two GPS traces and their free-space diagram [27]

which it did for this evaluation. Out of 17550 possible calculations only 114 were possible within 5057 ms. In [15] various possibilities of calculating the Fréchet distance are displayed. There is also an implementation[52] available for the methods described in the paper. In this work the displayed examples on the web page were first considered for performance evaluation. The 1.1-approximation of Euclidean (in 2 dimensions) can calculate 8052 distances in 5002 ms, the L-1 in 2 dimensions all 17550 distances in 2319 ms and the L-infinity in 2 dimensions can also calculate all 17550 distances in 3346 ms. Conclusively, the Fréchet distance has a significant longer calculation time than the Hausdorff distance, but has also a stronger predication about the similarity of two trajectories. Additionally, implementations of the Fréchet distance have a high variance in execution time.

3.4 Angle Components

The distance mentioned for points, edges, and trajectories do not directly give information about the similarity regarding the direction. Indirectly, one can infer an information about the difference in direction when comparing different distance measurements, e.g., see Section 3.2 and Figure 3.2b. Nevertheless, the information about the direction is important and should be added as a direct

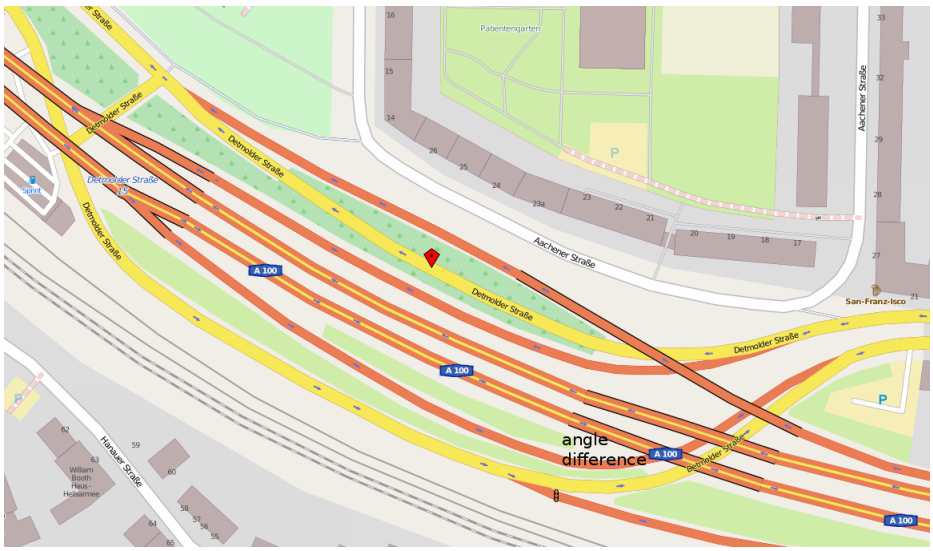


Figure 3.5: A difference in angle at a highway departure

component to the distance measure. For example, the "zick zack" lines depicted in Figure 2.4 show the semantic difference between the Hausdorff and the Fréchet distance. The main difference one can spot directly is that the two curves always follow different directions, and therefore they should have higher distance which can be expressed by the Fréchet distance to a certain degree. Nevertheless, just adding the angles or slopes of the two curves as additional information also helps to find out that the two curves should have a higher distance than their Hausdorff distance. Hence, in this case the Hausdorff distance and additional information about the angle can replace the Fréchet distance, but also vice versa. The question remains if there are scenarios which cannot be expressed by the Fréchet distance, but via a combination of a distance measure and a difference in angle or slope. Figure 3.5 displays the highway departure "Detmolder Straße" of the highway "A 100" in Berlin, Germany. This is a typical departure of a highway where the cars leave on the right hand side of the highway, but then are directed to an exit on the left of the highway. Figure 3.6 depicts two idealized trajectories according to the real world model and shows the difference in angle which arises on a highway departure. In this case the angle is 60°and marks a strong difference between both trajectories. If the ϵ for the Fréchet distance is set to the value depicted in the figure, the trajectories are matched regardless of the strong difference in angle and the different course of roads.

As we have seen, it is important to directly measure the angle of the trajectories and use it for the matching of trajectories. Next, there are several alternatives to combine the angle difference with the previously presented dis-

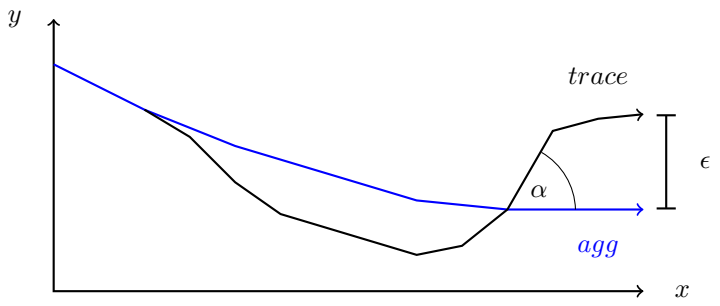


Figure 3.6: The aggregation represents a highway and the individual trace represents a departure from the highway

tance measures. First, it has to be noted that for calculating the angle, two points are necessary. This means that point-based comparisons also have to take into account the edge after or before the point or both which destroys simplifications that were possible with point-based distances, e.g., the iteration has to take the successor or predecessor into account. One approach is to integrate the angle as third dimension. Therefore, the angle has to be weighted to normalize the dimensions or in other words to fit to the other two metric values. This weighting can also be adjusted according to the preference for the angle. For a point-based approach with Euclidean distance the calculation is:

$$d = \sqrt{(x_a - x_b)^2 + (y_a - y_b)^2 + w \times (\alpha_a - \alpha_b)^2}$$

where point a is compared to point b. Nevertheless, if w is static, this calculation does not normalize the angle. A difference in angle is independent from the length of the edges. The angle can be normalized according to the distance of the edges, but this has to take into account all points of the edges:

$$d_a = \sqrt{(x_a 1 - x_a 2)^2 + (y_a 1 - y_a 2)^2 + w \times (\alpha_a 1 - \alpha_a 2)^2}$$
$$w = \omega \times \frac{d_a + d_b}{2}$$

where distance d_a is the distance of an edge at point a and w is then dynamically adjusted to the mean length of the edges of the points which are compared. The Euclidean distance with angle as third dimension has a higher calculation complexity than without the angle. Additionally, the edge distances have to be calculated to normalize the angle and the angle itself has to be calculated for two edges plus the calculation of the Euclidean distance is more complex for 3 dimensions than for 2. Within this work the execution time was evaluated for the Euclidean distance with angle "add-on" with all the same circumstances as the tests for the edge-based distances (although it is a point-based distance, the edges are needed for the angle calculations). The execution time for 66528 calculations was 705 ms which is a factor of 10.9 compared to only calculating

the Euclidean distance of two points.

For edge-based distances, adding the angle seems straightforward since already the edges are taken into account and no more additional data is needed. As mentioned in Section 3.2 the higher distance of the possible ones is taken for comparison. This distance can be adjusted with the angle as Euclidean distance with a third dimension:

$$d = \sqrt{d_h^2 + w \times (\alpha_a - \alpha_b)^2}$$

The execution time (same environment as before) for the distance via the perpendicular with the angle added was 768 ms for 66528 calculations which is a factor of 1.86 compared to the calculation without angle and a factor of 11.9 compared to the calculation of the Euclidean distance.

Furthermore, the angle can be integrated in the trajectory-based distances. For calculating the angle, the edges have to be taken into account for the calculation of the trajectory-based distances. Therefore, instead of points, the edges have to be made available to trajectory-based distances and for a trajectory with n points only $n - 1$ points can be compared because the additional point is needed for the calculation of the angle of the point before or after. The test scenario for the execution time evaluation for the trajectory-based distances including the angle is similar to the first test scenario of the trajectory-based distances with the difference that fewer calculations are done because of less available data. Due to the short length one trajectory had to be removed to ensure execution of all trajectory-based distances with angle extension. Except for one trajectory, in this scenario the same trajectories are evaluated and they include a combination of the following length of edges (not points) of the available trajectories: 5,5,5 and 5 without variation of length. The possible calculations in this scenario are 8816. For the Hausdorff distance with angle calculation the execution time is 412 ms which is a factor of 5.16 compared to the calculation of the Hausdorff distance without angle and a factor of 43.4 compared to the calculation of the Euclidean distance. The integration into the Fréchet distance is similar to the Hausdorff distance. The angle is seen as third dimension. Only two implementations of the 4 presented in Section 3.3 can be adjusted to use the angle extension because they can calculate the Fréchet distance for multiple dimensions. The implementation that were omitted only support 2-dimensional calculation. The L-1 implementation[52] with angle as third dimension can calculate 8816 combinations in 2639 ms with a factor of 6.4 compared to the Hausdorff distance with angle extension, a factor of 2.27 compared to the L-1 implementation without angle and a factor of 278.0 compared to the calculation of the Euclidean distance. The L-Infinity implementation[52] with angle as third dimension can calculate 8816 combinations in 1747 ms with a factor of 4.24 compared to the Hausdorff distance with angle extension, a factor

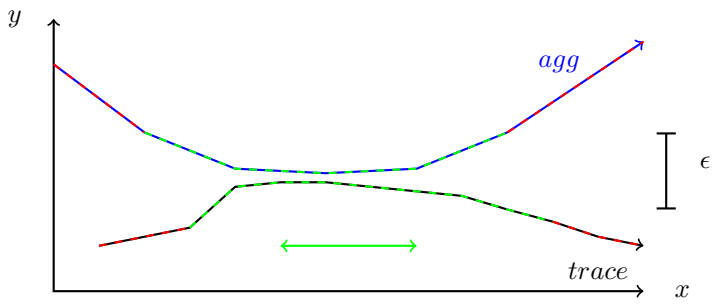


Figure 3.7: Iterative matching of trajectories

of 1.04 compared to the L-Infinity implementation without angle and a factor of 184.0 compared to the calculation of the Euclidean distance. Remarkably, the L-Infinity implementation is not significantly slower with angle extension than without even though the angle has to be calculated. This means that the calculation of the angle is of marginal complexity compared to the remainder of the calculation.

3.5 Iterative Matching

In the previous section the discussion was about the matching of whole objects: points, edges, and trajectories. Nevertheless, when merging a trajectory into an aggregation of trajectories, the decision whether it is matched or not should not be based only on the complete trajectory, but also on its parts or possible parts, also called subtrajectories. One approach to cope with the matching of subtrajectories is iterative matching. When matching iteratively, first one point or one edge is matched, then more points of the trajectory are added to the match until a certain distance threshold is reached and the matching gives the subtrajectory to be merged. Figure 3.7 depicts an iterative matching process based on the Fréchet distance ϵ. The green arrow illustrates the iterative process going in both directions. The green dashed edges show the matched part while the red dashed edges show the part which cannot be matched.

The iterative matching can be based on combinations of distance function. It can start with a point-based distance, then use an edge based distance and with 3 or more points for each trajectory trajectory-based distances can be used. Another important variation is if stepping back is allowed, comparable to the weak Fréchet distance. The next point of the trajectory and the next point of the aggregation might not fit, but the point before in the aggregation might fit

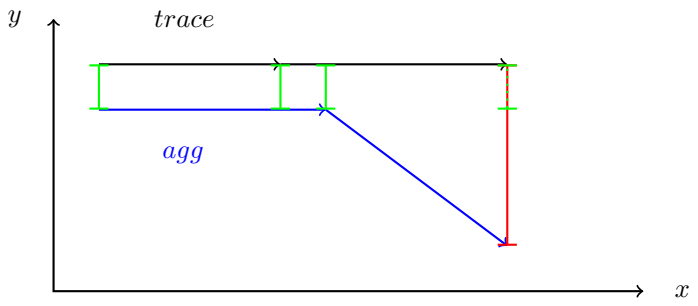


Figure 3.8: Matching of two edges to one edge

the current point in the trajectory to be matched. Figure 3.8 displays two edges of an aggregation and two edges of a single trajectory. While the edges on the left can be matched, the edges on the right cannot because of their high distance of the points on the right. Nevertheless, using the perpendicular, the right edge of the single trajectory can be matched to the left edge of the aggregation. Hence, in this case the match is possible if stepping back or keeping the current position is allowed.

Another aspect is the matching of parts of edges when edges are added to the trajectory. This is of relevance when a complete edge cannot be added because then the threshold is exceeded. In this case it is interesting to look at parts of the edge to add. Figure 3.9 shows a similar scenario as in Figure 3.8 but with aggregation and single trajectory switched. Here, the part of the single trajectory which can be matched is dotted green. Additionally, the perpendicular whose length is equal to the threshold is drawn in green and marks the end of the match of the trajectory. To find the part of edge which can be added to the matched trajectory, it is possible to search for a solution or to calculate the solution directly. If the distance function is seen as a black box then a search algorithm can find the part of the edge which should be added to the match. For searching a binary search is suitable because the distance function will tell if the threshold is exceeded or not. In a binary search the edge is splitted in two edges of equal length, then the edge which is connected to the already matched trajectory will be tested with the distance function. If the threshold was exceeded, the length will be reduced to one fourth and if the threshold was not exceeded the length will be increased to three fourth. This will be repeated until a certain precision is reached. Of course, the direct calculation of the part of the edge which can be added to the trajectory has a lower complexity and therefore the calculation is faster. Nevertheless, this has to be adjusted to the distance calculation and is not a general solution.

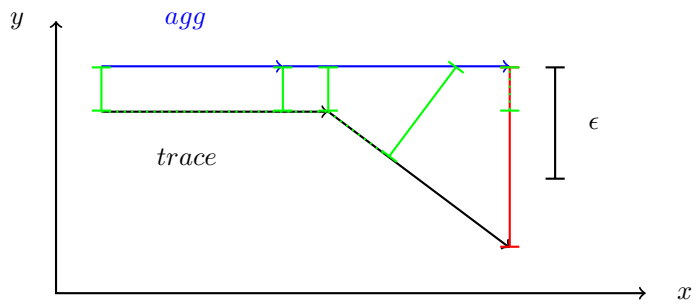


Figure 3.9: Partial matching of edges to the aggregation

Depending on the distance functions and the algorithm, the total complexity has a high variance.

3.6 Global Matching

This section discusses the matching of trajectories as a whole process. The scenario is to match a single trajectory to a trajectory aggregation where also a partial match is allowed. Preceding steps to the previously introduced matching methods can help to reduce the complexity of the global matching process. If the matching is based on thresholds of distances then preceding steps can exclude probes which are not necessary because they do not lead to a match. An illustrative example of a preceding step which falls into this category is the use of a bounding box to exclude elements which are not within this bounding box. Figure 3.10 depicts a single trajectory which is going to be included in to trajectory aggregation. Nevertheless, for matching with the threshold (or e.g. the Fréchet distance) ϵ only the part of the network which is within the permanent green bounding box (minimal bounding box plus threshold) needs to be taken into account. All other parts of the trajectory aggregation are also not matched if they are taken into account. To find the trajectories in the aggregation which are in the bounding box, the use of a spatial index is beneficial. Conclusively, the selection via the bounding box reduces distance calculations, but adds the creation and the search within a spatial index. Altogether, the use of the bounding box seems beneficial and is also able to limit the total calculation costs by keeping the distance calculations approximately proportional to the size of the single trajectory which is going to be added. The use of the bounding box can be extended to not only reducing the trajectory aggregation for the whole single trajectory, but also for its parts, e.g., its edges.

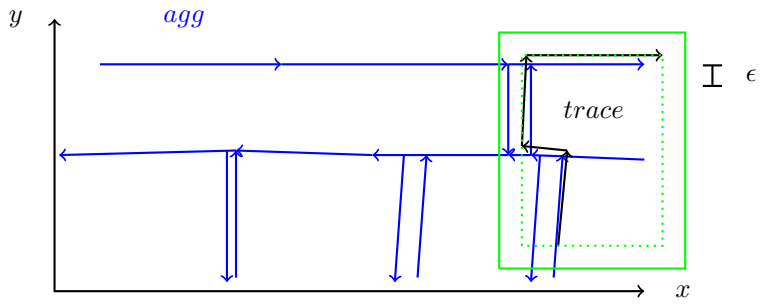


Figure 3.10: Marking the affected area by using a bounding box with threshold

Another aspect is the sequence of the matching. Normally, the matching is oriented on the single trajectory which is added to the aggregation. Based on this trajectory, elements to match have to be searched within the aggregation. For this purpose a spatial index is again advisable. Searching in a spatial index is often based on a bounding box search based on a point or a rectangle. Hence, this can be used to identify potential matches and after that a refinement step, or in other words doing the actual matching, is necessary. Which spatial index is used, does not need to be defined which means the spatial index is interchangeable. The interface to the spatial index needs to be defined beforehand, e.g., if the distance search uses a bounding box or a point. Figure 3.11 depicts a global matching process which is based on an edge-based distance comparison. Based on the *new* single *trajectory* the complete trajectory *aggregation* is *reduced* to a new *reduced aggregation* which is only used for processing the new trajectory. The new trajectory is processed by iterating through all edges (if not an edge-based distance is used then the global matching process might be different). First *a new edge is selected* and used to *search near edges* in the *reduced aggregation*. All *near edges* are taken into account for the actual *matching*. In the *matching* process the edge-based distance is completely calculated and the new *merge candidates* are determined. The process ends with the *merge candidates* as result.

3.7 Conclusion

This chapter introduced distances and processes to match trajectories. Methodological shortcomings of naive approaches can be fixed using different (starting the measurement from the crossing of the perpendicular) or additional measurements (angle components). Another important factor for the final result is

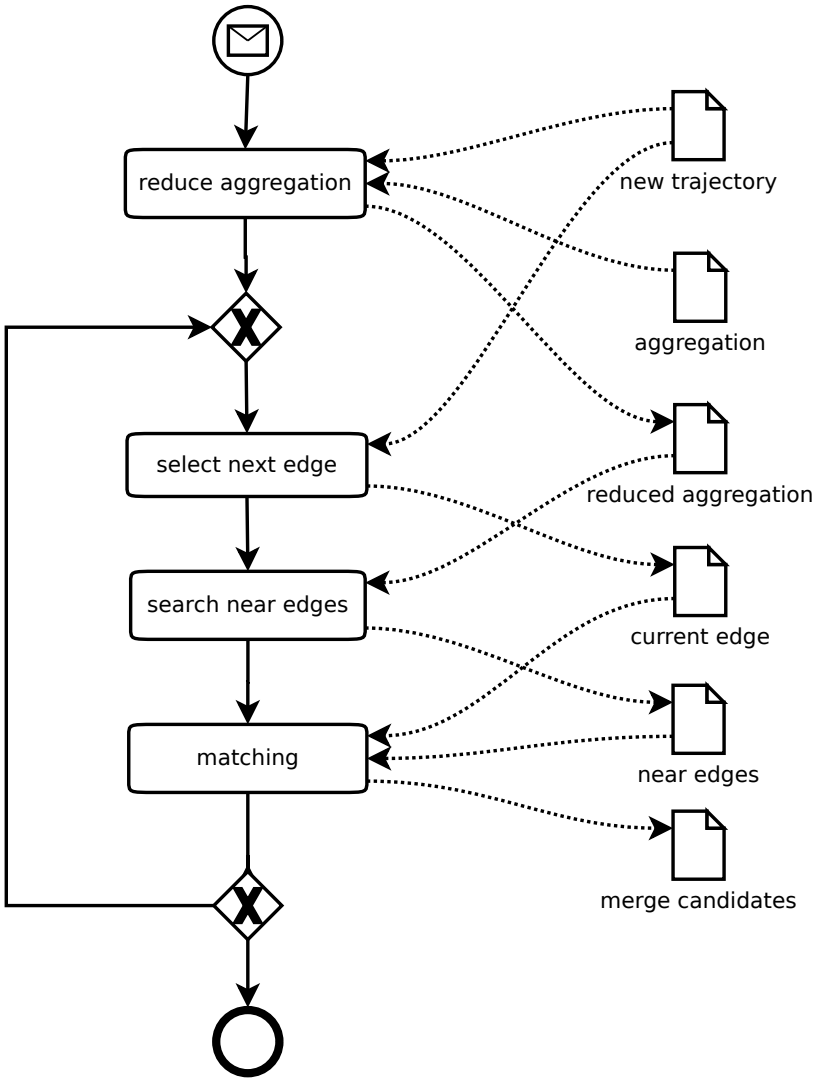


Figure 3.11: Global edge-based matching process

the process or the iteration strategy which is used to identify matchings based on a certain distance. While the success of naive approaches is only limited based on their methodological shortcomings, the success of more sophisticated approaches cannot be estimated without an exemplary evaluation. Therefore, it cannot be stated that more complex measurements like the Fréchet distance will provide better results than simpler distance measures which can also be enriched by angle components.

This chapter also provides performance measurements for the discussed distance measures. Within the point-based distances, the distances which are more adjusted to the shape of the earth have the higher calculation costs. The edge-based distances have calculation costs which are linear to the point-based distances and are more complex by about the factor 4. The complexity of the trajectory-based distances is highly dependent on the number of nodes of the trajectories. The complexity varies along different implementations, but is linear within them. Furthermore, the exemplary tests showed that the integration of the angle to edge-based distances increases the complexity by about the factor 2, but it increases the complexity of the trajectory-based distances by about the factor 6. The reason behind this is that a whole new dimension has to be added to the trajectory-based distances. Table 3.2 summarizes the execution times for all distance types.

point-based distances - 184212 comparisons			
distance	reference	execution time	factor
Manhattan Distance		87 ms	0.49
Euclidean Distance		178 ms	1
coordinate adjusted		230 ms	1.29
Haversine distance	[66]	598 ms	3.36
Vincenty's formulae	[53]	2163 ms	12.15
Vincenty's formulae	[73]	3090 ms	17.36

edge-based distances - 66528 comparisons			
distance	reference	execution time	factor
crossings of the perpendicular		413 ms	1
with angle component		705 ms	1.71

trajectory-based distances - 17550 comparisons			
distance	reference	execution time	factor
Hausdorff		159 ms	1
Fréchet JTS	[82]	+5057 ms	+31.81
Fréchet 1.1	[52]	+5002 ms	+31.46
Fréchet L-Infinity	[52]	3346 ms	21.04

Table 3.2: Distance execution times

Chapter 4

Merging of Trajectories

After the matching of points, edges or trajectories (objects), the matched candidates should be merged together. Depending on the global matching process the outcome is either a match of a set of objects with a set of objects or directly an object with an object. The outcome of the merging is a modified aggregation where the objects of the single trajectory have been merged into.

An important aspect for the merging of trajectories is the evolution of the aggregation over time. The more trajectories are used to build the aggregation the more reliable the aggregation should be. Furthermore, the aggregation should become more stable over time. To ensure a process after which the aggregation becomes more stable after more mergings, an information about the stableness of the aggregation has to be stored and linked to the object in the aggregation. The most straightforward way to do this is to count the number of objects which were already merged into the object of the aggregation. This count can be added as attribute to the object of the aggregation. Additionally, this count should be taken into account for the merging itself. Since a higher count expresses a higher stableness, the object should be more inert when merging. The following formula expresses the inertness of an aggregation through a weight which can be realized as a count:

$$o_n = w_a \times o_a + (1 - w_a) \times o_t$$

where o_n is the new object of the aggregation, w_a is the weight of the old object of the aggregation (can be expressed with a count c: $w = \frac{c}{c+1}$), o_a is the old object of the aggregation and o_t is the object of the single trajectory. Of course, the prerequisite is that objects are directly merged with each other, not sets of objects and that the addition is defined for the objects.

This chapter is organized as follows: Section 4.1 discusses point-based merging approaches, Section 4.2 edge-based merging approaches, and Section 4.3 trajectory-based merging approaches. Section 4.4 discusses the global perspec-

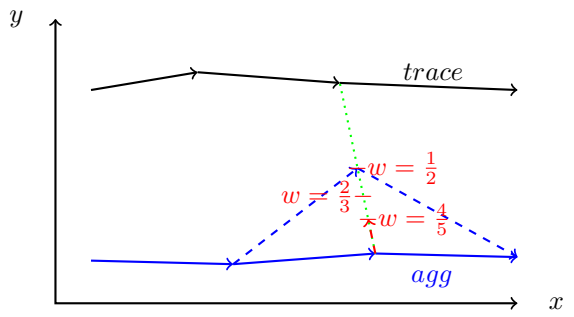


Figure 4.1: Point-based merging example with two trajectories

tive, including finalization steps to be taken into account. Section 4.5 concludes this chapter.

4.1 Point-based Merging

The input for point-based merging are either two points or two sets of points. Figure 4.1 show an example of a simple point-based merge with 3 different weights. The green dotted line connects the two points which should be merged. Depending on the weight the new point of the aggregation is set at a different position along this line. The new modified aggregation with weight $w = \frac{1}{2}$ is displayed with dashed blue lines.

Normally, the merging process has to cope with multiple points to be merged or sets of points should be merge with sets of points because they were all matched together. There are different approaches, how multiple points can be handled in the merging process. One approach is to create a new point as the average of all points which are marked to be merged together:

$$p_n = \overline{w_a} \times \overline{p_a} + (1 - \overline{w_a}) \times \overline{p_t}$$

where p_n is the new point of the aggregation, $\overline{w_a}$ is the average weight of the old set of points of the aggregation, $\overline{p_a}$ is the average of the old points of the aggregation and $\overline{p_t}$ is the average of the set of points of the single trajectory. First, the points in the set of points of the aggregation are used for an average point of the aggregation ($\overline{p_a}$). Also for the single trajectory the average point is calculated. After that, the new point of the aggregation is adjusted with the weight. Figure 4.2 shows an example of merging 3 points in each set (aggregation an single trajectory) to one point. The precondition is that the threshold is set that all points can be matched with each other on the other trajectory. The weight of the aggregation is $\frac{1}{2}$. This approach can reduce the number of points in

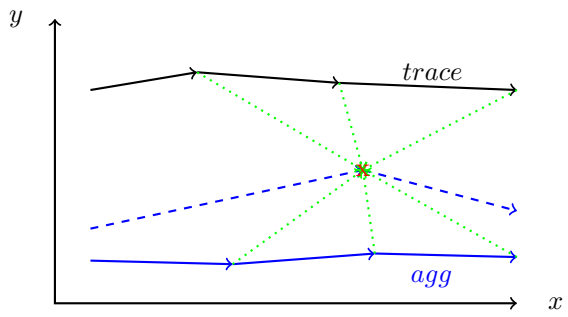


Figure 4.2: Point-based merging example with one new point for sets of points

the aggregation because many points are replaced with one point. Basically, the granularity has a lower bound restricted by the threshold so that all matches within the threshold are combined. One advantage might be the harmonic frequency of points in the aggregation. A disadvantage is the restriction of precision to the threshold. It seems advantageous to be able to have a higher precision not restricted by the threshold and a moderate threshold that still enables finding not very close but close matches. Hence, the threshold and the possible precision should be independent.

Another approach is to iterate the points of the aggregation and sequentially merge all matched points of the single trajectory with the one current point of the aggregation:

$$p_n = w_a \times p_a + (1 - w_a) \times \overline{p_t}$$

where p_n is the new point of the aggregation, w_a is the average weight of the old point of the aggregation, p_a is the average of the old point of the aggregation and $\overline{p_t}$ is the average of the set of points of the single trajectory. Figure 4.3 shows an example with one new point for each old point of the trajectory aggregation where the weight of the aggregation is $\frac{1}{2}$. Each old point is combined with every matched point of the single trajectory. First, the points of the single trajectory are averaged and then the weight is applied to the point of the aggregation and the averaged point of the single trajectory. This process keeps the sampling rate of the aggregation so that it can be defined independently from the threshold. In a process where new trajectories are added to the aggregation without a previous match, the sampling rate can be defined and this rate stays the same after each merging of this type. Additionally, looking at a global merging perspective, a process based on an iteration of the matched points of the aggregation is straightforward to process because the order of the task is already defined by the iteration through the aggregation.

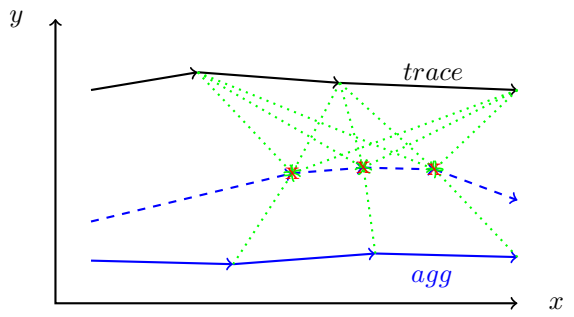


Figure 4.3: Point-based merging example with one new point for each iteration of the aggregation

4.2 Edge-based Merging

As mentioned for the point-based merging, also for the edge-based merging the input can be a set of edges of the trajectory aggregation and a set of edges of single trajectory. Additionally to the possibilities of different scenarios, which edges are mixed together, there are many possibilities to calculate an average of two edges, not only by the arithmetic mean. Therefore, in the following the scenario is restricted to an iteration of edges of the trajectory aggregation where one edge of the aggregation has multiple matches in the single trajectory. Using the arithmetic mean, one option to merge is to calculate the mean of start and end nodes separately. Figure 4.4 shows the result of averaging start and end nodes. Nevertheless, one problem emerges: The end node of one edge was in the old aggregation the start node of the other edge. Nevertheless, in the new aggregation this node is now moved onto two separate locations. This causes a connection problem of the two nodes which cannot be solved by just connecting them which can be seen in Figure 4.4. The connection has a completely different direction than the edges that are connected. A straightforward solution is to merge the nodes which previously had the same location. Figure 4.5 displays the result with a merged node in the middle which was displaced due to the previous merging step. Here, the connection can stay the same as in the old aggregation.

For the merging of edges information about the direction is available which means that using the perpendiculars in the merging is possible. Figure 4.6 shows a merging for the purple edge in the aggregation where crossings to the single trajectory are found and according to these crossing and the weight of the aggregation (in this case $\frac{1}{2}$) the new merged points of the edge of the aggregation are set. If the crossings are known then for the start and end node the new

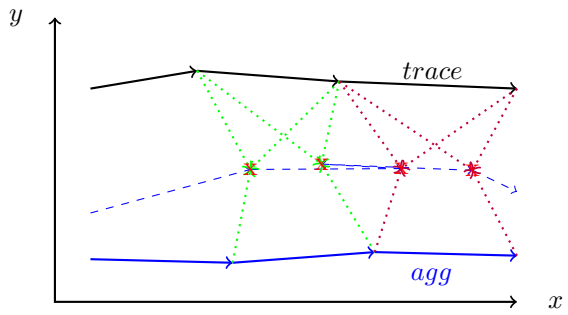


Figure 4.4: Edge-based merging example with iteration of the aggregation and averaging of start and end nodes

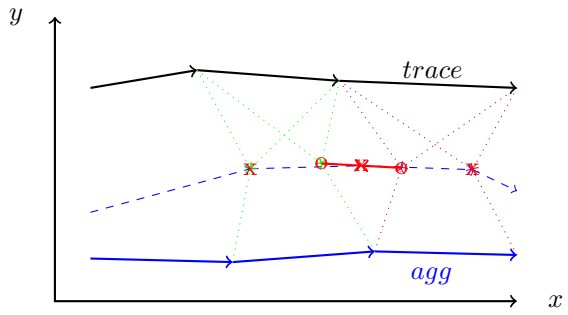


Figure 4.5: Edge-based merging example with iteration of the aggregation and averaging of start and end nodes and additional merging of displaced nodes

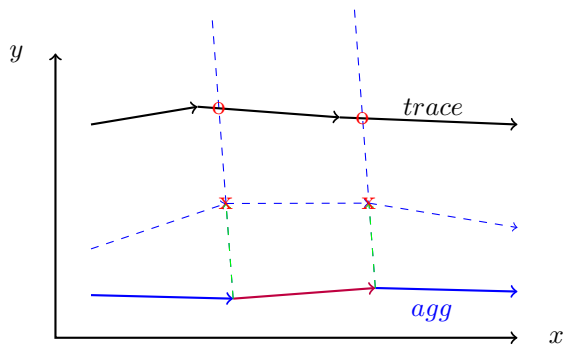


Figure 4.6: Edge-based merging example with iteration of the aggregation and finding new nodes along the perpendicular

merged node is calculated with

$$p_{n,se} = w_a \times p_{a,se} + (1 - w_a) \times p_{c,se}$$

where $p_{n,se}$ is the new start or end node of the edge of the trajectory aggregation, $p_{a,se}$ is the old start or end node of the edge of the trajectory aggregation and $p_{c,se}$ is the crossing of the perpendicular of the start or end node of the trajectory aggregation with the matched edges of the single trajectory.

One feature of this method is that the points or edges of the trajectory aggregation are only moved parallel. This keeps not only the number of edges constant. Additionally, the distance between points in the trajectory aggregation stays nearly constant. Nevertheless, this property also has a strong disadvantage. If the single trajectory is more fine-grained and has a higher precision which is favorable for some scenarios, a less fine-grained trajectory aggregation does not adapt to the higher precision if the number of points and their distance cannot be changed.

Nevertheless, also the perpendiculars of the single trajectory and their crossings on the trajectory aggregation can be used additionally. They can be used to increase the precision of the trajectory aggregation where the granularity of the single trajectory is significantly higher. One possibility is that always a new point is created in the trajectory aggregation is created when a crossing of the perpendicular of the single trajectory and the trajectory aggregation has a higher distance to a point of the trajectory aggregation than a certain threshold. A constant factor multiplied by the length of the edge of the single trajectory can be used as threshold. This is suitable because the length of the edge of the single trajectory indicates the granularity of the single trajectory at this edge. Figure 4.7 shows the crossings which were found and can be used as new points in the trajectory aggregation. This increases the granularity and can be used

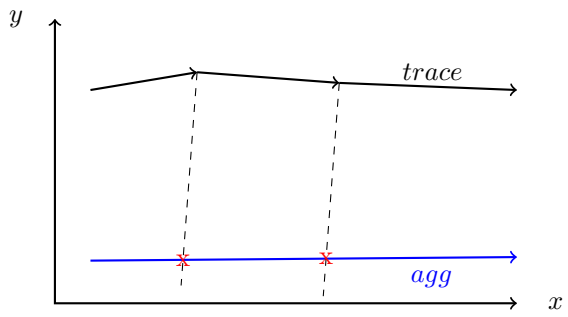


Figure 4.7: Increasing the granularity of the trajectory aggregation by creating new points at the crossings of the perpendiculars of the edge of the single trajectory with the trajectory aggregation

as a preprocessing step for the previously introduced merging methods.

Matching and merging methods are fundamental parts of aggregating trajectories. Nevertheless, the complete process of matching and merging and the order of matchings and mergings are important for the final result. Algorithm 1 shows a pseudocode of a complete integration using the perpendicular distance (Figure 3.3) and edge-based merging using the perpendicular from the aggregation (Figure 4.6). The procedure *Integrate* calls the two main parts, the procedures $Match$ and $PriorityMerge$.

The procedure $Match$ fills the data structure *matches* which describes matches with an edge of the individual trajectory *edge*, a connection (or edge) within the aggregation *con* and the calculated perpendicular distance *distance*. In line 6 the connections in the aggregation are filtered before a perpendicular distance is calculated. It is preferable to choose a data structure for the connection in the aggregation which can filter the results in a way that everything within the threshold using the perpendicular distance is included, but not many more connections. After the results are filtered, the actual distance is calculated (line 8). If the distance is less than the threshold the result is stored as a match within the previously mentioned data structure *matches*.

The procedure $PriorityMerge$ iterates the matches. The matches are iterated according to their priority which means in ascending order sorted by their distance. This ensures that first all the best matches are merged and only if a node in the aggregation was not merged already it is available for merging in a match with a little higher distance. Here, one can see that this order is important for the final result. If the matches are iterated in a random order, nodes are moved according to matches which are not considered to be the best match and after that are not available for merging within a better match. Nevertheless,

this is only one preferable approach. It is also possible to merge a node based on all matches. This leads to a different result which can also be preferable. In lines 14 and 17 it is checked if either the first node (from) or the second node (to) of the aggregation was not already used in a merging. If this is true, then the procedure $MergeNode$ is called.

$MergeNode$ calculates the perpendicular of the connection of the aggregation crossing the previously selected node of the connection of the aggregation. After that the crossing of this perpendicular and the edge of the single trajectory is calculated (line 22). After that it is checked whether the crossing lies on the edge of the single trajectory. Only if that is the case the node of the aggregation is moved along its perpendicular towards the edge of the single trajectory. The procedure $MoveConnection$ executes this movement. Finally, if the return value of the $MergeNode$ procedures indicates a successful merge ($true$) the node is added to $alreadyMergedNodes$.

4.3 Trajectory-based Merging

Taking a whole trajectory into account when merging creates even more alternatives than taking points or edges into account. The trajectory-based merging has as input two trajectories which can be subtrajectories that were matched before. Though the matching was already exhaustive with two trajectories, it is not necessary to take into account that there can be sets of trajectories as input. Nevertheless, the two matched trajectories as input can be seen as two sets of points or two sets of edges. This means that the previously discussed point-based merging and edge-based merging can also be applied to trajectories. This can either be done directly by regarding the two trajectories as two sets which were matched. Nevertheless, an indirect way by iterating the matched points or edges seems more appropriate because it can take into account the closeness of the sub-items (points or edges).

4.3.1 Merging with Normalization

Having discussed the possibility of applying former merging methods, we will now focus on methods that are only possible when the complete trajectory is known. Figure 4.8 shows a merging process where the granularity is harmonized throughout the complete trajectory. First, a distance threshold is defined. This can be done, taking into account the average distances on each trajectory. It is preferable to choose the smaller average distance of the two trajectories which is in this example the single trajectory to increase the precision to the precision of the more precise trajectory. Taking the average of the complete trajectory

Algorithm 1 Integration of single trajectory in aggregation

```
1: procedure INTEGRATE(individualTrajectory, aggregation)
2:   matches ← MATCH(individualTrajectory, aggregation)
3:   PRIORITYMERGE(matches)
4: procedure MATCH(individualTrajectory, aggregation)
5:   for each edge in individualTrajectory do
6:     nearCons ← FINDNEARCONNECTIONS(edge, aggregation)
7:     for each con in nearCons do
8:       distance ← GETPERPENDICULARDISTANCE(edge, con)
9:       if distance < threshold then
10:        matches ← {edge, con, distance}
11:  return matches
12: procedure PRIORITYMERGE(matches)
13:  for each match in matches do     ▷ assuming matches sorted by distance
14:    if match.con.fromNode not in alreadyMergedNodes then
15:      if MERGENODE(match.con.fromNode, match.con, match.edge) then
16:        alreadyMergedNodes ← match.con.fromNode
17:    if match.con.toNode not in alreadyMergedNodes then
18:      if MERGENODE(match.con.toNode, match.con, match.edge) then
19:        alreadyMergedNodes ← match.con.toNode
20: procedure MERGENODE(node, con, edge)
21:  perpendicular ← GETPERPENDICULAR(node, con)
22:  crossing ← GETCROSSING(perpendicular, edge)
23:  if CROSSINGWITHINEDGE(crossing, edge) then
24:    con ← MOVECONNECTION(crossing, node)
25:    return true
26:  return false
27: procedure MOVECONNECTION(crossing, node)
28:  node ← (node.weight × node + 1 × crossing) / (node.weight + 1)
29:  node.weight ← node.weight + 1
```

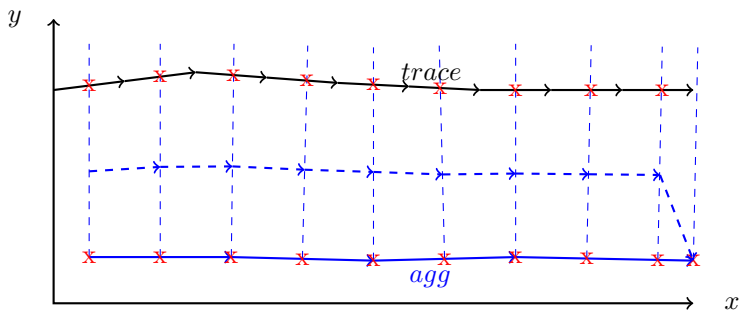


Figure 4.8: Merging trajectories by merging points found by distance thresholds

avoids exceptions for the precision, for example when recording a trajectory with a time interval and staying more time on traffic lights. After finding the distance threshold, the perpendicular for every point at an interval of the distance threshold is calculated and a crossing with the single trajectory is found. The points within this distance threshold of the old trajectory aggregation are than replaced by the new point of this distance threshold found via the points on the perpendicular, adjusted with a weight (in this case $\frac{1}{2}$). For the final point of the trajectory aggregation there was no match via the perpendicular on the single trajectory. Therefore, it was not replaced by another point and is still a part of the new trajectory aggregation.

4.3.2 Sweep Algorithm

Another approach to merge a single trajectory with a trajectory aggregation is to use a modified sweep-line algorithm from TRACLUS[43]as shown in Figure 2.5 in Section 2.1.3. In [43] the sweep-line algorithm is used to find a cluster representation for multiple edges. Nevertheless, this scenario is different from the scenario in this work. In this work, we want to find a representation for two trajectories, not edges and furthermore, one trajectory can have another weight than the other. In the original work, the algorithm uses a constant $MinLns$ which is set before and describes the minimal amount of edges that the sweep-line has to cross to set a new point for the representative trajectory. The angle of the sweep-line is the perpendicular to the average angle of all edges with equal weight. The algorithm detects every change of the number of crossings with the sweep-line and, if $MinLns$ is fulfilled, sets a point. All the points that were set are connected and the direction is given by the sweep direction.

To use the sweep-line algorithm within this work, it has to be extended, but can also be restricted. First, the parameter $MinLns$ can be set to 2 and

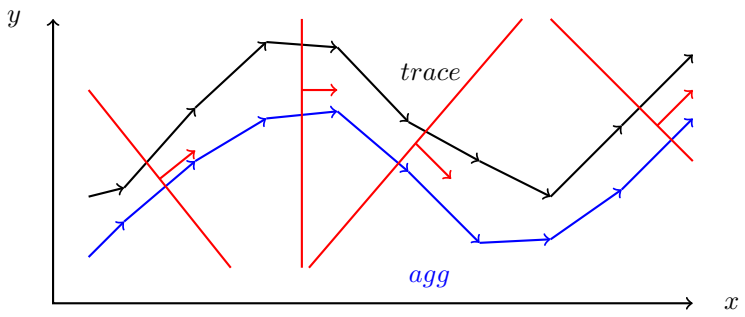


Figure 4.9: Merging trajectories by generating points when the sweep-line detects changes

does not need to be adjusted. Furthermore, the trajectories can also be seen as multiple edges. Nevertheless, it has to be defined, what happens when an end of an edge is also the start of another edge at exactly the same location. In this case it makes sense to cope with this as a change of the number of crossing with the sweep-line to adjust the new trajectory to every change in the original trajectories. Another issue is the weighting. The weight of the trajectory aggregation can be used when setting the new point as the average of the crossings of the sweep-line. When calculating this average (which is restricted to 2 points), The crossing with the trajectory aggregation and the crossing with the single trajectory can be averaged with respect to their weight. Another issue is the direction of the edges of the trajectory. In the matching process, trajectories which change their direction will be matched if this change is made by both trajectories. The problem arises that the sweep-line will only follow one direction and can cause unwanted mergings or even sweep along the wrong direction for some edges. One way to prevent this behavior is to introduce multiple sweep-lines. If the directions of the edges differ more from each other than a certain threshold to the current sweep-line then a new sweep-line is introduced or the sweep-line is adjusted to the directions of the current edges. Figure 4.9 illustrates a changing sweep-line while "sweeping" along both trajectories. The sweep-line is adjusted to the direction of every new edge it passes by. This adjustment can be done as an exponential moving average: $\alpha_{s,c} = \beta \times \alpha_{e,c} + (1 - \beta) \times \alpha_{s,c-1}$, where $\alpha_{s,c}$ is the direction of sweep-line at the current position, β is a weighting factor which is set as a parameter before, $\alpha_{e,c}$ is the direction of the edge, the sweep-line passes by, and $\alpha_{s,c-1}$ is the direction of the sweep-line before the adjustment. When the sweep-line is newly adjusted, it is important that it does not pass by at already passed nodes, which

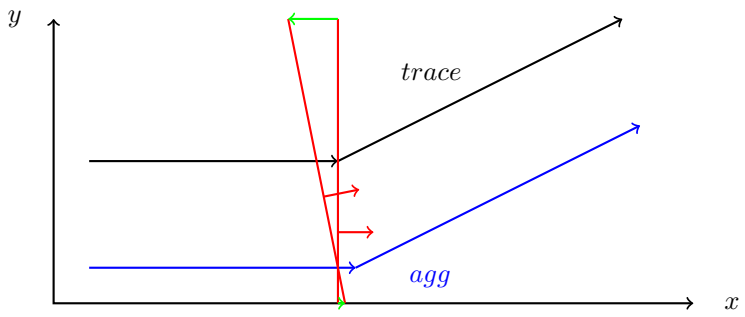


Figure 4.10: Adjusting the sweep-line backwards to prevent leaving out nodes

can be caused by a direction change. To prevent this, already passed nodes have to be marked. Furthermore, it is important to ensure, that, due to a direction change, no nodes are left out. This can be done by adjusting the direction of the sweep-line always backwards as Figure 4.10 displays. The two green arrows show the adjustment of the sweep-line. The sweep-line with the previous direction crosses the single trajectory and the trajectory aggregation. The crossing which the new sweep-line will still cross is chosen that the other crossing is on the side of the direction of the sweep when the area is splitted by the sweep line as a half-plane.

4.4 Merging: Finalization Steps

Additionally to the adjustment of the aggregation there exist cases where an individual trajectory which should be integrated adds new topological information to the aggregation network. Figure 4.11 shows an example of a T-crossing where the aggregation (blue) exists out of two traces where one is leaving the main road (top left to bottom middle) and one is entering the main road (bottom middle to top right) and the individual trajectory (black) follows the main road. After the normal merging step, the aggregation is adjusted towards the individual trajectory (dashed blue). Nevertheless, we get information from the individual trajectory which is the information that both existing trajectories of the aggregation are actually connected (red arrow). To detect this circumstance one can make use of the parts of the aggregation which were merged with the individual trajectory and look at the individual trajectory if these parts are more or less (one has to set a threshold in meters or points) directly connected. Taking this into account the individual trajectory cannot only contribute to the precision of the aggregation but also to their topology.

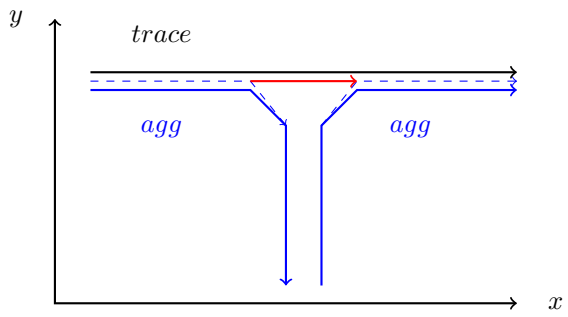


Figure 4.11: Increasing the granularity of the trajectory aggregation by creating new points at the crossings of the perpendiculars of the edge of the single trajectory with the trajectory aggregation

To improve the aggregation results the following rules were used for the connection of parts of the aggregation with the help of the individual trajectory:

1. Thresholds for the connections were introduced:

   (a) a threshold for the angle which is checked for the connection to be created in comparison to the last connection of the previous part and in comparison to the first connection of the following part.

   (b) a threshold for the distance of the connection

2. It is checked that the parts are not already connected via a slightly different trajectory. Therefore, a threshold for the steps which are used for the search is set.

3. If there is no direct connection it is checked if there is already a connection by first going back and then forward. If this applies then the last node of the previous part of the aggregation is merged with the node before until there is only a path in direction forward left. This can prevent doubled connections which are in all checked cases redundant.

4. The blocks which contain consecutive matchings are ordered by their minimal distance to each other and also sorted in consecutive order. For example, if the outgoing node of a connection is closer to an incoming node of a connection of another part than the incoming node to the outgoing then this part is ordered before the other part.

4.5 Conclusion

This chapter discussed different merging approaches including point-based, edge-based, trajectory-based, and sweep line approaches. For each approach possible deficiencies with plausible corrections were shown. Altogether, the overall result is not only dependent on local merging methods, but also how the global process is defined and how it is integrated within the global matching approach. Finalization steps can be added to the process to minimize effects which may occur when adding multiple trajectories to the trajectory aggregation. The main goal of these steps is to normalize the in-between results to prevent degenerating effects. The most important example is the creation of connections. Once a connection is set it should be checked if a new connection which is close is not already represented by this existent connection. The result quality of the approaches can only be verified by a scenario-based evaluation. All presented approaches are methodologically suited to produce good and consistent results.

Chapter 5

Privacy Preservation

One aim of this work is to provide a collection process of GPS trajectories which preserves the privacy of a user of the system. Privacy issues of the collection itself have already been discussed in Section 2.2.3. In this section the focus is on securing that the stored data does not violate anyones privacy. On a more detailed level that means that the trajectory aggregation does not reveal facts about an individual which the subject does not want to be revealed.

[13] discusses obfuscation methods for location-based data and the according user acceptance. Figure 5.1 displays the discussed obfuscation methods. The method of deletion is shown in Figure 5.1a. Deletion means the deletion of points in the trajectory which can violate someones privacy. This can be applied for the home and work location and for additional sensible locations which are often visited by the subject. For each location a radius can be set which will be used to remove all close points. Nevertheless, an attacker can assume a location within the radius when the points are cut away when entering and leaving this circle. Additionally, the combination of a home and work location, even obfuscated by deletion, can be unique for a subject. Figure 5.1b shows obfuscation by randomization. Randomization moves every location randomly from the original location the new location. The movement is randomized and also limited to a certain radius. Nevertheless, as one can observe in the three displayed scenarios, the original route is easy to reconstruct if the points can still be linked with the complete trajectory. Even the highest displayed level of randomization still provides much information for reconstruction, but destroys much precise information. Discretizing, shown in Figure 5.1c, replaces precise points with an information in which cell in a grid the location is. This is similar to randomization, but avoids reconstruction via distribution analysis. It has the effect that the position of the location afterwards is less fine-grained than the original data. Figure 5.1d depicts subsampling. Subsampling removes points from the

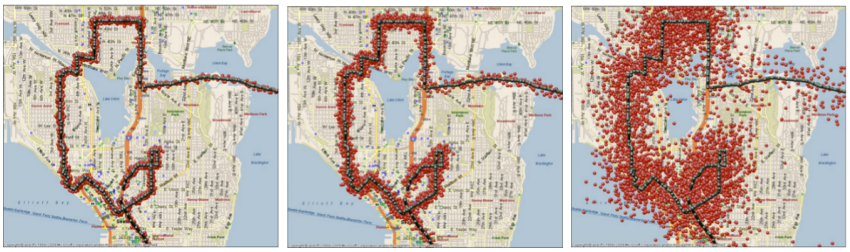
trajectory to increase the time window of the recorded trajectory. The result is less precision of movements in between, but precise points when available. The reconstruction of the trajectory can be done with a map as background information and a routing algorithm to connect the available points. Mixing, displayed in Figure 5.1e, mixes the location of one user with the location of other users. A radius for the location is chosen to mix the location with at least a certain amount of other locations of other users. This is the only presented obfuscation method which takes the identifiability of one user into account. If the location is mixed then the user cannot be distinguished within this group of users. Mixing was the most accepted method discussed in [13]. Basically, it generalizes by varying a radius until a certain anonymity, or k-anonymity[72], is fulfilled. Additionally, other generalizations (instead of adjusting the radius) are thinkable. Nevertheless, this generalization is described for one single location. Taking a complete trajectory into account, the generalization has to be based on the complete trajectory.

This chapter is organized as follows: Section 5.1 discusses the k-anonymity anonymization metric for trajectories and Section 5.2 discusses the integration of k-anonymity in the matching and merging approach. The use of differential privacy as anonymization metric for trajectories is considered in Section 5.3. Section 5.4 concludes this chapter.

5.1 k-Anonymity for Trajectories

According to [13], it is likely that a generalization approach with respect to k-anonymity[72] (see mixing) can be a favorable choice for users who want to obfuscate their location or trajectory information. k-Anonymity is a privacy measurement and does not imply a certain algorithm. Nevertheless, to make data records indistinguishable from each other, one has to generalize the data. This can be done automatically using heuristics[4] or manually. Normally, the generalization is based on quasi-identifiers (attributes which can be used in combination with other quasi-identifiers to uniquely identify a person) and sensitive attributes (attributes that reveal information that should not be revealed about a person). Therefore, the sensitive attributes are not included in the generalization process because they do not provide information to identify someone. Nevertheless, taking trajectories into account, any information, the trajectory provides (nodes, edges, connections), can be used as quasi-identifier. Additionally, any information can be a potential sensitive attribute which the user does not want to be revealed. Nevertheless, this can be ignored since all information is already seen as quasi-identifier and therefore needs to be generalized. Conclusively, the result of generalizing trajectories is no individual attribute and

(a) Deletion



(b) Randomizing

(c) Discretizing

(d) Subsampling

(e) Mixing

Figure 5.1: Obfuscation methods for location-based data[13]

additionally no distribution over individual attributes, but completely generalized trajectories.

The generalization of one-dimensional attributes can be done via hierarchies, the creation of ranges, or the suppression of digits. In some cases also an average function can be taken into account. Nevertheless, this can increase the difficulty of the clear distinction between value ranges. The generalization of trajectories is more complex because the data is multi-dimensional. Related work has already been discussed in Section 2.1.1, see Figures 2.1 and 2.2. While the presented methods in Figure 2.1 describe an approach where the focus is on a more general movement basis (a crowd moves somehow from A to B), the method displayed in Figure 2.2 has a focus on the movement itself (the crowd moves from A to B and takes this estimated path). Since this is preferable for a map construction after the anonymization it will also be the focus of this work to describe as exactly as possible how the movement happens. In [54] the courses of the road are removed from the data set. Nevertheless, in this work the focus are on the courses of the road and how they build a road network. Both approaches can be combined without restricting each other. If someone knows that a group of people went from A to B, there is no additional sensitive information revealed if he or she additionally knows how the exact path was.

5.2 Integration in Matching and Merging

k-Anonymity can be ensured by generalization and generalization is what was already described by the matching and merging processes. The new object (node, edge, trajectory) of the trajectory aggregation is a generalization of the old object of the trajectory aggregation and the single trajectory that was added. The level of k-Anonymity which can be fulfilled can be guaranteed specifically for every object. The guaranteed fulfillment of a combination of object is equal to the participating object with the lowest fulfillment.

5.2.1 k-Anonymity and weight

We can use the weight to see how many individual trajectories were already added to a certain object in the trajectory aggregation. The weight can easily be converted to a value of k and vice versa: $w = \frac{k}{k+1}$. To ensure k-Anonymity for a user who adds a new single trajectory, k-Anonymity can be checked in the matching phase. In the matching phase only objects with an asserted k that is just one below the requested k or higher are allowed to be matched. This ensures that during the merging phase only objects with a k that is equal to the requested k will be created.

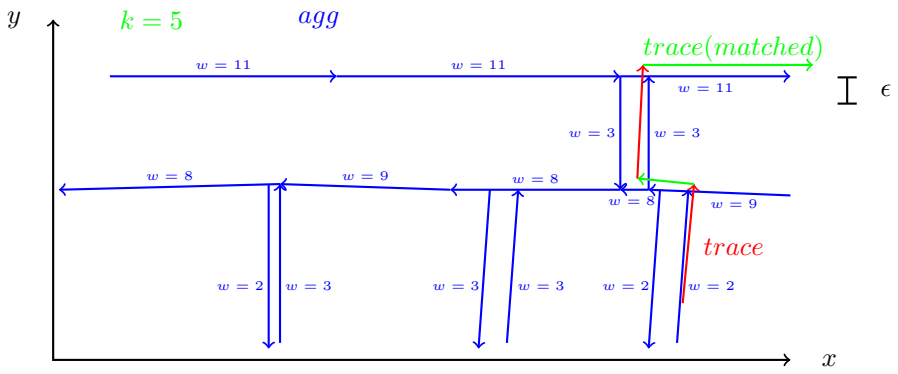


Figure 5.2: Using the edge weight to match edges according to k as threshold

The generalized object is a representative of the merged objects. Nevertheless, it is an average representation. Normally, one would anonymize according to ranges. A range can be created with the average object by just adding the threshold that was used to generate the object. This can be either static or dynamic. If it is dynamic, it needs to be linked to the object. Another possibility is the calculation of a variance of a object. A variance can be stored as an attribute of the object and adjusted with every merge. In this case, the merging has to identify the variance between the object of the single trajectory and the object of the trajectory aggregation. A variance can be the calculated distance between the two objects according to the applied distance measure in relation to the length of the object of the trajectory aggregation or, if no length is available, the absolute value. This variance can be adjusted in every merge: $v_n = \frac{v_o \times n + v_c}{n+1}$, where v_n is the new variance, v_c is the current variance, v_o is the old variance, and n is the count for the already merged objects (equal to k).

Figure 5.2 illustrates an exemplary scenario with edges which are matched against each other under the condition of the fulfillment of a given k. In this example k is set to 5 and each edge of the aggregation has a specific weight (w). If k is equal or more than w then the edge can be matched otherwise it is filtered out due to privacy resctrictions. Furthermore, the distance has to be fulfilled. In this scenario, one can imagine a main road in both directions (from left to right and from right to left where alread more individual trajectories were added and therefore with edges with a higher weight. However, on the side road less individual trajectories were added and therefore the edges have a lower weight. Hence, edges from the individual trajectory to be added (trace) are matched on the main road, but not on the side road.

5.2.2 Finding Subtrajectories

To be able to match and merge trajectories, they have to be partitioned if they do not match completely. One way to partition is to iteratively match trajectories (see Section 3.5). Nevertheless, this matching process has to be adjusted if we want to preserve privacy with k-Anonymity. Since it is a good heuristic to continue the search with the neighbors of already matched objects, the search for matches should not be restricted by objects with invalid matches (matches with a k lesser than the required k). However, after the unmodified matching, there is a need for a filter step. The filter step has to reduce the matched input by the invalidly matched objects. For point-based matches and edge-based matches it is sufficient, to just remove the objects of the trajectory aggregation with a lower k. If in the matched set all objects of the trajectory aggregation do not fulfill the required k, then the complete match is obsolete. If the matched set can be reduced because all other matched objects are still matched without the removed object. Nevertheless, with trajectories this is not true. It distorts the result if one edge of the trajectory aggregation in the match is removed, but the counterparts in the single trajectory are kept. Hence, it is necessary to partition the match on the basis of the parts of the matched trajectory aggregation with a sufficient k and repeat the matching for these parts, taking only the matched trajectory of the single trajectory into account. Figure 5.3 illustrates the matching within a filter step. It is based on the match displayed in Figure 3.7. After the match the filter step detects an edge with insufficient k (displayed in continuous red) and partitions the trajectory in the trajectory aggregation into two subtrajectories (one dashed in green and one dashed in yellow). After the partitioning the filter step runs an iterative matching with a static subtrajectory of the trajectory aggregation and continued matches on the matched single trajectory. In this case, all matched parts can be matched again and are not intersected. Nevertheless, it is also possible to leave out previously matched parts and to continue the merging with intersected subtrajectories of the previously matched single trajectory.

In [54] subtrajectories are created by using characteristic points. These characteristic points are determined by user-specified thresholds of properties of the point. These properties include the minimum angle of a turn, the minimum duration of a stop, and the maximum distance between two extracted points. These are more static properties and do not include the context of the analyzed point. A dynamic approach is described in [43]. In this approach the costs of a loss in precision are calculated for a specific point. If the costs are low, the point is removed (not a characteristic point) and if the costs are high then the point is kept (characteristic point). Overall, this is a weighting of preciseness against

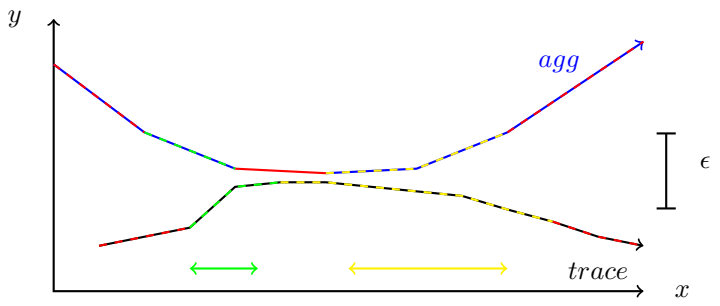
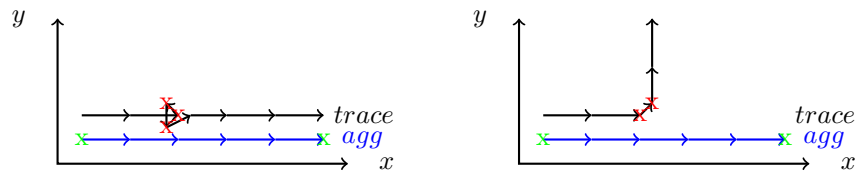


Figure 5.3: Iterative matching of trajectories within a filter step

conciseness. In this work it is also possible to create subtrajectories according to characteristic points. Both approaches are suitable for this work. Nevertheless, it is important that characteristic points of the trajectory aggregation and the single trajectory are chosen on similar locations (matchable distance in between). This should be true for parts of the trajectory aggregation and the single trajectory that can be matched to each other. The before mentioned static approach (by thresholds) and the dynamic approach (weighting) can be adjusted to foster this requirement. For the static approach, the distance to characteristic points of the trajectory aggregation can be included as an additional condition. The presented thresholds were used as a disjunction. The distance to characteristic points of the aggregation can be added as a conjunction to ensure that every characteristic point of the single trajectory is near a characteristic point of the trajectory aggregation. Nevertheless, this will also prevent the creation of characteristic points that cannot find a match in the trajectory aggregation and prevent the creation of new trajectories in the trajectory aggregation. Therefore, it should be possible to identify characteristic points if the single trajectory leaves the trajectory aggregation at this point. Figure 5.4 displays two cases for the identification of characteristic points. In the case of a GPS error at a traffic light (Figure 5.4a), characteristic points can be created because angle thresholds are exceeded. In this case, the creation can be prevented, which is preferable, because the distance to characteristic points of the trajectory aggregation is too high. In the other case (Figure 5.4b), when characteristic points are created and the single trajectory actually leaves the trajectory aggregation, the creation is also stopped which is not the desired result. In this case, this prevention of the creation of a characteristic point is not leading to a good result. Therefore, it needs to be checked if after the questionable characteristic point, the single trajectory actually leaves the trajectory aggregation. This is best checked by the decreasing distance to one characteris-



(a) Single trajectory with GPS errors inducing characteristic points

(b) Single trajectory leaving trajectory aggregation

Figure 5.4: The identification of characteristic points of the single trajectory next to the trajectory aggregation

tic point of the aggregation. If the distance to any characteristic point decreases steadily, then it can be assumed that the single trajectory does not leave the trajectory aggregation. This can be checked for a threshold in points or meters.

5.3 Differential Privacy for Trajectories

For relational data, differential privacy[24] is an accepted approach to anonymize data. For trajectory data, the focus is on a coarse-grained level[18] and not on a detailed level which looks at the exact course of the road (see also related work in Section 2.1.3). As already discussed in sections 2.2.3 and 2.2.4, in a centralized client-server approach, the server might be able to recalculate changes that were sent by the client and therefore reconstruct the single trajectory that was sent. To prevent this, the client can obfuscate the single trajectory by modifying the sequence of subtrajectories that are merged with the trajectory aggregation. A differential privacy approach can help to quantify the degree of modification or randomization.

5.4 Conclusion

This chapter discussed anonymization methods for location and trajectory data. k-Anonymity is an appropriate metric for location and trajectory data. Furthermore, it is suitable for a subsequent map construction because a generalization is necessary for achieving k-Anonymity but also for map construction. Conclusively, the methods used in this work do not resolve a conflict but provide a solution for two problems. While it is straight-forward to match points and edges which fulfill a certain k the matching of subtrajectories is more complex to ensure the overall fulfillment of k. The additional steps to be taken into account are described in this chapter. Differential privacy for trajectory data

seems not suitable (no approach available) for fine-grained trajectory data. It can, however, be used within the global setup of the anonymization process.

Chapter 6

Evaluation

In this chapter the methods for matching and merging are evaluated. Synthetic cases and demo scenarios are chosen which represent best the requirements of a trajectory aggregation. For both evaluation scenarios the aggregation algorithms are evaluated and if necessary optimized. The aggregations were calculated using Java 1.7 on Ubuntu 14.04 LTS with 16 GB memory and an Intel® Core™ i7-4600U CPU @ 2.10GHz quad core processor.

This chapter is organized as follows: Section 6.1 discusses the evaluation of the synthetic cases and Section 6.2 the evaluation of the demo scenarios. Both sections include an introduction to the cases, an evaluation with the edge-based approach and an evaluation with the trajectory-based approach. Section 6.3 concludes this chapter.

6.1 Synthetic Cases

Synthetic Cases are used to evaluate the behavior of an algorithm under ideal circumstances. For synthetic cases one has a clear expectation what the result may be, depending on the algorithm. One has the possibility to create cases where the result can determine the underlying algorithm or at least the type of underlying algorithm. Synthetic cases are also used to determine the correctness of an algorithm. In this case a certain result is expected and an algorithm which creates the expected result is correct while an algorithm, which does not, is incorrect.

6.1.1 Introduction to the Cases

The synthetic cases are created to check for certain properties of an algorithm for matching and merging. They differ intentionally from cases in the real

world to test properties which cannot be tested in real world examples. They include only one track for the aggregation and one track for the individual to be independent from effects which might appear in repeated aggregation.

Figure 6.1 shows all synthetic cases. Figure 6.1a shows two converging and diverging traces. This case is created to test if a simple matching and merging works. All algorithms should be able to match the inner parts and not the outer parts, depending on the threshold of the distance which is used. There should not be any gaps within the match and the merging should only create merged parts which are amidst both traces. Increasing the threshold should result in the same or a higher amount of matchings and decreasing the threshold should result in the same or a lower amount of matchings. A threshold of 0 should lead to no matchings and a threshold of infinity or the maximum value of a data type should lead to a complete matching set (for edge-based algorithms every edge of one trace with every edge of the other trace).

Figure 6.1b shows a similar scenario but this time the tracks are diverging and converging two times. This scenario can evaluate that the algorithm is able to match and merge multiple parts of the trajectory independent from each other. The algorithm should be able to create exactly two blocks which are matched and merged. Moreover, all the properties tested in the previous case (Figure 6.1a) are also demanded in this scenario. Nevertheless, it is a good distinction to evaluate these properties also independently from the property of matching at multiple parts.

Figure 6.1c shows a scenario where one trace is shorter than the other while the shorter trace is very near and with a normal value for the threshold for the distance there should be matchings along the complete length of the shorter trace. This scenario can evaluate the ability of the algorithm to merge without gaps along the matching and to connect the merged part properly to the not matched part and in this way show a growth of the aggregation along a certain path.

Figure 6.1d shows a scenario where two traces cross each other multiple times due to a possible imprecision of the GPS signal. This scenario can evaluate the ability of the algorithm to create an aggregation which averages the two traces and by averaging creates an aggregation which is more stable than each of the input traces. A good aggregation should have the following properties. The aggregation should be shorter than the input traces. This expresses a more stable aggregation. Additionally, most (at least half) of the parts of the aggregation should be inside the area spanned with the two input traces. The shorter the aggregation and the more parts of the aggregation are within the spanned area the better the evaluated properties are fulfilled.

Figure 6.1e shows a scenario similar to the first scenario (Figure 6.1a). Nev-

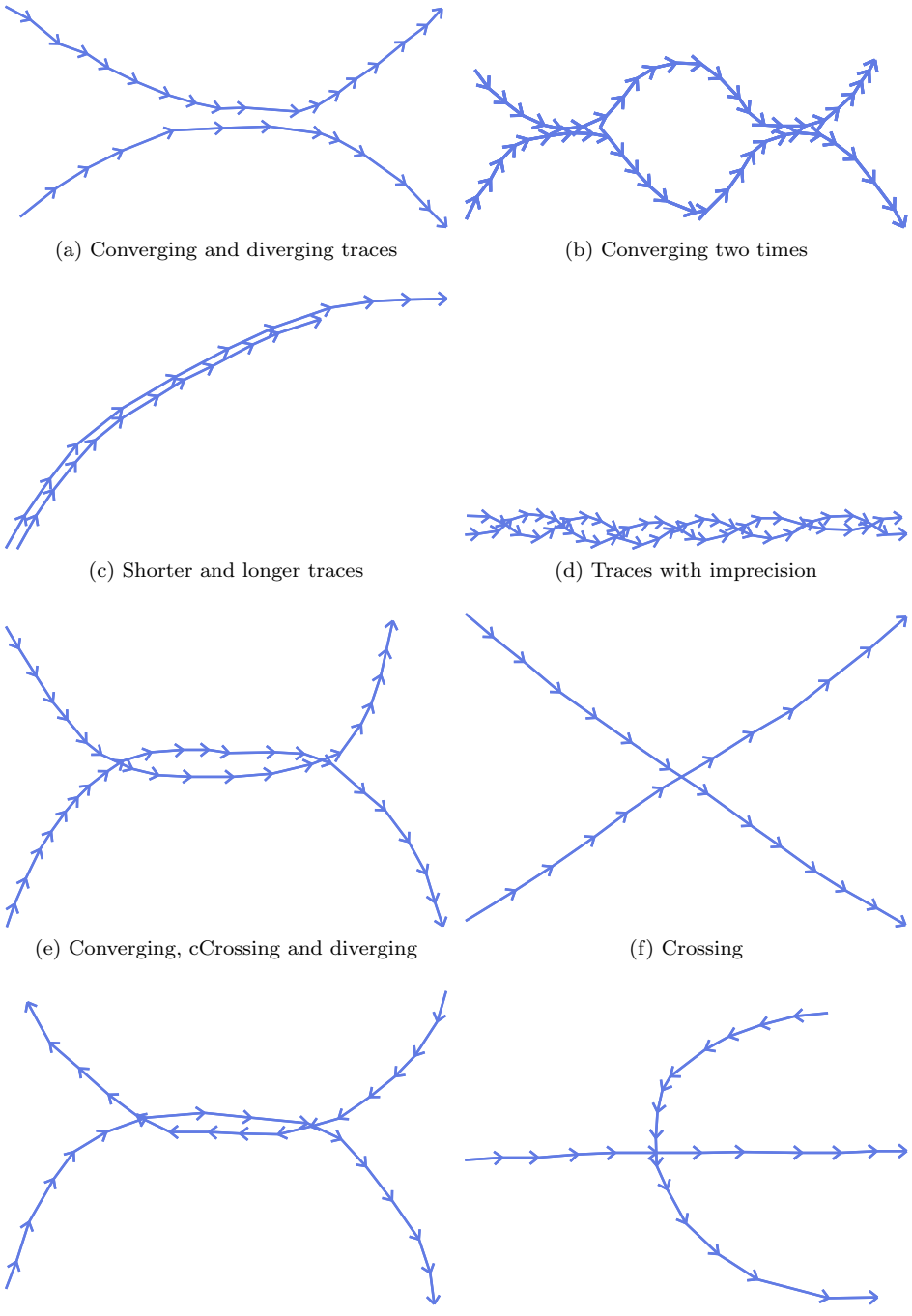
ertheless, in this scenario the traces cross each other after the converging of the trajectories. The traces cross a second time before they converge. The expected result should also be similar to the first scenario and also the properties should hold. Additionally, an optional quality criteria is the position of the start and end of the merged part of the aggregation. The start of the merged part should be before the first crossing and the end of the merged part after the second crossing. Nevertheless, a strong influence of the angle within the distance measure can prevent this property. In this case one cannot state a lower quality based on a non expression of this property.

Figure 6.1f shows a crossing. Two traces cross each other without any indication of having a two streets crossing each other. In a real world example this can be two independent streets (separated by elevation with a bridge or a tunnel) or it can be two crossing streets. Based on only these two trajectories an algorithm should not merge the trajectories. Nevertheless, this is strongly dependent on the threshold of the distance and the influence of the angle in the distance measure. Hence, the property can only be formulated in a way that there should be a distance threshold where there are no matchings. A stronger property can only be formulated taking another case into account. For example a property might be that there exists a distance threshold where there are matchings for the scenario in Figure 6.1a but not for the scenario in Figure 6.1f.

Figure 6.1g shows two traces which converge, cross and diverge. Nevertheless, in this case the traces are in different directions and should not be matched and merged. Again, the dependency to the distance threshold and the influence of the angle is very high and a property can only be formulated in a way as it is already described for Figure 6.1f. The same accounts for Figure 6.1h which shows a scenario of two traces which cross each other while one trace is expressed as a curve. The final distance of the traces is high enough that it should not be confused with a trace merging into another as it occurs in a highway drive-up.

6.1.2 Results with Edge-Based Algorithm

For the evaluation an edge-based matching and merging algorithm is used. For the distance calculation the distance along the perpendicular, a point-to-point distance and the angle difference are taken into account. If the two edges have actual crossings that can be created along some perpendicular (actual crossing means that the crossing is with the boundaries of an edge) then the distance between the point where the perpendicular starts and this crossing is used for further calculation (see also Figure 3.1). The perpendiculars starting on the start and end points of each edge are taken into account. If there is no actual crossing then the higher distance of the distances between the start points or the



(a) Converging and diverging traces

(b) Converging two times

(c) Shorter and longer traces

(d) Traces with imprecision

(e) Converging, cCrossing and diverging

(f) Crossing

(g) Converging, crossing and diverging in opposite direction

(h) Crossing with a curve

Figure 6.1: Synthetic cases of GPS traces for aggregation

end points is used instead of the perpendicular distance. The resulting distance is then combined with the angle difference of the two edges. The resulting distance is first normalized by the averaged length of both edges to fit the angle difference which does not depend on the length of the edges. Afterwards, the angle is exponentiated by an exponent which can be given (here the default 2 is used) and then weighted with a weighting factor which also considers that the scales cannot be directly compared to each other (metrical scale vs. degrees). In this case the weighting factor 0.00002 is used. If matched (distance below threshold) the matches are stored as pairs of the edge of the aggregation and the edge of the individual trajectory including their weighted distance. The matches are stored in blocks where each block includes only subsequent matches.

In the merging step these blocks are iterated. For each match the aggregation is adjusted along its perpendicular towards the individual trajectory (see Figure 4.6). Therefore, only actual crossings (see above) are taken into account. This prevents a double movement of one node of the aggregation as well as it prevents movement of nodes of the aggregation which are more at the border of the matchings (which is a more secure merging in terms of preventing creating strong angle differences within the aggregation).

If there were parts of the individual trajectory before or after the matched blocks then these parts are connected to the nodes of the aggregation which were moved in the merging before. Therefore, the nodes in the matched block are ordered based on their connection to each other. Afterwards the non-matched part of the individual trajectory before the matched block is connected to the first node of the matched block and according to this the non-matched part of the individual trajectory after the matched block is connected to the last node of the matched block.

The algorithm used for this evaluation also takes finalization steps into account (see Figure 4.11).

Figure 6.2 shows the resulting aggregation using the described algorithm. The input traces (blue) are still partly visible, but in the background. The resulting aggregation is black, the edged which were actually used for merging are highlighted red and the perpendiculars which were used for the movement are shown in yellow. The more trajectories determined the edge of the aggregation, the thicker the edge is painted. In all cases, except for Figure 6.2b, the thickness is either 1 or 2. All scenarios in Figure 6.2 are in the same order as they are in Figure 6.1.

Figure 6.2a shows the aggregation of two converging and diverging traces. The result fulfills all the criteria that are demanded by this scenario. The inner parts are matched while the outer parts are not matched (the matched parts are not directly shown, but one can conclude that all visible input traces which did

not disappear in behind in the background were matched). Furthermore, the merged part of the aggregation is completely amidst the two input trajectories and the connections to the merged part are correctly identified.

Figure 6.2b shows the result of a similar scenario. Nevertheless, this scenario is doubled in two points of view. First, the converging and diverging takes places two times. And second, the input traces are two for each track (which is not visible). In the result you can see that the aggregation has three different thickness levels which represent different weights. The merged part amidst the two tracks (merged with 4 trajectories in total) has the highest thickness. Furthermore, all the other parts, except one edge, have a thickness which represents a weight of 2. Altogether, the result fulfills all the criteria which are demanded. Particularly, the criteria of two separated inner mergings is fulfilled. Nevertheless, one has to note the one edge which keeps the weight of 1 although two 2 trajectories is not merged. The reason behind this is the change of the angle due to the first merge. The second trajectory is not merged because the edge was moved in a way such that it cannot find an actual crossing of the perpendicular any more and thus this edge is not moved towards the individual trajectory. This behavior cannot be finally evaluated as good or bad behavior because judging just on the basis of the created track in the aggregation the result looks convincing if one wants to create a road network afterwards.

Figure 6.2c shows the aggregation of two trajectories which are on the same track and where one trajectory is shorter than the other. The edges were matched without gaps and the connection to non-matched parts looks correct. In this case one can observe the role of the ordering with which the trajectories are inserted in the aggregation. Here, the trajectory above was inserted first. This is the reason that as a result of the merging the start node and the node just after the end of the trajectory below are not moved. The perpendicular of the starting node finds no actual crossing on an edge of the trajectory below and neither finds the node just after the end of trajectory below.

Figure 6.2d shows the aggregation of two imprecise tracks. The criteria that most parts of the resulting aggregation are between the two input trajectories is fulfilled, judging visually the amount should be over 95%. Here again, one can see that it does matter in which order the trajectories are inserted. The trajectory starting above and ending below was inserted first and therefore the first track in the aggregation. This is the reason that the non-merged part (start and end node of the first track in the aggregation) are used to connect the merged part with. The trajectory inserted after is not connected because all parts were matched (difference between matched and merged parts). This behavior has to be considered good because the assumption is reasonable that if all parts were matched then there is only one track.

Figure 6.2e shows the aggregation of converging, crossing and diverging trajectories. The result fulfills all demanded criteria. Additionally, the start of the merged part is before the first crossing and the end of the merged part is after the second crossing. the condition that the merged part is completely amidst the two trajectories is not necessarily fulfilled although one cannot judge otherwise just by looking at the result. It is possible that parts that are near the crossing are not precisely amidst. Nevertheless, this is not a bad indication since one has to look at all the properties the algorithm looks for in the matching and merging process. One can always reason that these properties are more important than the fulfillment that the merged part is completely within both trajectories.

Figure 6.2f shows the aggregation of two crossing trajectories. Here, the main criteria that there is no matching does not hold for the threshold settings. One can see that the two trajectories are matched and merged with each other. As already described in the introduction of the cases, this is not a wanted behavior because there is a good possibility that these trajectories are independent from each other (tunnel and bridge for example). This case has the same settings for the distance threshold as the other cases. To change the behavior of this case one has to change the threshold (or thresholds) for the distance. There are basically four possibilities to change the unwanted behavior:

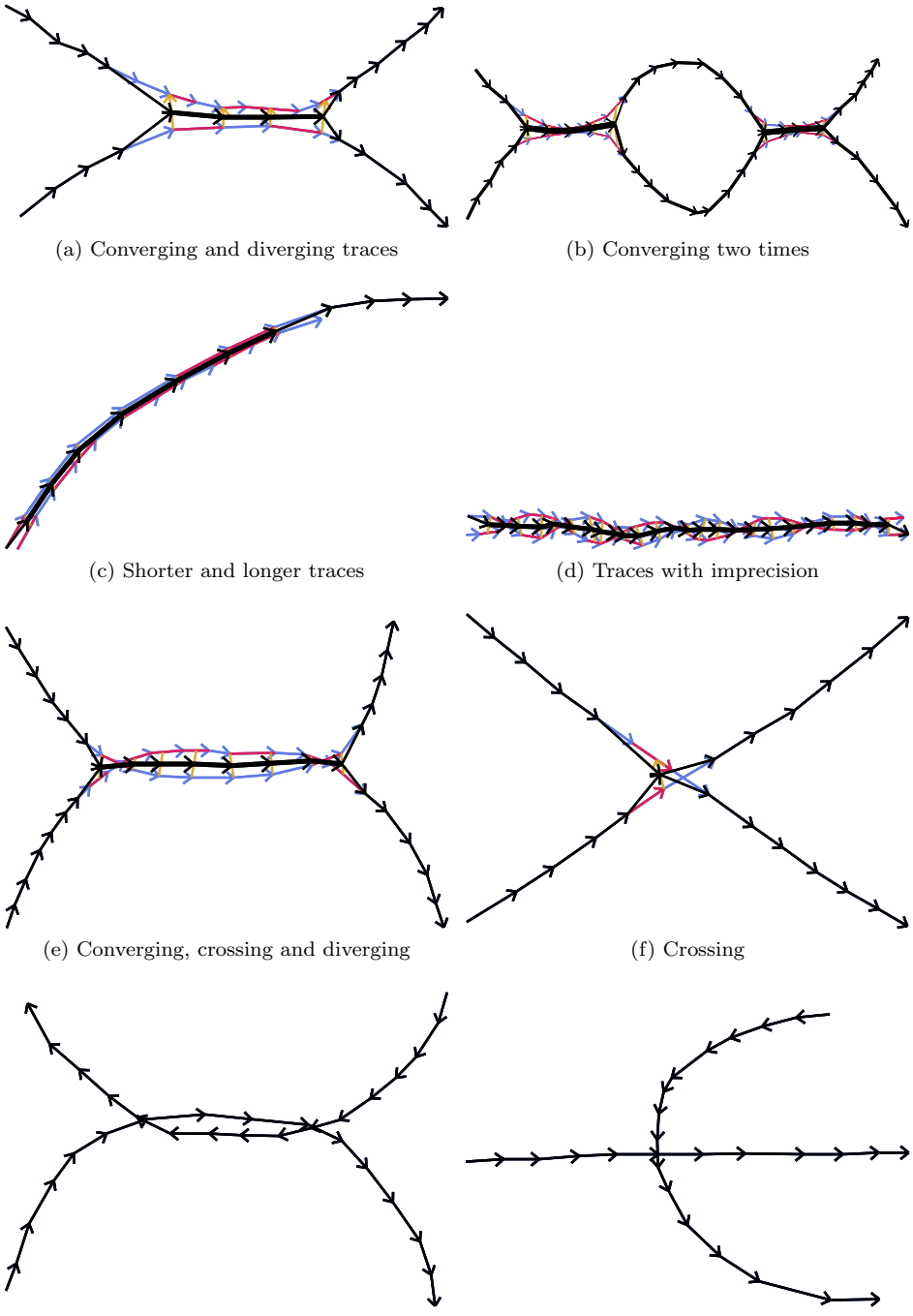
1. A decrease of the distance threshold (distance is the resulting distance include normal distance and angle distance components) until there are no more matchings within this scenario. Nevertheless, this also strongly influences the result of the other scenarios and maybe prevent matchings which are wanted.

2. An increase of the exponent of the angle difference increases the exponential behavior of the weight which penalizes a higher angle difference even more. It is likely that this change can have a good influence on this scenario while preventing bad behavior in other scenarios.

3. An increase of the weight of the angle difference in comparison to the other distance elements, particularly the distance with the use of the perpendicular. This leads to a higher influence of the angle difference. It is also likely that this can eliminate the matchings in this scenario and still keep wanted matchings in other scenarios.

4. The introduction of an upper limit of the angle difference itself can also have a positive effect.

Figures 6.2g and 6.2h show that there are no trajectories matched and merged which is the demanded criteria for these scenarios.



(a) Converging and diverging traces

(b) Converging two times

(c) Shorter and longer traces

(d) Traces with imprecision

(e) Converging, crossing and diverging

(f) Crossing

(g) Converging, crossing and diverging in opposite direction

(h) Crossing with a curve

Figure 6.2: Synthetic cases of GPS trace aggregation with edge-based matching and merging

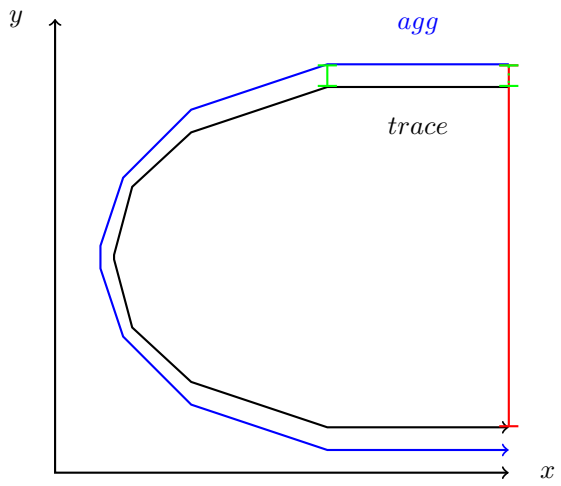


Figure 6.3: Two possible crossings with the perpendicular

6.1.3 Results with Subtrajectory-Based Algorithm

The subtrajectory-based algorithm which is used for the evaluation uses the Fréchet distance as distance metric with 3 dimensions: x, y and the angle between edges. For the following test cases, the L-Infinity implementation of the Fréchet distance is used (see Section 3.3). The angle is weighted to fit to the metric or Cartesian values. To find complete blocks for merging the iterative matching shown in Figure 3.7 is used. The actual merging is comparable to the merging approach of the edge-based algorithm used for the evaluation. Analogous, the nodes of the trajectory aggregation are moved towards the crossings of the perpendiculars. Nevertheless, in this case there is no underlying single match of edges of input trajectory and trajectory aggregation. This is the reason why in the merging process the crossings are searched within the matched blocks with each edge of the input trajectory and each edge of the trajectory aggregation. Figure 6.3 shows an example where there are two possible crossings of the perpendicular outgoing from the starting node of the trajectory aggregation. The dotted green perpendicular leads to a near crossing which is favorable, but the red perpendicular shows a crossing far away on a part of the input trajectory which is far away. It should be noted that the complete input trajectory is matched as one matched block with the complete trajectory aggregation in this case. Because of this effect a threshold for the length of the perpendicular is introduced.

Figure 6.5 shows the results of the subtrajectory-based algorithm on the synthetic scenarios. The results are comparable to the results with the edge-

based algorithm. Nevertheless, there are some differences. One has to keep in mind that the thresholds are not directly comparable and many differences can be explained by different thresholds.

The aggregation shown in Figure 6.5a has less matched edges of the trajectory aggregation (below) than the one shown in Figure 6.2a. This fact alone can be explained with a sharper threshold. Nevertheless, one can also observe a higher number of matches of the input trajectory (above) because the created connections connect smaller parts of the input trajectory. There is an explanation which is typical for the iterative matching. Figure 6.4 shows different matching results which can result from the iterative matching approach. Every result is created with the same distance. Conclusively, the result is highly dependent on the expansion in the iterative process:

1. An expansion which starts at the beginning of the trajectory aggregation and expands first on the input trajectory will lead to a result shown in Figure 6.4a.

2. An expansion which starts at the beginning of the trajectory aggregation and expands first on the trajectory aggregation will lead to a result shown in Figure 6.4b.

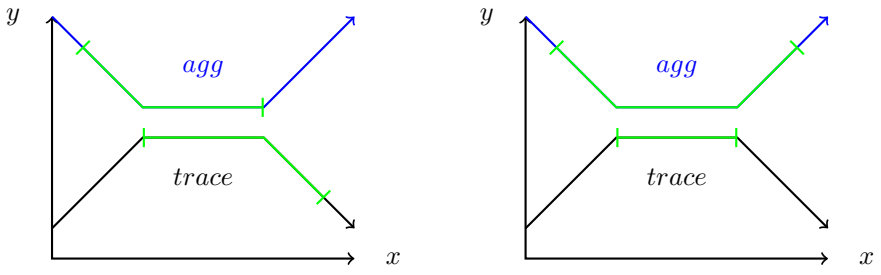
3. An expansion which starts at the beginning of the input trajectory and expands first on the input trajectory will lead to a result shown in Figure 6.4c.

4. An expansion which starts at the beginning of the input trajectory and expands first on the trajectory aggregation will lead to a result shown in Figure 6.4d.

The result on display in Figure 6.5a is mostly comparable to the result shown in either Figure 6.4b or Figure 6.4b depending on which trajectory is seen as the input trajectory and which as the trajectory aggregation. The same behavior of the iterative matching approach applies to Figure 6.5b.

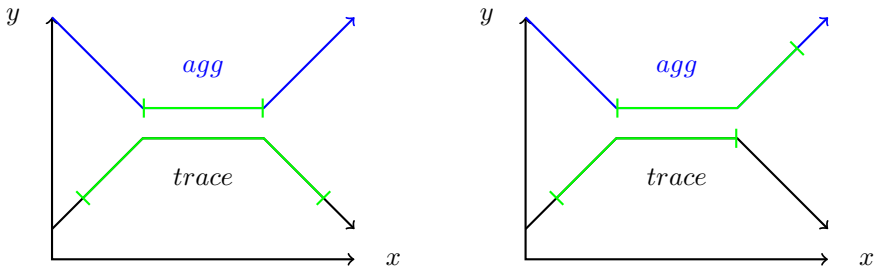
Figures 6.5c, 6.5d, 6.5g, and 6.5h show results with no identifiable qualitative difference to the results of the edge-based algorithm.

Figure 6.5e shows a result where there are again comparably smaller parts of the input trajectory (starting and ending below) which are not matched than of the trajectory aggregation (starting and ending above). In this case this also leads to a direction change (shortly before the created connection) in the resulting trajectory aggregation which looks imprecise and lowers the fulfillment of the demanded criteria in this scenario to create a resulting trajectory which lies to a high amount inside the area which is enclosed by the two trajectories crossing each other.



(a) Iterative matching result 1

(b) Iterative matching result 2



(c) Iterative matching result 3

(d) Iterative matching result 4

Figure 6.4: Possible results with iterative matching and a subtrajectory distance

The result shown in Figure 6.5f fulfills the criteria of this scenario while the result shown in Figure 6.2f does not. Nevertheless, this can be explained only taking distance threshold into account. Furthermore, the thresholds can be adjusted in a way that the edge-based algorithm fulfills these criteria, too.

6.2 Demo Scenarios

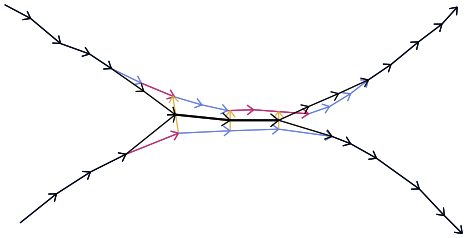
Demo scenarios are real-world scenarios with trajectories captured with the help of positioning systems (most popular: GPS). Real-world scenarios test an algorithm for practical use. GPS traces are recorded in a more uncontrolled manner than the synthetic cases are created. One has to take into account GPS imprecision and maybe add cleaning steps before the aggregation of the trajectories (see Section 2.2.6). For real-world scenarios the expected result is that the scenario is correctly represented in the aggregation. The aggregation should represent averaged tracks. It should not induce errors and for most of the input trajectories one should be able to say that the aggregation is a better representation for the actual track than the input trajectory.

6.2.1 Introduction to the Cases

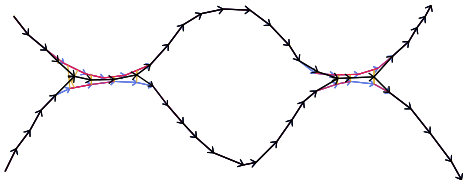
To evaluate the practical use the real-world scenarios depict typical scenes from a street network. All input trajectories are taken from the GPS trace collection of OpenStreetMap and were tagged as recorded while driving in a car. There is no information about the record frequency or the GPS signal quality. Nevertheless, all trajectories which obviously contained wrong information were removed manually. All scenarios are near to or within the borders of the city Berlin, Germany. Table 6.1 contains information about the locations of the scenarios.

Figure 6.6 shows the input trajectories for the real-world scenarios. Compared to the synthetic cases the real-world scenarios have more input trajectories and the record frequency and quality varies. Often one can detect the main roads just by the number of trajectories.

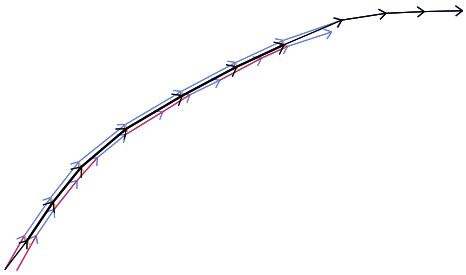
Figure 6.6a shows a street crossing. One challenge in this scenario is to find all the possible connections of the crossing but to leave out the impossible connection, particularly the connection which are not represented in the trajectories. From right to left there is a one-way street so there should be no connection from the right lane coming from the bottom to the right part of the one-way street and no connection from the left lane coming from the top to the right part. For the left part of the one-way street vice versa. All other connections that are possible are also represented by the GPS traces. Another challenge is an imprecision of the data. One can see a little gap at the bot-



(a) Converging and diverging traces

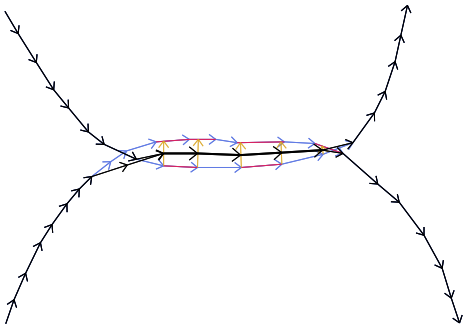


(b) Converging two times

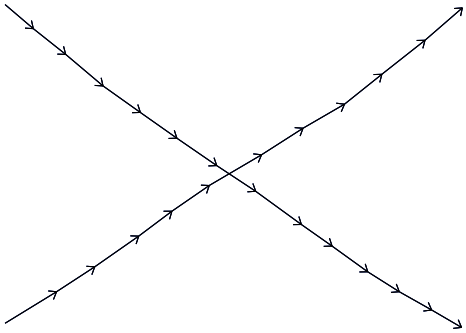


(c) Shorter and longer traces

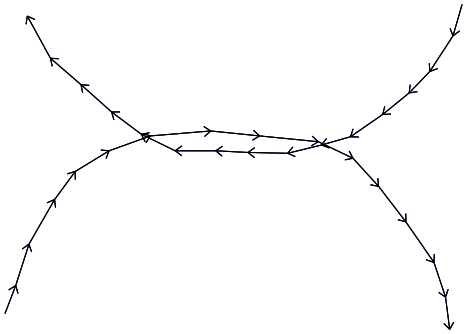
(d) Traces with imprecision



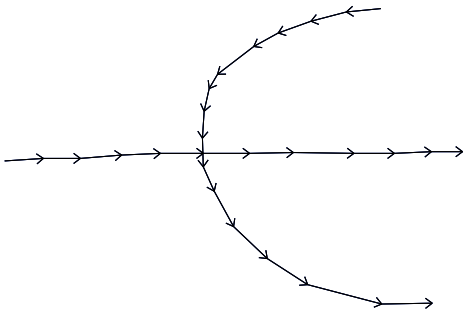
(e) Converging, crossing and diverging



(f) Crossing



(g) Converging, crossing and diverging in opposite direction



(h) Crossing with a curve

Figure 6.5: Synthetic cases of GPS trace aggregation with subtrajectory-based matching and merging

case	longitude	latitude	description
1	13.3831663 13.390314	52.4395487 52.4414339	crossing at station Alt-Mariendorf
2	13.3599312 13.3637779	52.4985748 52.5006771	crossing at station Kurfürstenstraße (traces only represent a T-crossing)
3	13.2419335 13.2440909	52.6457982 52.6480122	highway drive-up on A111 at Hohen Neuendorf
4	13.2408021 13.2421745	52.6365708 52.6375575	highway departure on A111 at Hohen Neuendorf
5	13.3480869 13.3511905	52.5131064 52.5146227	part of a circle at Siegessäule, Tiergarten
6	13.2808814 13.2846144	52.5052292 52.5076504	a highway below a street at Messe Nord, ICC
7	13.4252746 13.4314731	52.5165846 52.5206277	a circle at Strausberger Platz
8	13.2932016 13.2969084	52.5043006 52.5066953	a place amidst two one-way streets at Amtsgericht Charlottenburg

Table 6.1: Location information for real-world scenarios

tom between the two lanes. Nevertheless, due to GPS imprecision not all the trajectories are on the correct lane in the right direction. Here, especially the distance threshold is important to avoid creating multiple lanes in one way.

Figure 6.6b shows the input trajectories of a T-crossing. The challenges are similar to the previous scenario. Even though it is likely that this connection exists, there are no traces connecting the top with the left. Hence, the algorithm should not create these. Nevertheless, there are connections from the left to the bottom in both directions. Both streets have lanes in to directions and again the trajectories are not always on the correct lane due to GPS imprecision.

Figure 6.6c shows the input trajectories of a highway drive-up. In this scenario there is only one connection which should be identified. While the drive-up has only one direction, the highway has two. Again, not all trajectories are clearly distinguishable by their position in each direction.

Figure 6.6d shows the input trajectories of a departure from a highway. The challenges for the aggregation algorithm are the same as for the highway drive-up.

Figure 6.6e shows the input trajectories of a part of a circle. There are two tracks (and two trajectories) entering the circle and one track (two trajectories) leaving the circle. In this scenario there are multiple connections in multiple areas which should be detected. This scenario can also be seen as a composition of the previous scenarios. Therefore, it also contains the same challenges. Nevertheless, it also tests the behavior of the algorithm aggregating a composition

of multiple different connections in different places.

The input trajectories shown in Figure 6.6f represent a two-way street (from left to right) over a highway. This scenario is the real-world version of the synthetic scenarios shown in the Figures 6.2f and 6.2h. Additionally to the synthetic scenarios, this scenario includes multiple directions and a realistic count of input trajectories. The expected result is that the trajectories are merged in lanes going in different directions and, more important, that there is no connection detected.

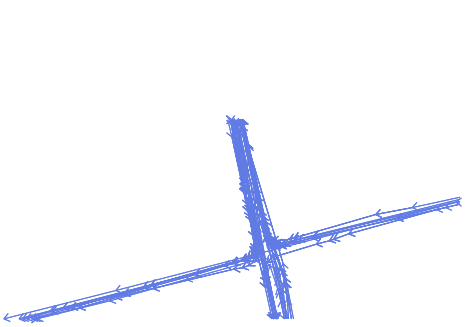
Figure 6.6g shows the input trajectories of a circle traffic. Compared to Figure 6.6e it shows a smaller but complete circle traffic with more trajectories to aggregate. All the possible connections that are possible in this standard circle traffic are represented by the trajectories. Additionally, due to the higher number of trajectories this scenario has also a higher variation of frequencies and qualities. The expected result is that all represented connections are created only once and no additional incorrect connections are created.

Figure 6.6h shows the input trajectories of a place which is enclosed by two one-way streets. This can be seen as a rectangular circle traffic with only two connected roads. Additionally to the circle scenarios, this scenario is a test for a different shape with basically the same function.

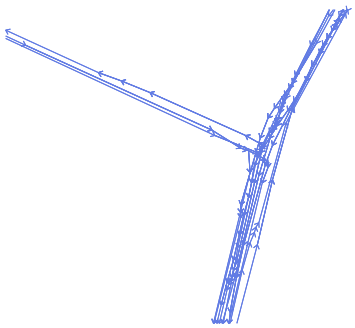
6.2.2 Results with Edge-Based Algorithm

The edge-based algorithm for matching and merging is the same as described in Section 6.1.2. Additionally, for the evaluation of the real-world scenarios in combination with the edge-based algorithm a cleaning process is used before to clean the input trajectories (see Section 2.2.6). In the cleaning process the edges of the input trajectories are normalized, edges below 0.3 meters are filtered out by omitting points and edges above 13.0 meters are partitioned in edges of equal length by the factor that the edge is longer than the threshold of 13.0 meters. The points that would create edges creating a heading change with an angle of more than 30.0 degrees are also omitted. Furthermore, these cleaning thresholds as well as the distance thresholds are varied in different scenarios to improve the result. Additionally to the cleaning of the GPS trajectories, the finalization, particularly the connecting of parts of the aggregation, is important for the real-world scenarios (see Section 4.4).

Figure 6.7 shows the results of the aggregation. The results of the cleaning step are not shown, but are also an intermediate result. The only visual difference to the raw input trajectories is the partitioning of the longer edges. This can be visually estimated by looking at the resulting aggregation. The length of the edges of the aggregation is comparable to the length of the edges of the

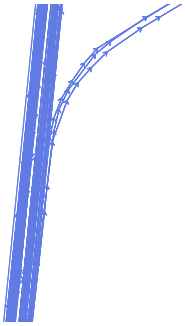


(a) Street crossing

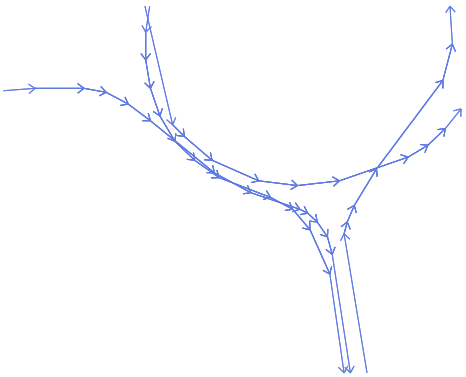


(b) T-crossing

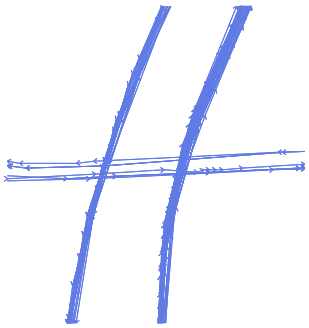
(c) Highway drive up



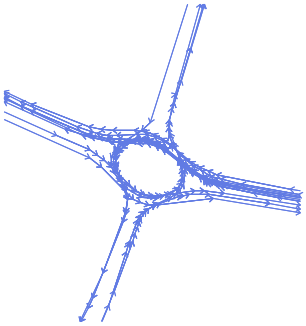
(d) Highway departure



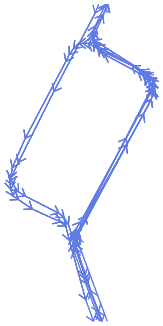
(e) Entering and leaving of a circle



(f) Highway below street



(g) Circle traffic



(h) A place amid lanes

Figure 6.6: Real cases (in Berlin) of GPS trace aggregation

cleaned trajectories. The coloring of the different shown parts is described in Section 6.1.2. Overall, the real-world cases have in most cases a thicker line because the thickness of the line represents the weight of the aggregation and in the real-world scenarios the number of input trajectories is higher which increases the weight of the aggregation.

Figure 6.7a shows the aggregation of the crossing scenario. All connections which are represented by the input trajectories can be found in the aggregation. The connection from right to bottom is represented, too. Nevertheless, this connection has moved significantly to the top. The reasons for this strong movement are on the one hand the distance threshold which still allows a matching between the input trajectory which is more at the bottom when changing the direction and the trajectory aggregation more above and on the other hand the rules for connecting which are shown in Section 4.4, particularly the rule to merge previous parts keeps the connection as compact as possible. Another issue which seems odd at first sight is the starting node of the aggregated trajectory coming from the bottom. There exist trajectories which start directly at the bottom, but this is not represented in the aggregation. There are again two reasons behind this. One reason is again the distance threshold which allows a matching of the parts below with the start of the aggregation which prevents a connection of parts which cannot be matched. The other reason is the iterative approach itself. The first trajectory on this track which was added to the trajectory aggregation did not start directly at the bottom but more or less were the resulting aggregation still starts. And because everything below was still matched there was no part that was identified for adding to the aggregation.

The aggregation of the T-crossing scenario is on display in Figure 6.7b. In this scenario every connection was found properly. There is one turn with the street coming from top left and one turn from top left to the bottom. The other trajectories are just straightly following their path. The trajectories coming from top right going to the bottom were all merged together and also the ones which were near or on the other lane were correctly identified as a match because of the same direction. The scenario also shows the same phenomenon that the aggregation going to the bottom seems cropped. The same explanation as before holds. The first trajectory in the aggregation on this track stopped more or less where the resulting aggregation stops. And there was no part added because all parts below could be matched to the last part of the existing aggregation.

Figure 6.7c shows the aggregation of the highway drive-up. In this case there is only one demanded connection (the one onto the highway) which can also be found in the trajectory aggregation. The aggregation represents two independent tracks where one is connected with another track. The average which was calculated for the aggregation also has the impression that it represents more or

less the middle of the real street. The angle of the drive-up is realistic although this is not necessary for the construction of a road network afterwards. More important is that no additional connections were found which are not represented by the input trajectories which is ensured by the rules described in Section 4.4 and applied to the real-world scenarios. The part of the highway going to the top a final edge which has a lower weight and a deviated angle. The explanation is similar to the missing parts in the previous scenarios. The first part added on this track to the aggregation was more right than then the average of all added parts. After the insertion of the first trajectory on this track the other parallel parts were matched, but all were merged with the previous edge which had a better fitting angle and therefore the lower distance. Altogether, the slight deviation to the real average at the borders of the scenario are caused by the iterative nature of the overall algorithm, the order with which the trajectories are inserted and do not occur when extending the scenario throughout these borders.

Figure 6.7d shows the aggregation of the departure of a highway. This case is similar to the previous case. The only difference is the direction of the connection from (before: to) the highway. Nevertheless, the result of the connection of the departure of the highway does not look as smooth in comparison to the previous scenario. Still, there is an explanation and there is no bad effect for the creation of the road network afterwards. Again, this phenomenon is also partly dependent on the iterative aggregation of the input trajectories. Here, the connection was made relatively early in the process (within the first 25% of the input trajectories). After the connection was created there were many trajectories on the right lane of the highway which were added to the aggregation, but only a few (one or two) which were on the departure of the highway. Due to the merging, the aggregation was then moved more to the left because of the high number of trajectories on the highway and the connection was moved, too.

The aggregation of a part of a circle traffic is shown in Figure 6.7e. This is a more complex scenario, looking at the number of connections which the input trajectories represent. The amount of input trajectories is lower than in other scenarios. Nevertheless, different input trajectories are better to see and the actions of the aggregation algorithm can be followed more easily. The connections which are represented in the input trajectories are also created in the trajectory aggregation and no connections are made which are not represented.

Figure 6.7f shows the aggregation of the scenario with a highway which is below a normal street. There is a gap between the two directions of the highway, going from top to bottom. In this scenario no connections should be made because every direction is independent from the other directions and there are no turns possible. The resulting aggregation shows that there were no

connections created.

Figure 6.7g shows the aggregation of the circle traffic. In comparison to the scenario shown in Figure 6.7e, this scenario represents the complete circle traffic and includes more input trajectories. Nevertheless, the requirements are similar and the qualitative result is the same. All connections are found and no additional connections are created. As in the other scenario, all directions are also clear and do not deviate from the input directions.

The aggregation of the place surrounded by two one-way street is shown in Figure 6.7h. There is only one connection represented in the input trajectories which has to be created within a finalization step which is the T-crossing at the top. This was created. The results for this scenario are again clear and contain all semantic informations as well as they represent all directions of the input trajectories and more or less do not deviate from them.

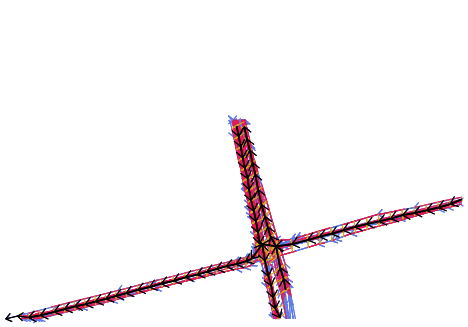
6.2.3 Results with Subtrajectory-Based Algorithm

The subtrajectory-based algorithm for the demo scenarios uses the same data cleaning methods as the edge-based algorithm for the demo scenarios, see description in sections 6.2.2 and 2.2.6. The algorithm is already summarized for the synthetic scenarios (see Section 6.1.3). Figure 6.8 shows the resulting trajectory aggregations using the subtrajectory-based algorithm. Altogether, the results are again very good to compare with the results of the edge-based algorithm. Nevertheless, there are some differences which can be pointed out.

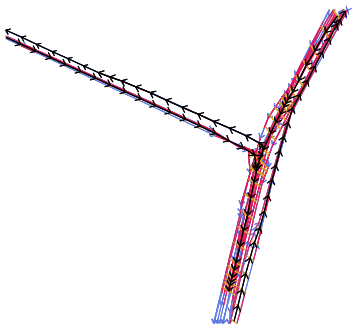
Figure 6.8a shows the resulting aggregation of the crossing scenario. Unfortunately, the connections cannot be judged only visually. Nevertheless, it appears that at least one connection is doubled or in the wrong direction. Still, a doubled connection can be detected afterwards by a road construction mechanism. Additionally, this is highly dependent on the thresholds which were set for the scenario. Overall, this result still fulfills the criteria for this scenario.

The result on display in Figure 6.8b shows the resulting aggregation of the T-crossing scenario. Qualitatively, this result is equal to the result of the edge-based algorithm in this scenario. The same can be said about the scenarios shown in Figures 6.8c, 6.8d and 6.8f.

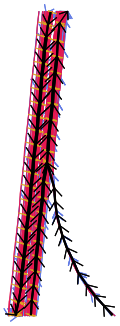
Figure 6.8e shows one difference between the result of the subtrajectory-based algorithm and the edge-based algorithm. The track leaving the circle to the middle bottom is merged based on two input trajectories in the result shown in Figure 6.7e, but is not merged in the result shown in Figure 6.8e with the subtrajectory-based algorithm. Figure 6.9 shows the challenge for the iterative merging process. It shows trajectory aggregation with a connection. Starting on the left a track can follow two path in the trajectory aggregation. While



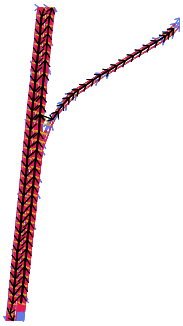
(a) Street crossing



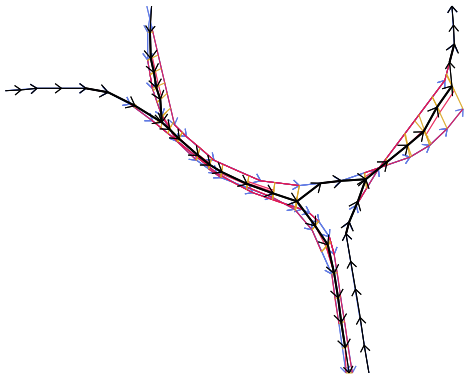
(b) T-crossing



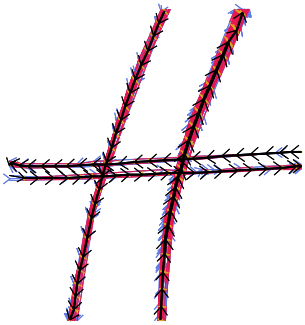
(c) Highway drive up



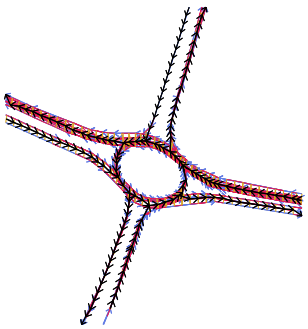
(d) Highway departure



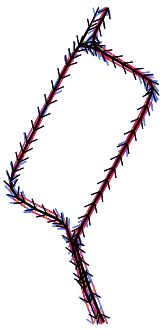
(e) Entering and leaving of a circle



(f) Highway below street



(g) Circle traffic



(h) A place amid lanes

Figure 6.7: Real cases (in Berlin) of GPS trace aggregation with edge-based matching and merging

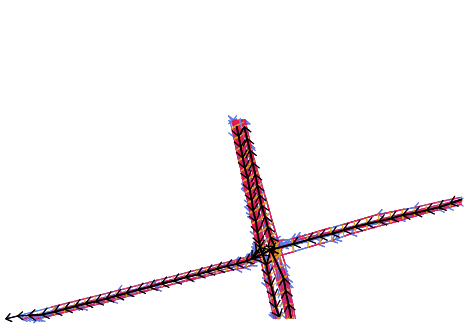
matching iteratively, the algorithm has to decide at the connection which path to expand. At the moment of the decision the complete matching is not available and the resulting distance comparing the both next nodes to each other has no relevant qualitative information. This is why the algorithm just decides for the path which is first in the iteration. Contrary to the challenge illustrated in Figure 6.4, this can be optimized. One way is to first extend on path completely until the distance threshold allows no more extension. Nevertheless, this also increases the algorithm's complexity and, dealing with a high number of input trajectories, this effect marginalizes because than the next input trajectory will be expanded on the more promising path.

The resulting trajectory aggregation on display in Figure 6.8g has all connections which are represented by the input trajectories. The course of the tracks represents the input trajectories well. Nevertheless, some parts near the connections are not influenced by as many input trajectories as the trajectory aggregation shown in Figure 6.7g and according to this their weight is lower. For example this can be seen on the left part of the circle (not the tracks towards and from). The cause is the same as described for Figure 6.8e.

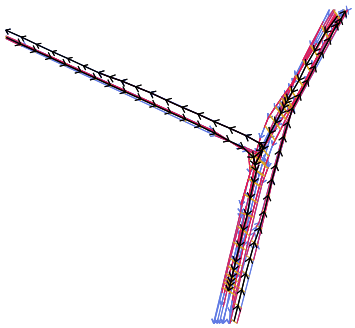
In Figure 6.8h one can see another effect of the same cause. The trajectory aggregation on display misses one connection which was created for the trajectory aggregation shown in Figure 6.7h. One can see the missing connection coming from the place going to the top of the scenario. The connection was not created because there were always matchings found at the existing track which prevented the expansion in the other direction.

6.3 Conclusion

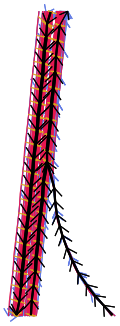
The evaluation showed the feasibility of the two tested approaches. The first approach uses an edge-based distance measure and an edge-based merging while the second approach uses a trajectory-based distance measure and a modified edge-based merging. Both approaches use data cleaning and finalization steps. Between the results of the two approaches there are no differences which can be directly linked to the fact that one uses an edge-based distance measure and the other one a trajectory-based distance measure. Both approaches are feasible to provide good aggregation results. The errors that were revealed during the evaluation could be fixed either by adjusting parameters for matching and merging or by adding rules to the finalization steps and were independent from the matching and merging approaches.



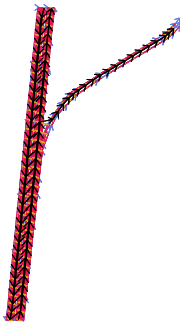
(a) Street crossing



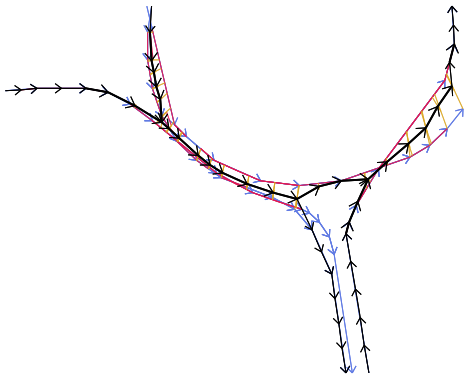
(b) T-crossing



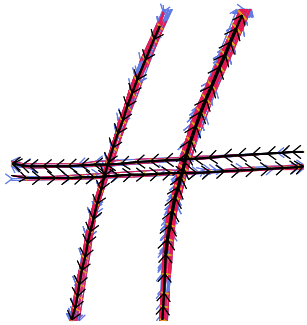
(c) Highway drive up



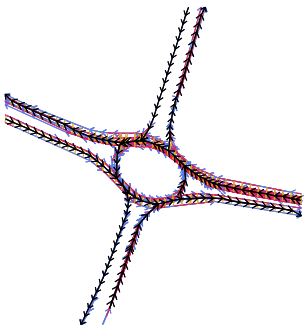
(d) Highway departure



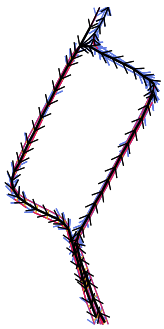
(e) Entering and leaving of a circle



(f) Highway below street



(g) Circle traffic



(h) A place amid lanes

Figure 6.8: Real cases (in Berlin) of GPS trace aggregation with subtrajectory-based matching and merging

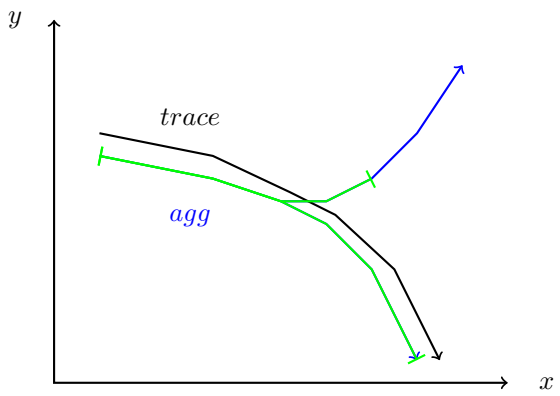


Figure 6.9: Expansion at connections

Chapter 7

Conclusion

In this thesis the anonymization of trajectories with the objective of creating a map afterwards is discussed. The thesis focuses on k-anonymity and preserving privacy through generalization. Chapter 2 introduced the related work and set the foundation for the following chapters including an architectural discussion of the software. In Chapter 3 different distances and processes to match trajectories are discussed. Simple distances need extensions to be promising for trajectory aggregation of a road network. With these extensions they seem as promising as more complex distances. Chapter 4 discussed different merging approaches for trajectories. To some merging approaches small extensions seem reasonable. However, none of the presented approaches can be directly excluded from evaluation because it is not promising. Privacy Preservation is discussed in Chapter 5. Due to the nature of the trajectory aggregation being a generalization approach, the appropriate metric for anonymization is k-anonymity. The evaluation in Chapter 6 showed good and appropriate results for both approaches which were evaluated. However, it cannot be said that each approach has distinctive properties which the other approach does not have.

An important new aspect concerns the creation of a road network. While on other works detailed information on the course of the road is omitted, this is preserved in this work. The main challenges of matching and merging are discussed in an exploratory manner. While first describing possible methods, the most promising ideas are used for the evaluation. The evaluation shows no significant qualitative difference between an edge-based and a subtrajectory-based algorithm.

The remaining of this chapter includes Section 7.1 for limitations, Section 7.2 for lessons learned, and Section 7.3 for future work.

7.1 Limitations

The software developed as part of this work creates knowledge which was not there before. The result of a trajectory aggregation is always correct in the sense that it was computed as designed. However, it is not important that the calculation is correct, it is important whether the result satisfies user expectations and whether it is proximate to the actual road network. Therefore, the conclusion is limited by the selected algorithms and the selected experiments. The observations made in the experiments can only show tendencies which might be proven wrong extending the space of the observation. The quality of the results is subject to the writer and cannot be proven. However, the selected evaluation approach is targeted to provide objective evidence of the feasibility of the evaluated approaches. It includes exhaustive synthetic cases and real-world examples which should cover a majority of the application scenarios.

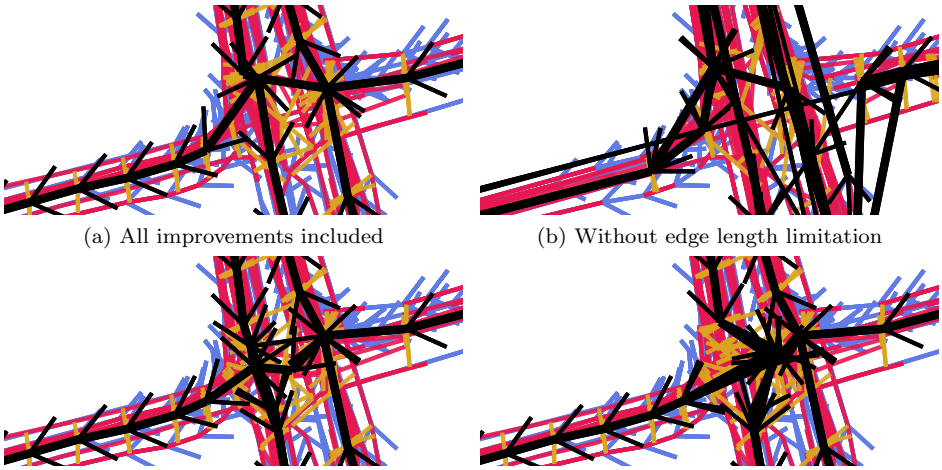
7.2 Lessons Learned

Because it is not possible to prove the quality of the results, subjective statements and tendencies are important. The original aim of the thesis was to compare different approaches to generalize trajectory data. With different approaches the main concerns were distances and merging methods. Nevertheless, during the work many small improvements were made which had a significant effect on the results. It was often the case that the result did not appear satisfying and according to a deficiency an improvement was made. Two tendencies were remarkable:

1. Stepwise improvements based on deficiencies of the results influenced the quality of the results much more than exchanging a distance or adjusting the distance threshold.

2. The more complex the distance and the merging approach the more complex the measures to fix errors in the results became.

Figure 7.1 shows different trajectory aggregations from the crossing scenario. The scenario has many input trajectories and many connections with a high area density (many connections per m^2). For all results the edge-based algorithm was used. While changing the distance has no significant effect on the results (compare Figure 6.7a with Figure 6.8a), omitting small improvements lead to a significantly worse result. Figure 7.1b shows a result without limiting and normalizing the edge length. Because of a high difference in edge length, also the distance increases. Resulting, one threshold does not fit multiple edge lengths any more and with the same threshold parts are added to the



(a) All improvements included

(b) Without edge length limitation

(c) Without merging of near connections

(d) Without checking for existing connections

Figure 7.1: Street crossing (in Berlin) GPS trace aggregation with edge-based matching and merging

aggregation which with normalization would have been matched and merged. Not only the total length of the aggregation increases, but also the number of connections which are made and which are not necessary. Figure 7.1c shows another trajectory aggregation with unnecessary connections. In this case the merging of near connections while checking for new connections to be made was omitted. Therefore, there are no means to remove connections and in some cases connections are created on a different path but with the same function. Here, one can see two connections coming from right and going to the left at the crossing. This can be detected afterwards. Nevertheless, detecting this in advance guarantees a steady improvement of the aggregation and prevents the provocation of errors. The same can be said for the result shown in Figure 7.1d where the check for existing connections is omitted. Therefore, every connection which can be detected according to the input trajectory is created even though there already exists a connection on a slightly different path. Already visually one can see the chaos which is created omitting this simple refinement.

The subtrajectory-based algorithm requires corrections additionally to the corrections which are required by the edge-based algorithm. Figure 7.2 shows the effect of omitting angle and length restrictions for the perpendiculars which are used in the merging process. In comparison to the edge-based algorithm, the subtrajectory-based algorithm matches complete blocks together and does not calculate distances between edges. Therefore, the complete matching block is processed in the merging step and a perpendicular is created for each edge of the matched part of the trajectory aggregation and each edge of the matched part

(a) All improvements included

(b) Without perpendicular angle and length restriction

Figure 7.2: Place (in Berlin) GPS trace aggregation with subtrajectory-based matching and merging

of the input trajectory. If no additional measures are taken then the resulting trajectory aggregation is the one on display in Figure 7.2b. To avoid these long mergings there have to be restrictions introduced which in a way replace the edge-based distance (angle and length).

7.3 Future Work

The major parts of the future work include extensions to this work. There are many extensions possible. The evaluation can be extended to more distance measure as well as more merging methods. Additionally, the finalization steps and cleaning methods can be changed, extended, and improved. For example, the thresholds for the finalization steps are currently static and independent values. However, it can be assumed that they are at least partially dependent on the values which are set as thresholds for the normalization. Searching for already existent connections, the number of nodes is used as threshold. Instead of the number the length in meters seems also plausible. The work can also be extended as a system. The system is currently an application working on a desktop system with graphical user interface. It would be beneficial to distribute the system on a server and multiple devices with different roles as described in Section 2.2 and to evaluate the performance within a real-world scenario.

To create a prototype that is able to process a huge amount of data the next step can be parallelization. This is an important challenge because depending on the distribution of the system the privacy standards which can be fulfilled are different. Some possible distributions are already discussed in Section 2.2. Nevertheless, to decide for a distribution quantitative experiments should be done.

Concerning the main generalization algorithm, the possible extensions looked less promising than the discussed algorithms. Nevertheless, it is possible to extend the test scenarios, change the distance and the merging approach and to vary the parameters for finalization (Section 4.4) or to come up with another set of rules. For example, the threshold in steps for the connections can be replaced by a threshold in meters which might also produce good or even better results.

Chapter 8

Bibliography

[1] H. Alt and M. Godau. Computing the Fréchet distance between two polygonal curves. *International Journal of Computational Geometry & Applications*, 5:75–91, 1995.

[2] D. Anthony, T. Henderson, and D. Kotz. Privacy in location-aware computing environments. *IEEE Pervasive Computing*, 6(4):64–72, 2007.

[3] D. Ashbrook and T. Starner. Using GPS to learn significant locations and predict movement across multiple users. *Personal Ubiquitous Computing*, 7(5):275–286, Oct. 2003.

[4] R. J. Bayardo and R. Agrawal. Data privacy through optimal k-Anonymization. In *Proceedings of the 21st International Conference on Data Engineering*, ICDE '05, pages 217–228, Washington, DC, USA, 2005. IEEE Computer Society.

[5] J. Biagioni and J. Eriksson. Inferring road maps from global positioning system traces. *Transportation Research Record: Journal of the Transportation Research Board*, 2291(-1):61–71, Dec. 2012.

[6] BitMover, Inc. BitKeeper - the scalable distributed software configuration management system. `http://www.bitkeeper.com/`. Accessed: 2016-03-16.

[7] A. Bolbol, T. Cheng, I. Tsapakis, and J. Haworth. Inferring hybrid transportation modes from sparse GPS data using a moving window SVM classification. *Computers, Environment and Urban Systems*, 36(6):526–537, 2012. Special Issue: Advances in Geocomputation.

[8] F. Bonchi, L. V. Lakshmanan, and H. W. Wang. Trajectory anonymity in publishing personal mobility data. *SIGKDD Explorations Newsletter*, 13(1):30–42, Aug. 2011.

[9] C. Boucher and Z. Altamimi. ITRS, PZ-90 and WGS 84: current realizations and the related transformation parameters. *Journal of Geodesy*, 75(11):613–619, 2001.

[10] F. Brecht, B. Fabian, S. Kunz, and S. Müller. Are you willing to wait longer for internet privacy? In *Proceedings of the 19th European Conference on Information Systems (ECIS 2011), Helsinki, Finland*, 2011.

[11] F. Brecht, B. Fabian, S. Kunz, and S. Müller. Communication anonymizers: Personality, internet privacy literacy and their influence on technology acceptance. In *Proceedings of the 20th European Conference on Information Systems (ECIS 2012), Barcelona, Spain*, page 214, 2012.

[12] J. Brickell and V. Shmatikov. The cost of privacy: Destruction of data-mining utility in anonymized data publishing. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 70–78, New York, NY, USA, 2008. ACM.

[13] A. B. Brush, J. Krumm, and J. Scott. Exploring end user preferences for location obfuscation, location-based services, and the value of location. In *Proceedings of the 12th ACM International Conference on Ubiquitous Computing*, Ubicomp '10, pages 95–104, New York, NY, USA, 2010. ACM.

[14] K. Buchin, M. Buchin, J. Gudmundsson, M. Löffler, and J. Luo. Detecting commuting patterns by clustering subtrajectories. In S.-H. Hong, H. Nagamochi, and T. Fukunaga, editors, *Algorithms and Computation*, volume 5369 of *Lecture Notes in Computer Science*, pages 644–655. Springer Berlin Heidelberg, 2008.

[15] K. Buchin, M. Buchin, R. Leusden, W. Meulemans, and W. Mulzer. *Algorithms – ESA 2013: 21st Annual European Symposium, Sophia Antipolis, France, September 2-4, 2013. Proceedings*, chapter Computing the Fréchet Distance with a Retractable Leash, pages 241–252. Springer Berlin Heidelberg, Berlin, Heidelberg, 2013.

[16] Canonical Ltd. Bazaar. `http://bazaar.canonical.com/en/`. Accessed: 2016-03-16.

[17] L. Cao and J. Krumm. From GPS traces to a routable road map. In *Proceedings of the 17th ACM SIGSPATIAL International Conference on*

Advances in Geographic Information Systems, GIS '09, pages 3–12, New York, NY, USA, 2009. ACM.

[18] R. Chen, B. C. M. Fung, and B. C. Desai. Differentially private trajectory data publication. Technical report, 2011.

[19] D. Cvrcek, M. Kumpost, V. Matyas, and G. Danezis. A study on the value of location privacy. In *Proceedings of the 5th ACM Workshop on Privacy in Electronic Society*, WPES '06, pages 109–118, New York, NY, USA, 2006. ACM.

[20] Deutsches Zentrum für Luft- und Raumfahrt. Bis auf den Meter genau: Navigation mit Galileo. `http://www.dlr.de/next/desktopdefault.aspx/tabid-6804/11164_read-25462/`. Accessed: 2016-03-16.

[21] M. M. Deza and E. Deza. Encyclopedia of distances. `http://www.uco.es/users/ma1fegan/Comunes/asignaturas/vision/Encyclopedia-of-distances-2009.pdf`, 2009. Accessed: 2016-03-16.

[22] R. Dingledine, N. Mathewson, and P. Syverson. Tor: The second-generation onion router. In *13th USENIX Security Symposium, San Diego, California, USA*. Usenix, 2004.

[23] C. du Mouza, W. Litwin, and P. Rigaux. SDR-tree: a Scalable Distributed Rtree. In *International Conference on Data Engineering (ICDE'07), France*, pages 296–305. IEEE , 2007.

[24] C. Dwork. Differential privacy. In *The 33rd International Colloquium on Automata, Languages and Programming*, pages 1–12. Springer, 2006.

[25] T. Eiter and H. Mannila. Computing discrete Fréchet distance. `http://www.kr.tuwien.ac.at/staff/eiter/et-archive/cdtr9464.pdf`, 1994. Christian Doppler Laboratory for Expert Systems, TU Vienna. Accessed 2016-03-16.

[26] Ericsson. Ericsson mobility report, November 2013. `http://www.ericsson.com/res/docs/2013/ericsson-mobility-report-november-2013.pdf`, 2013. Accessed: 2016-03-16.

[27] J. Fischer. GPS track aggregation with use of Fréchet distance. Bachelor thesis, Freie Universität Berlin, Arbeitsgruppe Datenbanken und Informationssysteme, 2012.

[28] C. Garman, M. Green, and I. Miers. Decentralized anonymous credentials. `https://eprint.iacr.org/2013/622.pdf`, 2013. Accessed: 2016-03-16.

[29] Git Community. Git. http://git-scm.com/. Accessed: 2016-03-16.

[30] GlobalWebIndex. Global mobile application usage. http://blog.globalwebindex.net/Top-global-smartphone-apps, 2013. Accessed: 2016-03-16.

[31] Google. Android. http://www.android.com. Accessed: 2016-03-16.

[32] Google. Bbbike android client. https://play.google.com/store/apps/details?id=org.selfip.leinad.android.bbbike&hl=de. Accessed: 2016-03-16.

[33] Google. Maps. http://maps.google.com. Accessed: 2016-03-16.

[34] Google. Runtastic Laufen & Fitness - Android-Apps auf Google Play. https://play.google.com/store/apps/details?id=com.runtastic.android&hl=de. Accessed: 2016-03-16.

[35] J. Gudmundsson, A. Thom, and J. Vahrenhold. Of motifs and goals: Mining trajectory data. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '12, pages 129–138, New York, NY, USA, 2012. ACM.

[36] K. Haensch. Directive 95/46/EC of the european parliament and of the council of 24 october 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data, EUR-Lex - 31995L0046 - EN. http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=CELEX:31995L0046:EN:HTML, 1995. Accessed: 2016-03-16.

[37] M. Haklay and P. Weber. OpenStreetMap: User-generated street maps. *Pervasive Computing, IEEE*, 7(4):12–18, 2008.

[38] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: an update. *SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

[39] A. R. Hevner, S. T. March, J. Park, and S. Ram. Design science in information systems research. *MIS Quarterly*, 28(1):75–105, 2004.

[40] A. F. C. IV and C. Wenk. Geodesic Fréchet distance with polygonal obstacles. http://www.cs.utsa.edu/dmz/techrep/2008/CS-TR-2008-010.pdf, 2008. University of Texas at San Antonio. Accessed: 2016-03-16.

[41] J. H. Kang, W. Welbourne, B. Stewart, and G. Borriello. Extracting places from traces of locations. In *Proceedings of the 2Nd ACM International Workshop on Wireless Mobile Applications and Services on WLAN Hotspots*, WMASH '04, pages 110–118, New York, NY, USA, 2004. ACM.

[42] S. Karagiorgou and D. Pfoser. On vehicle tracking data-based road network generation. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '12, pages 89–98, New York, NY, USA, 2012. ACM.

[43] J.-G. Lee, J. Han, and K.-Y. Whang. Trajectory clustering: A partition-and-group framework. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, SIGMOD '07, pages 593–604, New York, NY, USA, 2007. ACM.

[44] F. G. Lemoine, S. C. Kenyon, J. K. Factor, R. G. Trimmer, N. K. Pavlis, D. S. Chinn, C. M. Cox, S. M. Klosko, S. B. Luthcke, M. H. Torrence, Y. M. Wang, R. G. Williamson, E. C. Pavlis, R. H. Rapp, and T. R. Olson. The development of the joint NASA GSFC and NIMA geopotential model EGM96. Technical report, 1998. NASA/TP-1998-206861.

[45] N. Li, T. Li, and S. Venkatasubramanian. t-Closeness: Privacy beyond k-Anonymity and l-Diversity. In *23rd International Conference on Data Engineering (ICDE)*, pages 106–115. IEEE Computer Society, 2007.

[46] Z. Li, J. Han, M. Ji, L.-A. Tang, Y. Yu, B. Ding, J.-G. Lee, and R. Kays. MoveMine: Mining moving object data for discovery of animal movement patterns. *ACM Transactions on Intelligent Systems and Technology*, 2(4):37:1–37:32, 2011.

[47] Z. Li, M. Ji, J.-G. Lee, L.-A. Tang, Y. Yu, J. Han, and R. Kays. MoveMine: Mining moving object databases. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, pages 1203–1206, New York, NY, USA, 2010. ACM.

[48] M. Lv, L. Chen, and G. Chen. Discovering personally semantic places from GPS trajectories. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, CIKM '12, pages 1552–1556, New York, NY, USA, 2012. ACM.

[49] A. Lysyanskaya. *Signature Schemes and Applications to Cryptographic Protocol Design*. PhD thesis, Massachusetts Institute of Technology, 2002.

[50] A. Machanavajjhala, D. Kifer, J. Gehrke, and M. Venkitasubramaniam. L-diversity: Privacy beyond K-anonymity. *Transactions on Knowledge Discovery from Data*, 1(1), 2007.

[51] Mercurial Community. Mercurial SCM. `http://www.mercurial-scm.org/`. Accessed: 2016-03-16.

[52] W. Meulemans. Source code. `https://www.win.tue.nl/~wmeulema/code.html`, 2014. Accessed: 2016-03-16.

[53] J. Mitlmeier. Generierung von Straßengraphen aus aggregierten GPS-Spuren. Master thesis, Freie Universität Berlin, Arbeitsgruppe Datenbanken und Informationssysteme, 2012.

[54] A. Monreale, G. Andrienko, N. Andrienko, F. Giannotti, D. Pedreschi, S. Rinzivillo, and S. Wrobel. Movement data anonymity through generalization. *Trans. Data Privacy*, 3(2):91–121, 2010.

[55] S. Müller, P. Mehta, and A. Voisard. *Web and Wireless Geographical Information Systems: 13th International Symposium, W2GIS 2014, Seoul, South Korea, May 29-30, 2014. Proceedings*, chapter Trajectory Aggregation for a Routable Map, pages 36–53. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[56] S. Müller, M. Janke, B. Fabian, and S. Kunz. Anonymisierung semantischer Daten. In *Statistische Woche*, 2010.

[57] National Imagery and Mapping Agency. World Geodetic System 1984. Technical Report TR 8350.2, 2000. `http://earth-info.nga.mil/GandG/publications/tr8350.2/wgs84fin.pdf`. Accessed: 2016-03-16.

[58] M. E. Nergiz, M. Atzori, and Y. Saygin. Towards trajectory anonymization: a generalization-based approach. In *Proceedings of the SIGSPATIAL ACM GIS 2008 International Workshop on Security and Privacy in GIS and LBS*, SPRINGL '08, pages 52–61, New York, NY, USA, 2008. ACM.

[59] B. Niehofer, R. Burda, C. Wietfeld, F. Bauer, and O. Lueert. GPS community map generation for enhanced routing methods based on trace-collection by mobile phones. In *Proceedings of the 2009 First International Conference on Advances in Satellite and Space Communications*, SPACOMM '09, pages 156–161, Washington, DC, USA, 2009. IEEE Computer Society.

[60] Nokia. Here. `http://here.com`. Accessed: 2016-03-16.

[61] D. O'Shea. Real-world drive tests declare a verdict on GPS/-GLONASS. `http://electronicdesign.com/test-amp-measurement/real-world-drive-tests-declare-verdict-gpsglonass`. Accessed: 2016-03-16.

[62] S. Reddy, M. Mun, J. Burke, D. Estrin, M. Hansen, and M. Srivastava. Using mobile phones to determine transportation modes. *ACM Transactions on Sensor Networks*, 6(2):13:1–13:27, 2010.

[63] S. Rezic. Bbbike. `http://www.bbbike.de`. Accessed: 2016-03-16.

[64] Runtastic GmbH. Runtastic. `https://www.runtastic.com/`. Accessed: 2016-03-16.

[65] M. I. Shamos. *Computational geometry*. PhD thesis, Yale University, 1978.

[66] R. W. Sinnott. Virtues of the Haversine. *Sky and Telescope*, 68(2):159+, 1984.

[67] I. Sivignon. DGtal: Fréchet shortcuts. `http://libdgtal.org/doc/nightly/moduleFrechetShortcut.html`. Accessed: 2016-03-16.

[68] L. Stenneth, O. Wolfson, P. S. Yu, and B. Xu. Transportation mode detection using mobile phones and GIS information. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, GIS '11, pages 54–63, New York, NY, USA, 2011. ACM.

[69] H. Su, K. Zheng, H. Wang, J. Huang, and X. Zhou. Calibrating trajectory data for similarity-based analysis. In *Proceedings of the 2013 ACM SIGMOD International Conference on Management of Data*, SIGMOD '13, pages 833–844, New York, NY, USA, 2013. ACM.

[70] L. Sweeney. Guaranteeing anonymity when sharing medical data, the Datafly system. `http://dataprivacylab.org/datafly/paper4.pdf`, 1997. Massachusetts Institute of Technology. Accessed: 2016-03-16.

[71] L. Sweeney. Achieving k-Anonymity privacy protection using generalization and suppression. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):571–588, 2002.

[72] L. Sweeney. k-Anonymity: A model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 10(5):557–570, 2002.

[73] The Android Open Source Project. Location. `https://developer.android.com/reference/android/location/Location.html`. Accessed: 2016-03-16.

[74] A. Thiagarajan, J. Biagioni, T. Gerlich, and J. Eriksson. Cooperative transit tracking using smart-phones. In *Proceedings of the 8th ACM Conference on Embedded Networked Sensor Systems*, SenSys '10, pages 85–98, New York, NY, USA, 2010. ACM.

[75] M. van Kreveld and J. Luo. The definition and computation of trajectory and subtrajectory similarity. In *Proceedings of the 15th Annual ACM International Symposium on Advances in Geographic Information Systems*, GIS '07, pages 44:1–44:4, New York, NY, USA, 2007. ACM.

[76] T. Vincenty. Direct and inverse solutions of geodesics on the ellipsoid with application of nested equations. Technical Report 176. `http://www.ngs.noaa.gov/PUBS_LIB/inverse.pdf`. Accessed: 2016-03-16.

[77] Vivid Solutions, Inc. JTS topology suite. `http://www.vividsolutions.com/jts/JTSHome.htm`. Accessed: 2016-03-16.

[78] M. I. Voitsekhovskii. Hausdorff metric. `https://www.encyclopediaofmath.org/index.php/Hausdorff_metric`, 2002. Accessed: 2016-03-16.

[79] Q. H. Vu, M. Lupu, and B. C. Ooi. *Peer-to-Peer Computing - Principles and Applications.* Springer-Verlag Berlin Heidelberg, 2010.

[80] C. Wenk, R. Salas, and D. Pfoser. Addressing the need for map-matching speed: Localizing global curve-matching algorithms. In *18th International Conference on Scientific and Statistical Database Management*, pages 379–388, 2006.

[81] C. Williams. Robolectric: Unit test your android application. `http://robolectric.org/`. Accessed: 2016-03-16.

[82] G. Xiaohui. FrechetDistance.java. `http://mytraj.googlecode.com/svn-history/r9/trunk/Frechet/src/geom/FrechetDistance.java`. Accessed: 2016-03-16.

[83] H. Zhu, J. Luo, H. Yin, X. Zhou, J. Z. Huang, and F. B. Zhan. Mining trajectory corridors using Fréchet distance and meshing grids. In *Proceedings of the 14th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining - Volume Part I*, PAKDD'10, pages 228–237, Berlin, Heidelberg, 2010. Springer-Verlag.

Appendix A

Graphical User Interface

A.1 Installation and Start

Agg2Graph does not provide any packages for installation at the moment. In order to be able to start the graphical user interface (GUI) one has to first download the complete project from GIT:

```
git clone git@git.imp.fu-berlin.de:semu/agg2graph.git
```

After cloning the project one can enter the directory:

```
cd agg2graph
```

There one can start the GUI with java (tested version 1.7.0_75) including all libraries in the classpath:

```
java -cp .:src:bin:libs/junit.jar:libs/org.hamcrest.core_1.3.0.
    v201303031735.jar:libs/jscience/jscience.jar:libs/sqlitejdbc-
    v056.jar:libs/opencsv-2.3.jar:libs/jsi-1.0b8.jar:libs/trove
    -2.0.2.jar:libs/dom4j-1.6.1.jar:libs/jaxen-1.1-beta-6.jar:libs/
    weka.jar:libs/swingx-ws-1.0.jar:libs/jmapviewer/JMapViewer.jar:
    libs/frechet.jar:libs/jts-1.8.jar:libs/swingx-all-1.6.4.jar:
    libs/hamcrest-all-1.3.jar:libs/processing.jar:libs/jfreechart
    -1.0.14.jar:libs/jcommon-1.0.17.jar:libs/jens.jar:libs/commons-
    jxpath-1.3.jar:libs/frechet-0.0.1-SNAPSHOT.jar:libs/apache-
    mime4j-0.6.jar:libs/bsh-1.3.0.jar:libs/cglib-nodep-2.1_3.jar:
    libs/commons-cli-1.2.jar:libs/commons-codec-1.6.jar:libs/
    commons-collections-3.2.1.jar:libs/commons-compress-1.5.jar:
    libs/commons-exec-1.1.jar:libs/commons-io-2.2.jar:libs/commons-
    lang3-3.1.jar:libs/commons-logging-1.1.1.jar:libs/cssparser
    -0.9.9.jar:libs/gpx.jar:libs/gpxparser-20130603.jar:libs/
    gpxparser-beta-0.3.jar:libs/guava-14.0.jar:libs/hamcrest-core
    -1.3.jar:libs/hamcrest-library-1.3.jar:libs/htmlunit-2.12.jar:
    libs/htmlunit-core-js-2.12.jar:libs/httpclient-4.2.1.jar:libs/
```

```
httpcore-4.2.1.jar:libs/httpmime-4.2.1.jar:libs/ini4j-0.5.2.jar
:libs/jcommander-1.29.jar:libs/jetty-websocket-8.1.8.jar:libs/
jna-3.4.0.jar:libs/jna-platform-3.4.0.jar:libs/json-20080701.
jar:libs/junit-dep-4.11.jar:libs/log4j-1.2.16.jar:libs/nekohtml
-1.9.17.jar:libs/netty-3.5.7.Final.jar:libs/operadriver-1.3.jar
:libs/phantomjsdriver-1.0.3.jar:libs/protobuf-java-2.4.1.jar:
libs/sac-1.3.jar:libs/selenium-java-2.33.0.jar:libs/xml-apis.
jar:libs/serializer-2.7.1.jar:libs/testng-6.8.jar:libs/android-
all-4.4_r1-robolectric-0.jar:libs/xercesImpl.jar:libs/xalan.jar
:libs/VectorGraphics2D-0.9.1_proguard_base.jar:libs/
VectorGraphics2D-0.9.1-javadoc.jar:libs/VectorGraphics2D-0.9.1-
sources.jar:libs/VectorGraphics2D-0.9.1.jar de.fub.agg2graph.
Main testui GPXMergeAggregation
```

In the start command above the $GPXMergeAggregation$ aggregation algorithm was selected. For an overview for the command line parameters one can execute the command without this parameter. The current available aggregation algorithms are:

```
- PathWCP
- PathFrechet
- PathAttraction
- PathIterative
- GPXMergeAggregation
- GPXFrechetAggregation
- EdgeAttraction
- EdgeIterative
- HausdorffWCP
- HausdorffFrechet
- HausdorffAttraction
- HausdorffIterative
- FrechetAttraction
- FrechetIterative
- ConformalAttraction
- ConformalIterative
- Original
- TraClus
- Others => PathWCP
```

The parameter `testui` tells the software to start in graphical user interface mode.

The GUI is shown in Figure A.1. After starting the tool one should set the aggregation algorithm if not already selected via command line parameters before. The graphical user interface is divided in five areas. The areas from top to bottom are:

1. the illustration of the part of the map which is currently on focus

2. the results of the intermediate steps shown in scrollable horizontal boxes

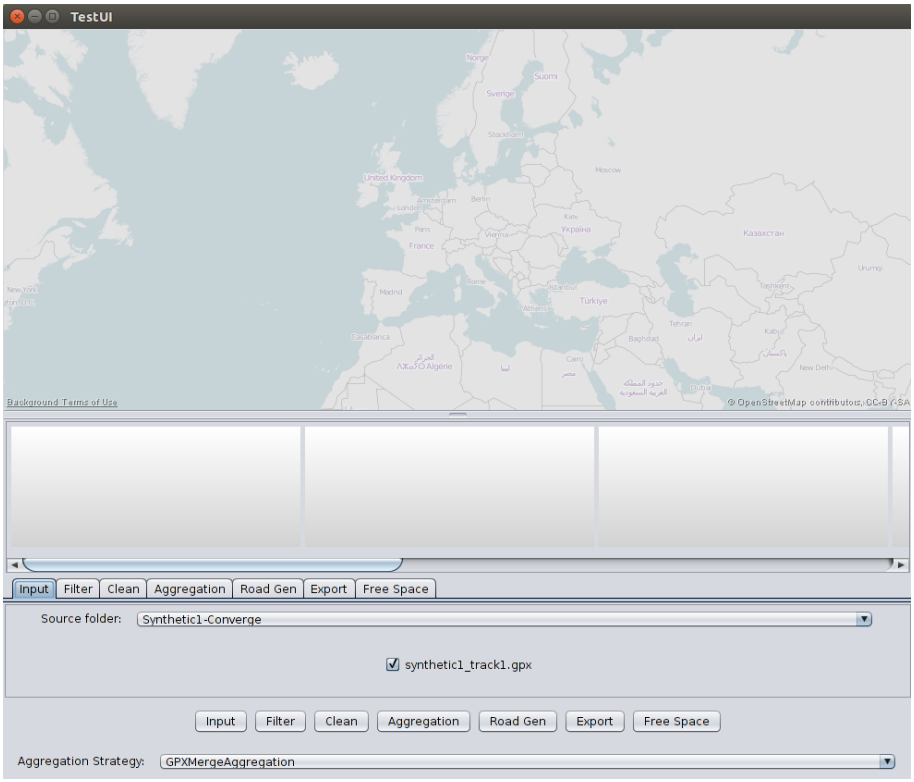


Figure A.1: GUI start screen

3. folders selected by tabs representing the parameters of the different steps of the aggregation process

4. execute buttons for the different steps

5. the selection of the aggregation algorithm

Please keep in mind that different aggregation algorithms offer different parameters to configure. Therefore, changing the selection of the aggregation algorithm will overwrite previously configured parameters.

A.2 Import

During the import step the files to be imported are selected and then imported. The selection of source folders is limited to subdirectories of a fixed directory. At the moment this is set to `test/input` starting from the root directory of the software. The import settings are already visible on the start screen show in Figure A.1 under the tab `Input`. Once the source folder is selected on can select multiple files within this folder. All trajectories of the input files are imported (one file might have one or more trajectories). To import the files the execute button `Input` needs to be pressed. Figure A.2 shows the screen after the files were imported. The map shows the two trajectories that were found in the selected input file. The two trajectories can also be seen in the first box of the intermediate step boxes.

A.3 Filter

After the cleaning finishes, the tab switches to the filter options. Currently, there is only the option to restrict aggregations with a weight (or k) smaller than the one which is set to be required. This is useful to test the anonymization for GPS trajectories. After adjusting the value of `kRequirement` one needs to press the button `Filter` to execute the filter.

A.4 Cleaning

Figure A.3 shows the screen after the filter options were applied. For cleaning there are more options which can be set. They are shown in Figure A.3. The options are:

1. `filterBySegmentLength`: A boolean value which enables the limitation of the segment length if set to `true`. The segment is equal to a trajectory

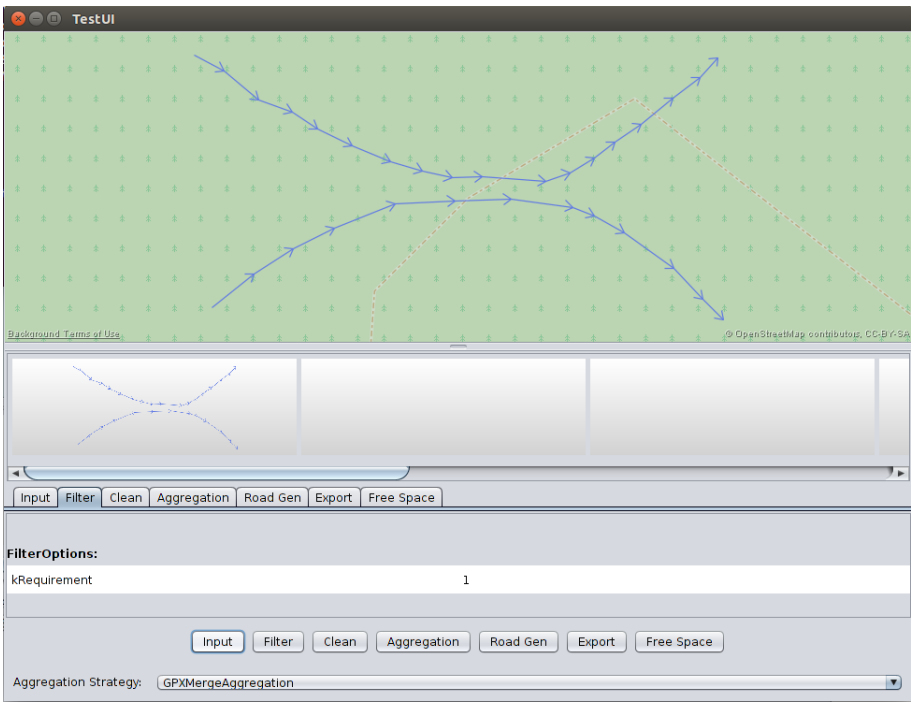


Figure A.2: GUI screen after input

found in an input file. This can be used to filter out very short trajectories which can be excluded already by their length as being useful for road network generation.

2. `minSegmentLength`: The minimum length of a trajectory to be considered if the option `filterBySegmentLength` is `true`.

3. `maxSegmentLength`: The maximum length of a trajectory to be considered if the option `filterBySegmentLength` is `true`.

4. `filterByEdgeLength`: A boolean value which enables the limitation of the edge or connection length if set to `true`. This is useful to normalize edges within a trajectory. When calculating the distance between edges, the length of the edges often influence the result and increase the difference of edges with different lengths. This effect is unwanted and can be reduced by normalization.

5. `minEdgeLength`: The minimum length of an edge if the option `filterByEdgeLength` is `true`. If the value falls below the threshold, the point (or node) to be inserted is omitted.

6. `maxEdgeLength`: The maximum length of an edge if the option `filterByEdgeLength`

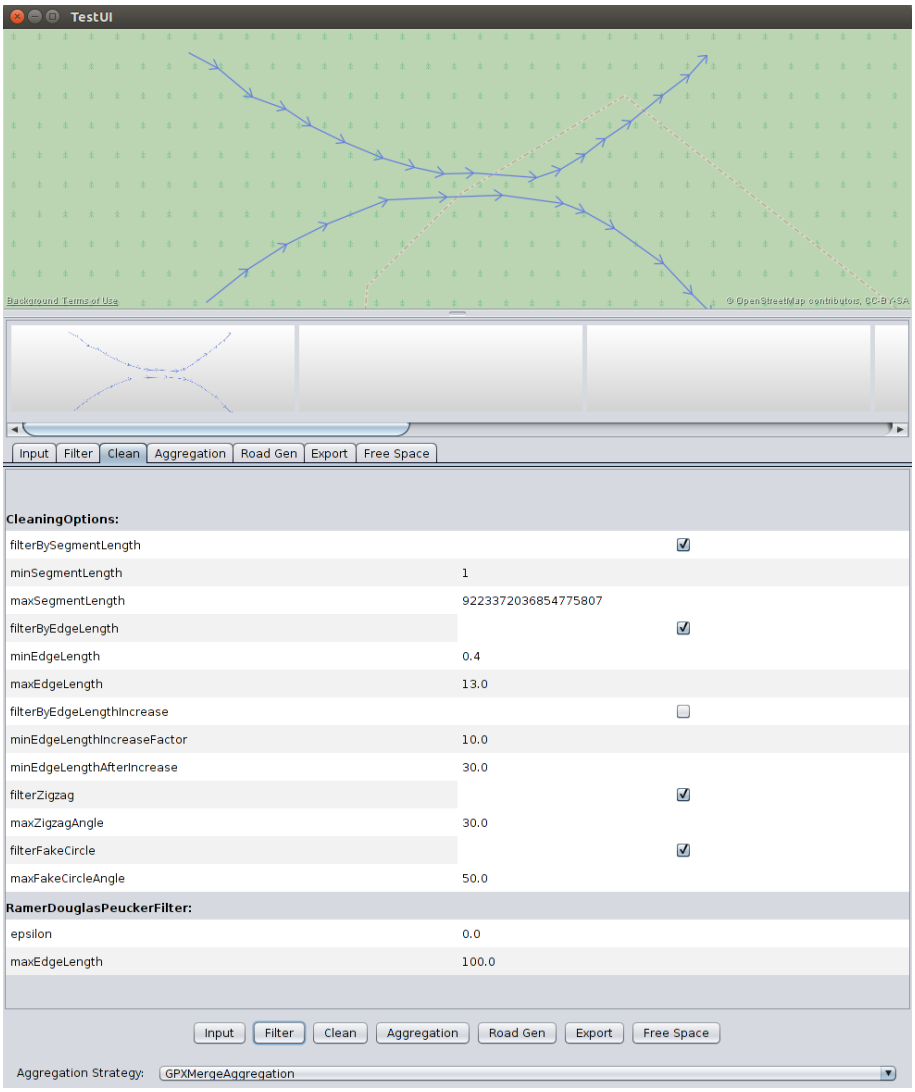


Figure A.3: GUI screen after filter

is `true`. If the value exceeds the threshold, the edge is divided in as many parts which are necessary to fulfill the threshold.

7. `filterByEdgeLengthIncrease`: A boolean value which enables the partition of segments (trajectories) if the thresholds are exceeded if set to `true`. This option is useful for long absence of the GPS signal when a very long edge is created, but the segments should be partitioned instead because the edge created during the absence is not useful for road network construction.

8. `minEdgeLengthIncreaseFactor`: The minimum factor which is necessary for the partition to be applied. The factor calculates the current edge length divided by the previous edge length.

9. `minEdgeLengthAfterIncrease`: The minimum edge length of the current edge which is necessary for the partition to be applied.

10. `filterZigzag`: A boolean value which enables a filter for strong heading change of sequential edges if set to `true`.

11. `maxZigzagAngle`: The maximum allowed heading change if `filterZigzag` is `true`.

12. `filterFakeCircle`: A boolean value which enables a filter for sequential heading changes near 180 degrees if set to `true`.

13. `maxFakeCircleAngle`: A fake circle is detected is for two sequential edges the heading change is larger than 180 degrees. Setting the value for `maxFakeCircleAngle` reduces this threshold by the given value.

Furthermore, there are the options for the Ramer-Douglas-Peucker Filter:

1. `epsilon`: This value expresses a tolerance for deviations to the given route. The lower this value the nearer the filter output is to the original trajectory. The higher this value the more simplified the result.

2. `maxEdgeLength`: This is a possibility to normalize edge length after the filter is applied. The Ramer-Douglas-Peucker Filter will sum together edges which were normalized before because this is a free simplification. In order to keep the normalization it has to be normalized again.

A.5 Aggregation

Figure A.4 shows the GUI screen after the cleaning was applied. One can see another result in the vertical boxes for the intermediate results. For the

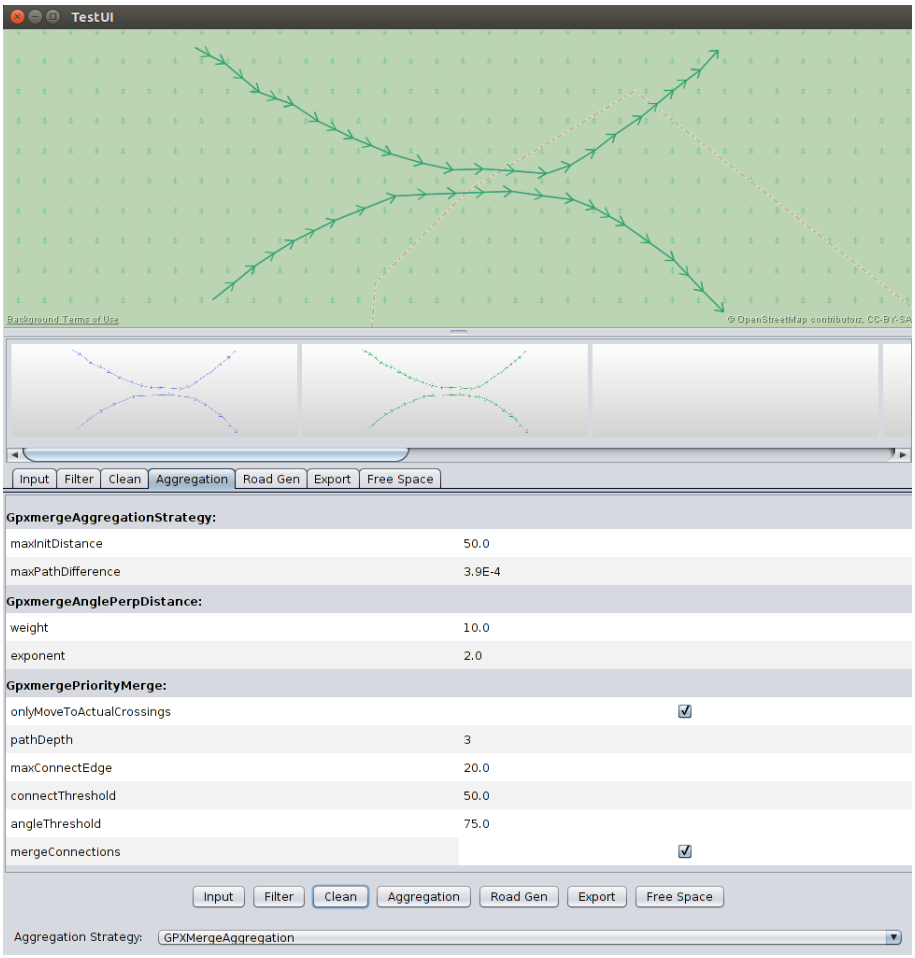


Figure A.4: GUI screen after cleaning

aggregation step settings for the matching, the distance and the merging can be given. For every aggregation strategy thepossible settings are different. Here, the settings for the `GPXMergeAggregation` are explained:

1. `GpxmergeAggregationStrategy`: This is the class which coordinates the matching process.
   (a) `maxInitDistance`: Before calculating the specialized distance function, this initial distance helps to filter candidates before applying a more complex distance function. This value is the threshold for a simple Euclidean distance for candidates to be considered.
   (b) `maxPathDifference`: In the case of the `GpxmergeAggregationStrategy` this is already the threshold for the applied distance function.
2. `GpxmergeAnglePerpDistance`: This is the class which represents the distance function.
   (a) `weight`: This is the weight which is given to the angle function of the distance in order to increase the importance of the difference in angle.
   (b) `exponent`: This is the exponent which is given to the value of the angle difference to exponentially include the difference in angle for a stronger effect of big differences.
3. `GpxmergePriorityMerge`: This is the class which coordinates the merging process.
   (a) `onlyMoveToActualCrossings`: When set to `true` matched nodes of the aggregation are only moved towards the individual trajectory if the perpendicular crosses an edge of the individual trajectory.
   (b) `pathDepth`: This number controls the connection which are made in the aggregation if the individual trajectory is along a track which is not connected in the aggregation. If such a gap is found then existing connections are searched within a path depth of this value.
   (c) `maxConnectEdge`: This is the maximum length a connecting edge can have. If this threshold is exceeded then the edge is partitioned in as many parts such that no edge is longer than this threshold.
   (d) `connectThreshold`: If this threshold is exceeded then no connection is created although indicated.
   (e) `angleThreshold`: If this threshold is exceeded then no connection is created although indicated.

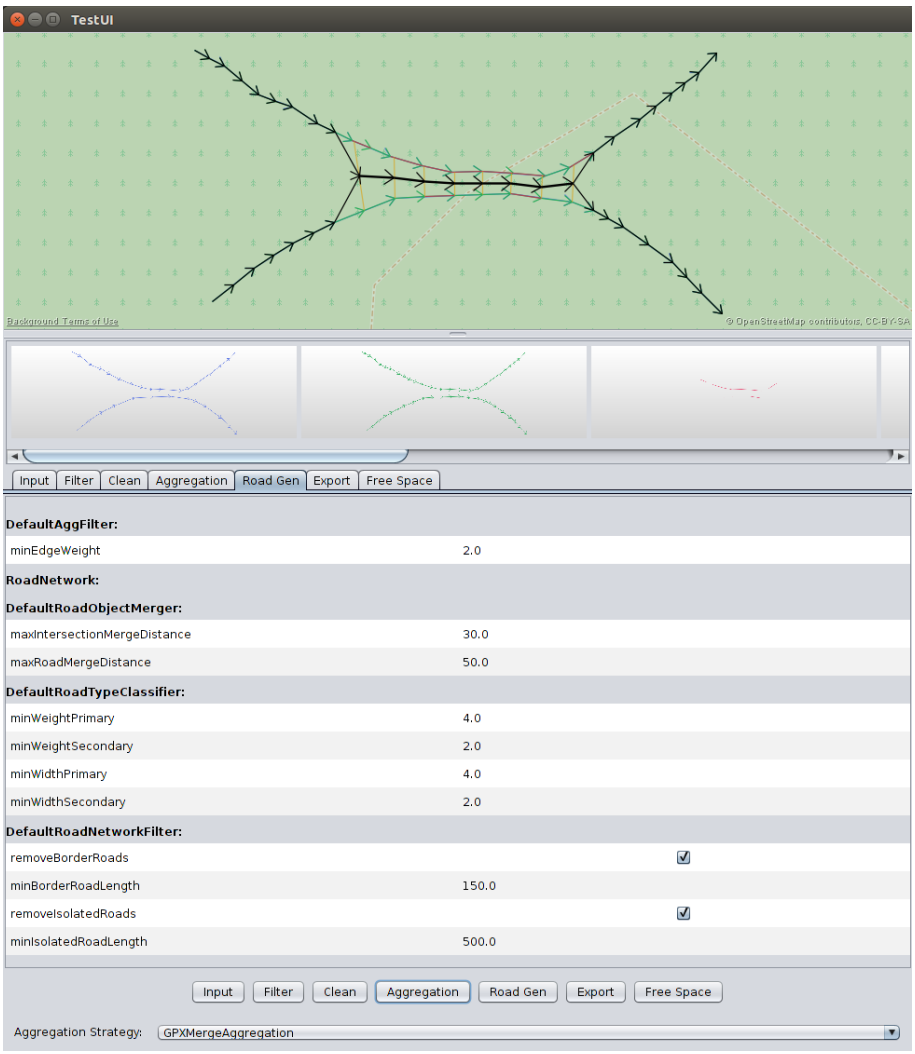


Figure A.5: GUI screen after aggregation

(f) `mergeConnections`: If this value is set to `true` then after creating a new connection it is checked whether it is possible to merge with an existent, using the pathDepth as search space.

Pressing the button `Aggregation` start the aggregation. The screen after the aggregation is shown in Figure A.5. The trajectory aggregation is shown in black and the perpendiculars used for the movement in yellow and the edges which took part in the merging process are marked red.

A.6 Road Generation

The settings for the road generation are shown in Figure A.5. The possible settings are:

1. `DefaultAggFilter`: A filter for the trajectory aggregation as prepocessing step before the road network generation.
   (a) `minEdgeWeight`: The minimum weight that an edge of the trajectory aggregation needs to have to be considered in the road generation step.
2. `Road Network`: Setting for the road network generation itself
   (a) `DefaultRoadObjectMerger`: Settings for the merging of connection in the trajectory aggregation. The end of a trajectory also accounts as a connection.
      i. `maxIntersectionMergeDistance`: This threshold in which connections of the trajectory aggregation are merged.
      ii. `maxRoadMergeDistance`: This threshold in which edges inbetween connections of the trajectory aggregation are merged.
   (b) `DefaultRoadTypeClassifier`: Settings for the tags which can be given a street in OpenStreetMap format to classify a street.
      i. `minWeightPrimary`: A minimum weight a trajectory in the trajectory aggregation has to have to be translated as primary street.
      ii. `minWeightSecondary`: A minimum weight a trajectory in the trajectory aggregation has to have to be translated as secondary street.
      iii. `minWidthPrimary`: A minimum width (calculated by the variance of the aggregated trajectories) a trajectory in the trajectory aggregation has to have to be translated as primary street.
      iv. `minWidthSecondary`: A minimum width a trajectory in the trajectory aggregation has to have to be translated as secondary street.
   (c) `DefaultRoadNetworkFilter`: The filter which is applied after the road network generation to improve the road network.
      i. `removeBorderRoads`: A boolean value which if set to `true` enables a filter which deletes roads created at the border of the bounding box of the trajectory aggregation.

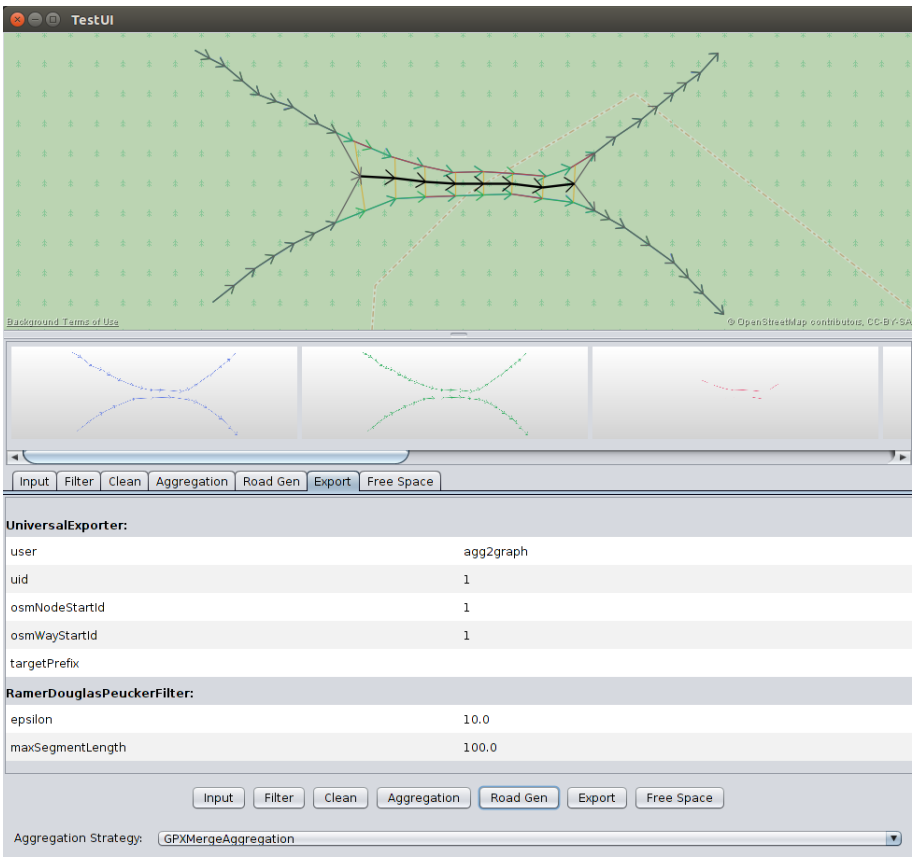


Figure A.6: GUI screen after road generation

ii. `minBorderRoadLength`: A minimum threshold for a road at the border to be identified as border road and deleted.

iii. `removeIsolatedRoads`: A boolean value which if set to `true` enables a filter which deletes roads without a connected crossing.

iv. `minIsolatedRoadLength`: A minimum threshold for a road to be identified as isolated road and deleted.

Pressing the button `Road Gen` start the road network generation.

A.7 Export

The screen after the road network generation is shown in Figure A.6. The setting for the export can also be seen in Figure A.6.

Appendix B

Zusammenfassung

Die vorliegende Arbeit beschreibt ein Verfahren zur Anonymisierung von Trajektorien, welches auf eine nachgelagerte automatisierte Kartengenerierung ausgerichtet ist. Eine automatisierte Kartengenerierung ist in der Literatur schon oftmals behandelt worden, vor allem seit die Verbreitung von Geräten mit GPS und Internetzugang zunimmt. Weiterhin gibt es bereits Verfahren zur Anonymisierung von GPS Spuren. Diese konzentrieren sich aber nicht auf eine mögliche spätere Verwendung zur Erstellung von Kartenmaterial. Der nachgelagerte Nutzen dieser anonymisierten Daten besteht vor allem in der späteren Datenanalyse, z.B. der Beantwortung von folgenden Fragen: Was sind hauptsächliche Verkehrsströme? Zwischen welchen Stadtgebieten gibt es die meisten Verbindungen? Dabei werden Detaildaten vernichtet, da sie für die Beantwortung nicht relevant sind. Diese Daten sind aber wichtig, um Aussagen über eine Karte zu treffen: Durch den genauen Verlauf von GPS Spuren kann auf den Straßenverlauf geschlossen werden, durch das Erkennen von Bewegungen können Abbiegevorschriften erstellt werden und durch die Varianz von GPS Spuren kann die Straßenbreite oder die Anzahl der Spuren ermittelt werden.

Mit einem speziell dafür ausgerichtetem Anonymisierungsverfahren, welches in einen Datensammelprozess eingebunden werden kann, werden Detailinformationen erhalten und übergeordnete Informationen verworfen. Wichtige Bestandteile sind die Erkennung von zusammengehörigen Trajektorien und die Zusammenfassung und Repräsentation von Trajektorien.

Die Evaluation vergleicht das Zusammenführen von GPS Spuren zu Aggregationen. Dabei werden insbesondere mehrere Distanzmaße und mehrere Strategien zur Zusammenführung miteinander verglichen. Eine wesentliche Aussage ist, dass kleinere Verbesserungen, welche darauf ausgerichtet sind, konkrete Probleme zu lösen, einen größeren Beitrag zu der Ergebnisgüte leisten können als die Anwendung unterschiedlicher Distanzmaße.

Appendix C

Abstract

The present paper describes a method to anonymize trajectories which focuses on a subsequent automated map construction. A process of automated map construction is intensively discussed in related work, particularly since the availability of devices with GPS and Internet access is increasing rapidly. Furthermore, there are already means to anonymize trajectory data. However, these concentrate not on a subsequent map construction. The subsequent use of these anonymized data is mostly data analysis which can, for example, answer the following questions: What are main traffic flows? Which districts have the highest number of connections in-between? Within this process detailed data is destroyed because it is not relevant to answer those questions. However, detailed data is necessary to provide evidence about a map: With the help of the exact course of the GPS traces, the exact course of the road can be deducted, with the analysis of movements turn rules can be created, and with the variation of the GPS traces the width of the road or the number of lanes can be determined.

With the help of a map construction targeted anonymization method, which can be integrated in a data collection process, detailed data is preserved and generic data is dismissed. Important elements are the identification of proximate trajectories and the aggregation and representation of trajectories.

The evaluation compares the merging of GPS trajectories to trajectory aggregations. Therefore, particularly multiple distance measures and multiple merging strategies are compared to each other. An important conclusion is that little improvements, which are targeted to solve practical problems, have a stronger impact on the quality of the result than the application of different distance measures.

Appendix D

Selbstständigkeitserklärung

Hiermit erkläre ich, die vorliegende Arbeit selbstständig und unter Angabe aller verwendeten Hilfsmittel angefertigt zu haben. Die Arbeit wurde vorher nicht schon einmal in einem früheren Promotionsvorhaben verwendet.
