THE UNIVERSITY OF QUEENSLAND
AUSTRALIA

A Study on Map-Matching and Map Inference Problems

Pingfu Chao
Master of Engineering

A thesis submitted for the degree of Doctor of Philosophy at
The University of Queensland in 2020
School of Information Technology and Electrical Engineering

Abstract

The recent upsurge of GPS(Global Positioning System)-equipped devices has enabled the tracking of users'/vehicles' locations. Meanwhile, the emergence of various location-based services emphasises the crucial role of the underlying digital map. However, the intrinsic inaccuracy of both maps and GPS trajectories has been a major issue that hinders the development of location-based applications, like navigation, location-based recommendation and traffic analysis. Several techniques are proposed to deal with the data quality issues in maps and GPS trajectories, respectively: (1) To avoid the excessive cost of ground surveying for map construction, the map inference algorithm aims to construct a digital map from GPS trajectories automatically. (2) To ensure the recency of digital maps, the map update algorithm focusses on updating an existing map using recent GPS trajectories. (3) To get rid of the GPS trajectory errors, the map-matching algorithm aligns a trajectory to the underlying map to find the user's actual travel path. In our thesis, we mainly focus on surveying the above techniques as well as proposing new solutions for solving the aforementioned data quality issues. Overall, our contributions are listed below.

- We conduct a comprehensive survey and experimental study of existing map inference algorithms. Due to the labour intensity of traditional map creation and the frequent road changes nowadays, map inference is deemed to be a promising solution to automatic map construction and updates. However, existing map inference algorithms suffer from low GPS accuracy, which makes the quality of the constructed map unsatisfactory. In this thesis, different from previous surveys, we (1) include the most recent solutions and propose a new categorisation of methods; (2) we study how different types of GPS errors affect the quality of inference results; (3) we compare the existing map inference quality measures on both real dataset and synthetic datasets, which are generated by our proposed data generators, regarding their ability to identify map quality issues. Overall, our study provides a guideline about (1) which inference method should be considered for each type of applications, (2) what trajectory quality should be guaranteed for map inference and (3) the direction of future work for quantitative map quality measures.

- We review the existing map-matching algorithms and study their performance under different data quality scenarios. As an indispensable pre-processing step for trajectories, the trajectory map-matching problem has been an ongoing research topic for more than two decades. In this thesis, we summarise and classify the existing map-matching algorithms based on their map-matching models, working scenarios and input data features. In addition, we conduct extensive experiments on several representative map-matching algorithms to compare their performance

under various working scenarios, data settings and performance requirements. The experiments are done on both real and synthetic datasets with different scales to (1) characterise the strengths and weaknesses of each algorithm category, (2) reveal how data quality issues affect the map-matching performance and (3) identify the remaining challenges in map-matching problems.

- We propose a co-optimisation framework that aims to solve the data quality of both GPS trajectories and maps simultaneously. Despite that many map-matching and map inference/update techniques have been proposed to deal with data quality issues on GPS trajectories and maps, respectively, calibrating one of the datasets relies on the other as a reference. Those reference data are required to be accurate, which is unobtainable in practice. Therefore, we propose a map-trajectory co-optimisation framework that takes the inaccurate map and trajectory data as inputs and mutually increases the accuracy of both. In our framework, both map-matching and map updates are run iteratively, and we propose two scores for each new map update, namely influence score and confidence score, to ensure the map is updated correctly and in a consistent way. Besides, our framework accepts most of the existing map-matching and map inference algorithms as candidate solutions in matching and update phases and receives straightforward performance boost through our framework.

- We develop a map service platform that supports the aforementioned data preprocessing/cleaning procedures. Considering the close relationship between map-matching and map inference/update and their broad applications, there is a lack of open-source tools providing those map-based data cleaning processes. In this thesis, we introduce the main features and functionalities of our map service platform, which supports multiple data cleaning processes, including map-matching, map inference and co-optimisation. Our platform provides various solutions for each type of cleaning process, which can be used for both future research comparisons and industrial applications.

Declaration by Author

Publications included in this thesis

Pingfu Chao, Wen Hua, Rui Mao, Jiajie Xu, and Xiaofang Zhou. *A survey and quantitative study on map inference algorithms*. Accepted by IEEE Transactions on Knowledge and Data Engineering (TKDE), 2020. –incorporated as Chapter 3.

Contributor	Statement of contribution
Pingfu Chao (Candidate)	Experiment design (60%) Experiment implementation (100%) Paper writing (70%) Idea discussion (30%)
Wen Hua	Experiment design (20%) Paper writing (30%) Proofreading (30%) Idea discussion (20%)
Rui Mao	Experiment design (20%) Idea discussion (20%)
Jiajie Xu	Proofreading (40%) Idea discussion (10%)
Xiaofang Zhou	Proofreading (30%) Idea discussion (20%)

Pingfu Chao, Yehong Xu, Wen Hua, and Xiaofang Zhou. *A survey on map-matching algorithms*. In Australasian Database Conference (ADC). Springer, 2020. –incorporated as Chapter 4

Pingfu Chao, Wen Hua, and Xiaofang Zhou. *Trajectories know where map is wrong: an iterative framework for map-trajectory co-optimisation*. In World Wide Web Journal (WWWJ), 2019. –incorporated as Chapter 5.

Pingfu Chao, Wen Hua, and Xiaofang Zhou. *An iterative map-trajectory co-optimisation framework based on map-matching and map update*. In International Conference on Database Systems for Advanced Applications (DASFAA), pages 305–309. Springer, 2019. –incorporated as Chapter 5.

Submitted manuscripts included in this thesis

Pingfu Chao, Yehong Xu, Wen Hua, and Xiaofang Zhou. *Classification and Comparison of Map-Matching Algorithms: An Experimental Study*. Expected to submit to WWWJ, 2020. –incorporated

Contributor	Statement of contribution
Pingfu Chao (Candidate)	Experiment design (60%) Experiment implementation (60%) Paper writing (100%) Proofreading (80%) Idea discussion (40%)
Yehong Xu	Experiment design (20%) Experiment implementation (40%) Idea discussion (30%)
Wen Hua	Experiment design (20%) Proofreading (20%)
Xiaofang Zhou	Idea discussion (30%)

Contributor	Statement of contribution
Pingfu Chao (Candidate)	Algorithm design (70%) Experiment implementation (100%) Paper writing (70%) Proofreading (40%)
Wen Hua	Algorithm design (30%) Paper writing (30%) Proofreading (30%)
Xiaofang Zhou	Proofreading (30%)

as Chapter 4

Other publications during candidature

- Junhua Fang, Pingfu Chao, Rong Zhang, and Xiaofang Zhou. *Integrating workload balancing and fault tolerance in distributed stream processing system.* World Wide Web Journal (WWWJ), pages 1–26, 2019.

- Pingfu Chao, Dan He, Shazia Sadiq, Kai Zheng, and Xiaofang Zhou. *A performance study on large-scale data analytics using disk-based and in-memory database systems.* In 2017 IEEE International Conference on Big Data and Smart Computing (BigComp), pages 247–254. IEEE, 2017.

Contributor	Statement of contribution
Pingfu Chao (Candidate)	Experiment design (60%) Experiment implementation (100%) Paper writing (70%) Proofreading (40%) Idea discussion (40%)
Wen Hua	Experiment design (40%) Paper writing (30%) Proofreading (40%) Idea discussion (30%)
Xiaofang Zhou	Proofreading (20%) Idea discussion (30%)

Contributor	Statement of contribution
Pingfu Chao (Candidate)	Experiment design (50%) Experiment implementation (60%) Paper writing (60%) Proofreading (30%) Idea discussion (40%)
Yehong Xu	Experiment design (30%) Experiment implementation (40%) Paper writing (30%) Proofreading (30%) Idea discussion (20%)
Wen Hua	Experiment design (20%) Proofreading (10%) Idea discussion (20%)
Xiaofang Zhou	Paper writing (10%) Proofreading (30%) Idea discussion (20%)

- Jiwon Kim, Kai Zheng, Sanghyung Ahn, Marty Papamanolis, and Pingfu Chao. *Graph-based analysis of city-wide traffic dynamics using time-evolving graphs of trajectory data*. In Australasian Transport Research Forum (ATRF), 38th, 2016.

- Mengxuan Zhang, Lei Li, Pingfu Chao, Wen Hua and Xiaofang Zhou. *Path Query Processing Using Typical Snapshots in Dynamic Road Networks*. Accepted by International Conference on Database Systems for Advanced Applications (DASFAA), 2020.

- Zhicheng Pan, Junhua Fang, Pingfu Chao, Wei Chen, Zhixu Li and An Liu. *Real-time Trajectory Similarity Processing Using Join-matrix*. Accepted by the 4th APWeb-WAIM International Joint Conference on Web and Big Data (APWeb-WAIM), 2020.

Other submissions during candidature

- Zonglei Zhang, Junhua Fang, Pingfu Chao, Wei Chen, Pengpeng Zhao and Zhixu Li. *A distributed spatial index with high update efficiency for real-time processing system*. Submitted to Information Sciences, 2020.

Contributions by others to the thesis

In all of the presented research in this thesis, Prof. Xiaofang Zhou, as my principal advisor, and Dr. Wen Hua and Prof. Kai Zheng, as my associate advisors, have provided technical guidance for formulating the problems, refining ideas as well as reviewing and polishing the presentation.

Statement of parts of the thesis submitted to qualify for the award of another degree

No works submitted towards another degree have been included in this thesis.

Research Involving Human or Animal Subjects

No animal or human subjects were involved in this research

Acknowledgements

It has been a long journey since the start of my PhD, and I believe I couldn't have gone this far without all the support from the people around me. Therefore, it is a great pleasure to convey my gratitude to them all in my humble acknowledgement.

First and foremost, I would like to give my earnest appreciation to my principal supervisor, Prof. Xiaofang Zhou, for offering me the opportunity to study here and for guiding me with great patience and kindness throughout my PhD period. Academically, his excellent research sense and professional knowledge helped me quickly overcome the early struggle of my study and build up my knowledge base. He made many suggestions and decisions were very critical and helpful to me and my goal of completing my PhD. Personally, his charming personality has always been the role model for me to learn from. Overall, it is truly a great honour and pure luck to be his student, and I wish I could repay him someday in the future.

I am deeply grateful to my associate advisors, Dr. Wen Hua and Prof. Kai Zheng, who have been supportive since the start of my PhD. I have not published or submitted any paper without their continual help in terms of problem discussion, paper writing, technical support and guidance. In addition, their excellence and success in research have set good examples for me and encourage me to continue my career as an academic researcher. Also, as good friends of mine, their kindness and companionship have helped me relieve the stressfulness and confusion during my study, which I can never thank them enough for.

Recalling these years of study in the Data Science group at UQ, so many names come to mind that I will never forget and would like to thank. Specifically, it is my pleasure to acknowledge Prof. Yufei Tao, Prof. Xue Li, A/Prof. Helen Huang (and her family), Prof. Shazia Sadiq, A/Prof. Gianluca Demartini, Dr. Lei Li (and his family), Dr. Junhao Gan (and his family), Dr. Hongzhi Yin (and his family), Dr. Sen Wang, Dr. Tony Chen, A/Prof. Sibo Wang, Dr. Jiwon Kim and Dr. Mohamed Sharaf. I also want to thank all my colleagues in our group, especially my office mates Mr Yu Liu, Mr Boyu Ruan, Mr Ruiyuan Zhang, Mr Runhui Wang, Ms Kexuan Xin, Mr Xuanyi Zhang, Ms Fengmei Jin, Ms Yehong Xu, Mr Douglas Alves Peixoto and Mr Mateusz Michalkiewicz with whom I have shared countless memorable and emotional moments. Their presence in my life has made my PhD study colourful and enjoyable.

Amongst all my colleagues, my special thanks go to Ms Dan He and Dr. Junhua Fang. Starting and finishing our PhDs at the same time, Dan and I have been fighting alongside each other the whole way and will continue to do so in the next few years. It is a blessing to have such a companion who always supports, encourages and pushes you and shares happiness and sorrow with you. As my former colleague and my best friend, Junhua has been helping me throughout my PhD and has been my best partner in terms of publications and cooperation. I wish my friendship with both of them (and of course our family members) will last forever.

Last but not least, my deepest gratitude goes to my family, including my darling Ms Wei Mao.

Although my parents have no idea what I am doing and what the purpose of this acknowledgement is, they have always trusted, supported and protected me throughout my lifetime. I can never thank them enough for bringing me into this beautiful world, teaching me how to be a man and giving me such a chance to study overseas and explore a new nation. I feel sorry for not accompanying them for almost seven years after graduation and I really appreciate them being healthy all the time so I can concentrate on developing my own career. I wish they will stay healthy, and I will work harder to relieve the burden on their shoulders in the future.

As for my dear love, you are the only one to whom I don't know how to express my sincere gratefulness in full. You are the one who cheers every progress I make and soothes every pain I gain; You are the one who blames me for not working hard but forces me to go to bed when I work too hard; You are the one who laughs at me, cries for me, leans on me, argues with me, takes care of me and fights alongside me; You are the one who witnesses what I witness, enjoys what I enjoy, hates what I hate and loves what I love; You are the one with whom I have shared the best five years of my life, and I know, you are the best I could ever have in my life.

Financial support

No financial support was provided to fund this research.

Keywords

map inference, map update, map-matching, map-trajectory co-optimisation, survey and experimental study

Australian and New Zealand Standard Research Classifications (ANZSRC)

ANZSRC code: 080604, Database Management, 100%

Fields of Research (FoR) Classification

FoR code: 0806, Information Systems 100%

Contents

1 Introduction **1**

1.1 Background . . . 2

1.1.1 GPS Trajectory and Road Map . . . 2

1.1.2 Data Quality Issues . . . 2

1.1.3 Data Preprocessing Techniques . . . 4

1.2 Problem Statement . . . 5

1.3 Motivations . . . 6

1.3.1 Map Inference Algorithms . . . 6

1.3.2 Map-Matching Algorithms . . . 7

1.3.3 Map-Trajectory Co-optimisation Framework . . . 8

1.3.4 Map Service Platform . . . 8

1.4 Main Contributions . . . 9

1.4.1 Map Inference Algorithms . . . 9

1.4.2 Map-Matching Algorithms . . . 9

1.4.3 A Co-optimisation Framework . . . 10

1.4.4 Map Service Platform . . . 11

1.5 Thesis Organisation . . . 11

2 Literature Review **13**

2.1 Data Quality Issues . . . 13

2.1.1 Spatial Trajectories and Trajectory Quality . . . 14

2.1.2 Maps and Map Quality . . . 15

2.2 Map-Matching . . . 18

2.3 Map Inference . . . 20

2.4 Map Update . . . 22

2.5 Summary . . . 23

3 Map Inference Algorithms **25**

3.1 Introduction 25

3.2 Algorithm Survey 27

3.2.1 Problem Statement 27

3.2.2 Survey and Categorisation 27

3.2.3 Road Abstraction 29

3.2.4 Incremental Branching 33

3.2.5 Intersection Linking 35

3.3 Quantitative Measure Evaluation 36

3.3.1 Map Quality Issues 37

3.3.2 Quantitative Measures 39

3.4 Synthetic Data Generator 41

3.4.1 Artificial Map Generator 41

3.4.2 Synthetic Trajectory Generator 44

3.5 Experiments 45

3.5.1 Experimental Settings 45

3.5.2 Evaluation of Measures 46

3.5.3 Evaluation of Inference Algorithms 48

3.5.4 Experiment Findings 52

3.6 Summary 52

4 Map-Matching Algorithms **55**

4.1 Introduction 55

4.2 Preliminaries 56

4.3 Map-Matching Models 57

4.3.1 Similarity Model 59

4.3.2 State-Transition Model 60

4.3.3 Candidate-Migration Model 62

4.3.4 Scoring Model 64

4.4 Tuning Techniques 65

4.4.1 Parameter Tuning 65

4.4.2 Data Preprocessing 65

4.5 Evaluation Metrics 66

4.5.1 Point-based Metrics 66

4.5.2 Route-based Metrics 67
4.5.3 Metrics without Ground-Truth 67
4.6 Experiments 68
4.6.1 Dataset Settings 69
4.6.2 Experimental Design and Results 71
4.6.3 Experiment Findings 75
4.7 Remaining Challenges 75
4.7.1 Trajectory Systematic Error 76
4.7.2 Parallel Road 76
4.8 Summary 77

5 A Co-optimisation Approach 79
5.1 Introduction 79
5.2 Framework Overview 81
5.2.1 Problem Definition 81
5.2.2 Co-Optimisation Framework 83
5.3 Co-Optimisation Model Design 86
5.3.1 Co-optimisation Quality Evaluation 86
5.3.2 Road Correctness Identification 88
5.3.3 Result Refinement 90
5.4 Running Time Optimisation 90
5.4.1 Direct Matching Area (DMA) 93
5.4.2 Shortest Transition Area (STA) 93
5.5 Evaluations 95
5.5.1 Experimental Settings 96
5.5.2 Evaluation Metrics 97
5.5.3 Map-Matching Evaluation 98
5.5.4 Co-optimisation Algorithm Evaluation 99
5.6 Summary 104

6 Map Service Platform 105
6.1 Introduction 105
6.2 Framework Overview 107
6.3 Implementation Details 109

6.3.1 Data Preprocessing 109
6.3.2 Map Services 111
6.3.3 Result Evaluation 112
6.3.4 Utilities 113
6.4 Case Study 114
6.4.1 Data Preprocessing 114
6.4.2 Map-matching and Evaluation 115
6.4.3 Visual Comparison 115
6.5 Conclusion 117

7 Final Remarks 119
7.1 Conclusions 119
7.2 Directions for Future Work 121
7.2.1 Trajectory Reconstruction on Low-Quality AVI Data 121
7.2.2 Pattern Learning for Map Intersection Inference 121

List of Figures

1.1 Example of map from Google Maps [51] 3
1.2 Example of GPS trajectory and its actual travel path 3

2.1 Example of travel route and sampled trajectory 16

3.1 Example of a wrong connection 31
3.2 Visually compare two constructed maps, no clear winner 37
3.3 Various map errors generated by synthetic map generator 43
3.4 Summary of experimental results of map inference algorithms 49

4.1 Summary of experimental results 72
4.2 Map-matching error caused by systematic errors 76
4.3 Different matching results on parallel road (HMM and SCO) 77

5.1 Framework overview 84
5.2 Example of two map-matching cases 93
5.3 Example of *STA-Circle* and *STA-Route* 95
5.4 Map-matching performance by varying θ_c 98
5.5 Effectiveness test for co-optimisation model 100
5.6 Efficiency and scalability test for co-optimisation model 102

6.1 Framework overview 107
6.2 Example of map-matching parameters 115
6.3 Quantitative evaluation of map-matching 116
6.4 Map-matching result visualisation on Google Maps 116
6.5 Map-matching result on Microsoft satellite image 117

List of Tables

2.1 Differences between online and offline map-matching scenario 19
2.2 Existing map-matching surveys 20

3.1 Summary of map inference methods 28
3.2 Comparison of public and private datasets 46
3.3 Measurement results under different topological error (TE) and road loss (RL) ratios 48
3.4 Measurement results under various other error types 48

4.1 Categorisation of map-matching algorithms in different scenarios 58
4.2 Summary of experiment datasets 70

5.1 Main symbols and associated meanings 84
5.2 Dataset specification 96

Chapter 1

Introduction

In recent years, the increasing popularity of positioning units, especially GPS-equipped (Global Positioning System) devices, has enabled the continuous tracking of users'/vehicles' locations. Operating as the sampling of objects' historical locations, trajectory data have become the backbone of various location-based applications, like navigation, fleet tracking, traffic management and location-based recommendation, which also heavily rely on the availability of high-quality digital maps. However, despite the ubiquity, the quality of both GPS trajectories and maps are unsatisfactory: (1) as a result of the instability of GPS signals [97], GPS trajectories usually represent objects' locations inaccurately; (2) there are still plenty of regions with blank or low-resolution maps around the world due to their low profitability and commercial interest; (3) since the maps are rarely updated given the extensive update cost, they are usually outdated and erroneous. In fact, without data correction, these quality issues can significantly affect the performance of downstream applications. For instance, navigating on an erroneous map may cause accidents, and insightful knowledge from the recorded trajectory is unusable if the corresponding travel path is not identified.

Different from common data quality issues, like data duplication and missing/incorrect values, the data quality issues occurring in GPS trajectory and road map data cannot be solved by generic data cleaning processes. Therefore, to eliminate the influence of GPS errors, the **map-matching** technique is proposed for trajectory data calibration. By matching the trajectory to the underlying map, it can identify the actual travel path of a trajectory. On the other hand, the **map inference** and **map update** techniques utilise trajectory data as the source to construct or update the road map automatically, which ensures map recency. Despite various solutions proposed in each category, there is a lack of research on how those techniques interact with each other and how the quality issues that occur in one type of data (e.g. road map) affect the correction process of another data type (e.g. GPS trajectory). Therefore, in our thesis, we first study the main features of GPS trajectory errors and map errors; then, we analyse the aforementioned techniques and propose our solution, which corrects both data types in one operation, termed as map-trajectory co-optimisation.

1.1 Background

1.1.1 GPS Trajectory and Road Map

A GPS trajectory is usually represented by a sequence of geographical locations sampled from a GPS-equipped device on the earth. To generate location samples, a GPS module communicates with multiple GPS satellites periodically and calculates its location each time by measuring its relative distances to them. Nowadays, as the GPS module is widely installed in mobile phones, vehicles and other portable devices, a massive amount of trajectory data is generated every day representing users' travel history. The samples from the same trajectory are generated by the same GPS device and are stored chronologically, with each sample recording the coordinates ($< longitude, latitude >$ or $< x, y >$ according to the coordinate system) and the corresponding timestamp t.

In terms of a road map, a map (also known as a road network) refers to a digital map that stores the geographical and topological information of a map region on the earth, an example shown in Figure 1.1. Different from traditional paper maps, a road map is stored as a graph whose vertices (red dots) represent the road intersections, and edges (green lines) depict the roads connecting them. Each road contains information, like road width, speed limit and number of lanes, while the intersections store attributes, like turning restrictions and traffic signals. In addition, some road maps also mark the locations with unique utilities, termed as Point-Of-Interest (POI, shown as pinned locations in Figure 1.1). As the map is the foundation of location-based services, it is widely used in various applications like navigation, route planning and traffic analysis.

1.1.2 Data Quality Issues

Despite the ubiquity of GPS trajectories and maps, the quality of both datasets is deficient. As the GPS device measures its distance to the satellites by tracking the GPS signals sent to them, which pass through the atmosphere, the measurement accuracy is always affected by multiple factors, like receiver/satellite clock synchronisation, ionosphere disturbance, path block/reflection and Dilution Of Precision (DOP) [40]. In addition to the measurement errors caused by unstable GPS signals, the sampling errors caused by low sampling rate also introduce uncertainty into the trajectories as the travel path between every two consecutive trajectory samples is completely unknown. As shown in Figure 1.2, the green dots are the sampled locations of the vehicle, which deviate from the actual

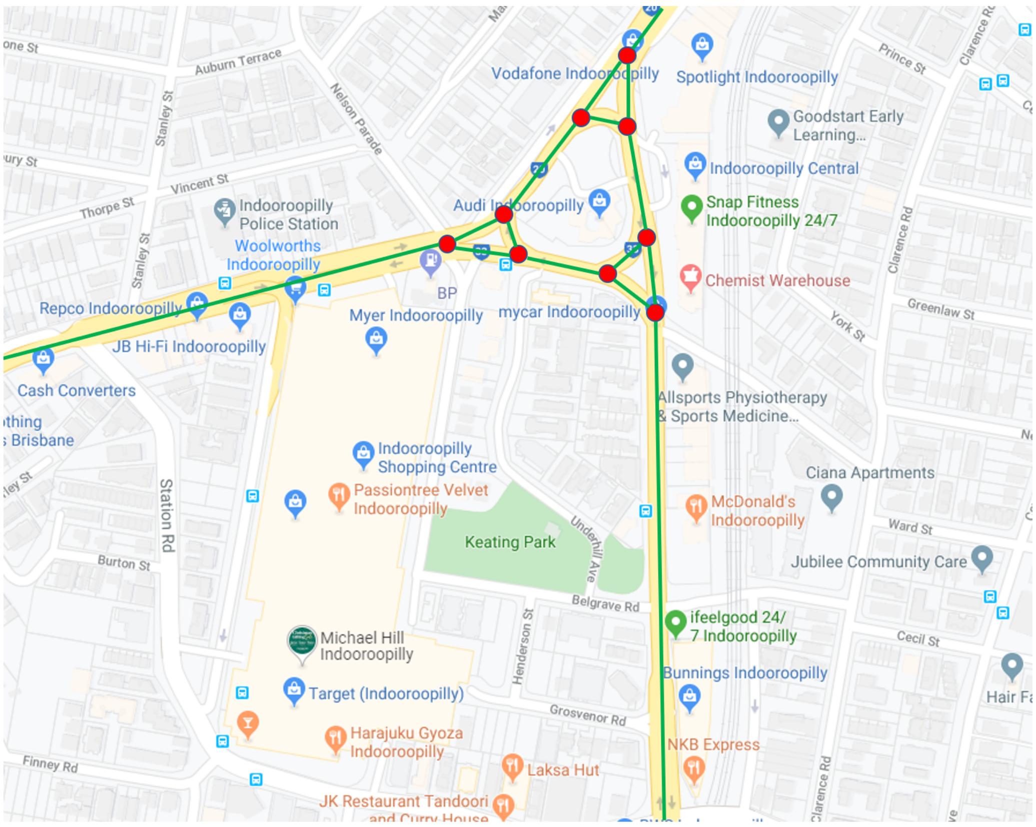
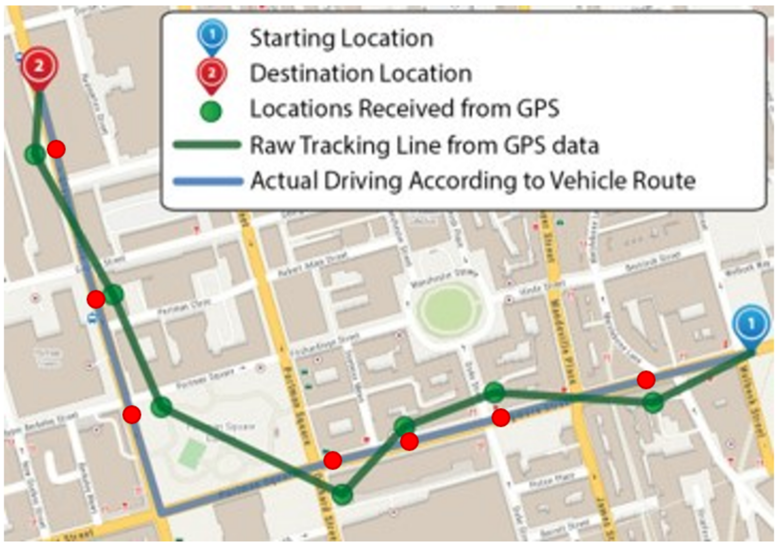


FIGURE 1.1: Example of map from Google Maps [51]

location (red dots) due to measurement error. Therefore, the combination of measurement errors and sampling errors makes the current GPS trajectories hard to represent the actual travel path of an object precisely.


Starting Location
Destination Location
Locations Received from GPS
Raw Tracking Line from GPS data
Actual Driving According to Vehicle Route
FIGURE 1.2: Example of GPS trajectory and its actual travel path

On the other hand, previously, road maps were constructed and updated through comprehensive ground surveys, which ensured high map accuracy and completeness. However, since ground surveys are usually labour-intensive and time-consuming, nowadays, the maps are updated less frequently such that the recent changes on road networks cannot be updated timely. In fact, several reports [116, 134] have pointed out that up to 15% of the road changes each year in some way around the world. The map recency issue has become a crucial factor affecting the reliability of location-based applications and causes problems, like incorrect navigation results and vehicle tracking failure. Therefore, there is a strong need for an automatic map update process that ensures frequent map updates with quality guaranteed.

1.1.3 Data Preprocessing Techniques

Due to data quality issues, both GPS trajectories and maps are required to go through a data preprocessing procedure for data correction. In addition to some generic data cleaning processes, like data deduplication and outlier detection, several techniques are proposed to deal with the data quality issues happening specifically in GPS trajectories and maps.

To address the trajectory quality issues, the map-matching problem is proposed to find the actual travel path for each trajectory. Following the idea that a vehicle usually travels on the road network, the trajectory samples should always be close to their actual locations on roads. Therefore, the map-matching process aims to align the trajectory samples to the nearby roads on the map and connect them so as to infer the actual path. With the help of the map-matching process, an actual travel path can be extracted from each trajectory, which is useful in various applications, such as traffic analysis, driving behaviour analysis, vehicle tracking and navigation.

To address the map recency issue, the map inference and map update techniques are proposed for map construction and updates automatically. Regarding the map inference, two major resources are being used as input for map inference: the aerial imagery [31] and GPS trajectory. Different from the aerial imagery inference which detects the map layout from static satellite images, the map inference on GPS trajectories utilises the idea that a road may occur in an area where multiple trajectories pass through it. As a GPS trajectory dataset has a larger volume than aerial images and contains more information, such as road connectivity, turning restriction and road speed, it helps improve the quality of the constructed map. Similar to map inference, the map update using GPS trajectories assumes that an initial map is given, the main goal of the map update is to infer the missing intersections/roads in the initial map as well as remove map components that no longer exist. The major difference between map inference and map update is that only the GPS trajectories that are believed to run on new roads are used in the map update process. Therefore, the main process of the map update is to first filter out the trajectories that run on the existing map, using map-matching methods, and then generate new roads by map inference algorithms. In general, most of the existing map update solutions [111, 133, 134, 143] are the combination of map-matching and map inference problems together with other data processing techniques, like map conflation and noise reduction [133, 134].

1.2 Problem Statement

As mentioned above, the main focus of this thesis is to study the data quality issues in GPS trajectories and road network, and the corresponding preprocessing techniques, namely map inference, map-matching and map update. Since all those techniques rely on both trajectory and road network data, we first give formal definitions of two data types. Typically, a trajectory is defined as follows:

Definition 1 (Trajectory)**.** *A **trajectory** Tr is a sequence of spatial points $Tr : p_1 \rightarrow p_2 \rightarrow ... \rightarrow p_n$ sampled from a continuously moving object. Each point p_i consists of a 2-dimensional coordinate $\langle x_i, y_i \rangle \in R^2$ and a timestamp t_i, i.e.: $p_i = \langle x_i, y_i, t_i \rangle$. Points in Tr are chronologically ordered, i.e.: $\forall i < j$ s.t. $t_i < t_j$, and every two consecutive points $\overline{p_i p_{i+1}}$ are connected and form a **trajectory segment**.*

In this thesis, we mainly focus on GPS trajectories, which are objects' trajectories sampled by GPS-equipped devices. Meanwhile, we give the definition of road network, which is also known as road map:

Definition 2. *(Road Network) A **road network** (also known as road map) is a directed graph $G = (V, E)$, in which a vertex $v = (x, y) \in V$ represents an intersection or a road end, and an edge $e = (s, e, l)$ is a directed road connecting vertices s and e with a polyline represented by a sequence of spatial points, termed as mini nodes, i.e.: $l : s \rightarrow n_1 \rightarrow n_2 \rightarrow ... \rightarrow n_k \rightarrow e$.*

The road network is a highly geographical graph, storing actual locations of intersections, shapes of road segments, the length of paths, etc. Normally, an edge is used to represent a complete road which starts from and ends at an intersection/road end, with no entry or exit in between. However, since most roads in real-world are not straight, a road is usually represented by a polyline l illustrating its actual shape.

The detailed definitions of the preprocessing techniques will be introduced in respective chapters. Here, we briefly show the ideas of them to help better distinguish those research problems. In terms of the map-matching, the objective of a map-matching algorithm is to find the object's actual on-road moving path by aligning its GPS trajectory to the underlying road network. Hence, the input of a map-matching process consists of both the trajectory and the road network, while the output is a sequence of road edges representing the object's actual travel path. On the contrary, both the map inference and map update aim to generate a road network from GPS trajectories. The core input of them is the GPS trajectories, and their output is a road network. In map inference, the GPS trajectories are merged into road edges and intersections to form the road network, whereas, in map update, it also takes an old map as input and use GPS trajectories to improve the quality of the obsolete map.

1.3 Motivations

In recent years, the pervasive use of GPS-enabled devices has produced massive number of users' vehicular trajectories. Since every vehicle must run on actual roads, on the one hand, these trajectories can be aligned to road networks via map-matching techniques [103] in order to support map-based applications, such as traffic congestion prediction and online navigation. On the other hand, they can potentially be used to automatically construct or update digital maps [5, 134]. In fact, despite their different objectives, the map-matching and map inference/update problems are correlated technically. Specifically, in most map update methods, a map-matching process is conducted in the first place to obtain trajectories that cannot be matched to the current map and hence used to generate new roads [111, 133, 134, 143]. Some map inference methods also leverage map-matching as a post-processing step to evaluate the quality of the constructed roads [18]. However, the intrinsic inaccuracy of both maps and GPS trajectories poses great challenges to all the above research problems: (1) maps that contain missing roads and inaccurate layout can cause GPS trajectories to be map-matched incorrectly; and (2) the measurement errors and sampling errors in GPS trajectories incur great uncertainty to the map inference and map update process, which leads to inaccurate map construction/updates.

Therefore, in our thesis, we mainly study the data quality issues in GPS trajectories and maps, together with the solutions, including map-matching, map inference and our proposed map-trajectory co-optimisation. Besides, to ensure our contributions are useful practically, we develop an open-source map service platform which supports map-matching, map inference and our proposed map-trajectory co-optimisation processes.

1.3.1 Map Inference Algorithms

The map inference problem was first studied in early 2000s [43] when high-quality Differential GPS (DGPS) trajectories were used as inputs to infer maps by a simple clustering method (k-means clustering). Since then, various types of solutions have been proposed aiming to construct high-quality maps using GPS trajectories with lower accuracy. Meanwhile, several survey papers [4, 5, 17, 42, 55] have been published summarising existing map inference algorithms from different perspectives. In particular, Ahmed et al. [4, 5] present a well-rounded introduction to previous algorithms based on their categorisation, together with evaluation measures, experiments and comparison. These works provide us with a great overview of the existing solutions and help new researchers quickly dive into this area with some available codes [5]. However, as they mainly focus on delivering the idea of map inference, most of their experiments are conducted through visualisation. Although other surveys compare the solutions via quantitative measures, they all lack sufficient candidates [5, 17, 55] and proper reasoning for their measures. Overall, none of the existing work clearly illustrates the advantages of each type of solutions, and there is no research on how GPS data quality affects their performance. As plenty of new solutions [44, 78, 100, 131, 132, 147, 161] have emerged since the last survey [5] and many of them are not able to be classified into any of the existing categorisations, in our thesis, we review the existing map inference methods and propose a new categorisation based on their methodologies. More importantly, we focus on the influence of data quality issue in GPS trajectories to the map inference solutions. Therefore, we conduct extensive experiments on map inference algorithms from different categories and identify their weakness to certain types of trajectory errors.

1.3.2 Map-Matching Algorithms

As the essential data preprocessing step, the map-matching problem has been studied for more than two decades [14]. However, despite the massive number of map-matching algorithms proposed, only a few surveys [28,56,74,103,114,136] classify or compare them. Quddus [103] et al. first summarised the early algorithms in 2007. It categorises the methods based on the matching principles (geometrical/topological) or the computation model (probability/advanced) adopted in the algorithms. Such categorisation was widely accepted but is now obsolete as it fails to classify most of the new methods proposed afterwards. Nevertheless, apart from some recent topic-specific surveys [56, 74], this categorisation is still widely adopted by recent papers [73, 101, 118, 122] and surveys [114], which indicates the need of a review in this field. Besides, recent works bring various new matching frameworks [112, 119] and tuning techniques [57, 63, 81, 95] to solve the map-matching problem on new types of positioning data (DGPS, inertial sensor, laser scanner [86], camera [71]) and new queries (lane-level map-matching [41,71,86]). Hence, it is worth conducting a comprehensive survey so as to classify the existing solutions and identify the strength/weakness of each type of methods. Therefore, in our thesis, we summarise the existing methods and propose a new categorisation according to their map-matching model, the working scenarios and data input. In addition, we experimentally compare the representative solutions from those categories and evaluate how trajectory data quality affects the map-matching result and the effectiveness of existing tuning methods.

1.3.3 Map-Trajectory Co-optimisation Framework

Although the map-matching and map inference have been extensively studied in the last two decades, there is still a lack of research on the map update, and the existing solutions are mainly simple combinations of map-matching and map inference processes. Overall, the current solutions in map-matching, map inference and map update are still defective: (1) To the best of our knowledge, all of the existing map-matching methods are based on an assumption that the underlying map is correct, which is not the real case. In practice, the chance of trajectory mismatch caused by the incomplete roadmap is significant; (2) Due to the trajectory noise and disparity, the accuracy of the state-of-the-art map inference methods is too low to be practical and there is a lack of evaluation steps for result pruning; (3) The existing map update method utilises the map-matching methods in an inefficient way. The current map update process spends most of the running time on map-matching only for finding unmatched trajectory points, while the original matching results are completely wasted. Therefore, in our thesis, we propose a map-trajectory co-optimisation framework which considers both the map and GPS trajectory quality issues and improve the quality of map-matching and map inference simultaneously through our iterative process.

1.3.4 Map Service Platform

In recent years, both map-matching and map inference/update have become hot research topics due to their crucial roles in spatial and spatial-temporal data management. Therefore, on the one hand, new research works in these fields are required to compare with existing solutions to show their performance superiority, on the other hand, as important data preprocessing procedures, these techniques are widely used in various applications in practice. Considering the popularity and the close relationship between these techniques, it is good to have a platform that supports these data preprocessing procedures for data quality improvement. However, to the best of our knowledge, despite the massive number of map-matching and map inference solutions proposed, no existing work tries to build a public platform that provides those map-based services. Therefore, in this thesis, we introduce our open-source map service platform for map and trajectory quality improvement, evaluation and comparison.

1.4 Main Contributions

The main contributions included in our thesis consist of three main aspects: (1) the insights of existing works, (2) the proposal of new solutions and (3) the development of a new practical platform.

1.4.1 Map Inference Algorithms

As mentioned in Section 1.3.1, in our thesis, we summarise the existing map inference methods and propose a new categorisation. To evaluate the major differences between various categories, we conduct a comprehensive experimental study on different map inference algorithms. The chosen candidates, including both up-to-date and classical solutions, are more diverse in terms of their methodology. Moreover, by introducing datasets with different scales and characteristics and deploying the solutions under the same platform, we are able to evaluate both their effectiveness and efficiency under different input features, which has never been done by any previous work. In the experiments, we study how the input GPS data quality affects the performance of various inference algorithms. To this end, we propose a synthetic dataset generator to simulate different types of GPS errors. Meanwhile, we discuss the difficulty of map inference evaluation and the weaknesses of current map quality evaluation system. To address the issues of existing quality measures, we elaborate on the possible map quality issues of constructed maps and propose an artificial map generator for evaluation, which produces maps with certain types of errors.

This work was submitted to the IEEE Transactions on Knowledge and Data Engineering (TKDE) in September, 2019. It received minor revision decision in January, 2020 and is currently being revised.

1.4.2 Map-Matching Algorithms

According to Section 1.3.2, in our thesis, we introduce the recent map-matching methods proposed since the last comprehensive survey [103] in 2007 and propose a new categorisation that classifies them according to their map-matching models, working scenarios and input data features. Besides, we decompose each algorithm into a core map-matching model and tuning techniques, which can better differentiate the existing solutions and identify the main contributor to their respective performance. Our proposed categorisation can better distinguish the existing methods from the technical perspective and apply to all existing methods which are impossible for previous categorisations. In addition to the survey, we conduct extensive experiments on representative algorithms of different categories and compare their strengths/weaknesses under both online/offline working scenarios. Our experiments mainly provide insight for (1) finding the best trade-off between online matching accuracy and latency, (2) evaluating the influence of input trajectory quality and map density to the map-matching performance and (3) testing the effectiveness of the existing tuning techniques. Finally, we analyse the common weaknesses and remaining challenges of the existing map-matching solutions and demonstrate them through visualisation and experiments. Furthermore, we further discuss the reasons and potential fixes for future research guidance.

The survey part of this work was published at the Australasian Database Conference (ADC) 2020. The full version is under submission to the World Wide Web: Internet and Web Information Systems (WWWJ), 2020.

1.4.3 A Co-optimisation Framework

Combining the solutions mentioned in Section 1.3.1, 1.3.2 and our objectives in Section 1.3.3, in our thesis, we propose an iteration-based map-trajectory co-optimisation framework to improve the quality of both maps and trajectories. To fulfil this goal, we propose quality measures to quantify the map-matching quality and the map quality, respectively, and our goal is to maximise the quality score. In fact, to guarantee the performance improvement, the most crucial part of our framework is to ensure the correctness of new map updates. Therefore, we propose two scores, namely confidence score and influence score, for road correctness evaluation. For each new road, the confidence score evaluates our confidence in inferring it, while the influence score measures its contribution to the improvement of map-matching results. Experimental results show that these scores can help better identify correct road updates over other outliers and meanwhile detect one-way roads. Regarding the framework, our iterative framework consists of a map-matching phase, a map update phase and our proposed co-optimisation model. Through the road filtering process and the control of quality measurements, the framework can guarantee a gradual improvement on both the map and trajectory-matching quality and overall superior performance after the iteration. Moreover, our framework is generalised to support existing map-matching/map inference/update methods and achieve even better quality results. Besides, we utilise a spatial index on trajectories to boost the process while preserving its correctness. Overall, the experiments on multiple-scale real datasets show the quality improvement upon the state-of-the-art map update methods with affordable overheads after being plugged into our framework. In addition to the quality improvement, our method can also run in a reasonable time compared with the existing map inference/update methods.

A portion of this work was published at the International Conference on Database Systems for Advanced Applications (DASFAA), 2019. The full version was published at the World Wide Web:

Internet and Web Information Systems (WWWJ), 2019.

1.4.4 Map Service Platform

Given the motivation mentioned in Section 1.3.4, we develop a map service platform that supports data preprocessing/cleaning processes, including map-matching, map inference and our proposed map-trajectory co-optimisation. To the best of our knowledge, our proposed platform is the first open-source system that both supports map-matching and map inference/update processes with multiple solutions. The platform enables the data quality improvement solely on GPS trajectories (map-matching), map (map inference/update) or on both datasets simultaneously (map-trajectory co-optimisation). Furthermore, the variety of options of different map-matching/map inference solutions ensures the best data preprocessing performance under different data inputs and provides a general platform for easy performance comparison for future research. In addition to the basic functionality, our proposed platform provides abundant toolkits for data conversion, data cleaning, data visualisation and data quality evaluation for both GPS trajectories and maps. Moreover, we implement various spatial tools, like different spatial indices (R-tree, grid, kd-tree, etc.), different coordination systems (WGS84 [142], GCJ-02 [141] and Mercator projection [140]) and different shortest-path algorithms (Dijkstra, A*, etc.), for ease-of-use and future research convenience. The platform is designed and implemented under clear modularisation, which supports easy plug-in for multiple programming languages (Java, Python, Go, etc.) and easy extensions. Therefore, future research on map-matching and map inference/update can be introduced and compared with existing solutions easily.

This project is released on Github (`https://github.com/Hellisk/map-service`) as a public repository.

1.5 Thesis Organisation

The rest of this thesis is organised as follows. In Chapter 2, we review the literature related to our topics. Chapter 3 introduces our study on map inference algorithms. Likewise, the survey and experimental study of the map-matching algorithm is presented in Chapter 4. In Chapter 5, we propose our map-trajectory co-optimisation framework, which utilises both map-matching and map inference algorithms. The map service platform is then elaborated on in Chapter 6. Finally, we conclude the thesis and discuss future works in Chapter 7.

Chapter 2

Literature Review

As aforementioned, the thesis mainly focusses on three research topics: map-matching, map inference and map updates. In fact, as the data preparation step of many downstream applications, all three topics are closely related to data quality problems occurring in trajectory and map data. Therefore, in this section, we will review the related literature of the following topics:

- **Data Quality Issues:** Data quality issues occur in both GPS trajectory and map data, which affects the processes of map-matching, map inference and map updates in different ways. Therefore, we first define these two data types and introduce their main features and quality issues.
- **Map-Matching:** In Chapter 4, we conduct a comprehensive survey on the existing map-matching algorithms. Therefore, in this section, we mainly introduce the existing surveys on these topics and the relationship between map-matching and other topics mentioned in our thesis.
- **Map Inference:** There are several map inference surveys conducted recently. To emphasise the importance of our experimental study elaborated in Chapter 3, we discuss the existing surveys in this section.
- **Map Update:** The map update can be regarded as a special case of map inference, so, in this section, we mainly explain the main idea of the map update and its relation to the map inference, the major works and their weaknesses.

2.1 Data Quality Issues

There are two types of data relevant to our research, namely spatial trajectories and road networks (also known as maps), which are defined in Section 1.2. In fact, both of the datasets contain quality issues when they are obtained. In this subsection, we mainly introduce the characteristics, the main causes of those quality issues and related research.

2.1.1 Spatial Trajectories and Trajectory Quality

Nowadays, various types of positioning systems, like the Global Positioning Systems (GPS), Wi-Fi, Bluetooth and cellular radio, have been vastly integrated into various types of portable devices, including vehicles, mobile phones, electric watches and other wearable equipment. These devices generate a wealth of trajectory data that record vehicle and pedestrian movement. Amongst all other alternatives, GPS data is the most popular data source due to its coverage and data quality. Ideally, a GPS trajectory should precisely describe the object's location. However, due to the low precision of the GPS devices [76], the trajectory points are always sampled inaccurately, which becomes the main challenge for all trajectory-based applications.

To better understand GPS errors, we first introduce the mechanism of the Global Positioning System. The GPS is comprised of 24 satellites at a semi-synchronous altitude around the earth transmitting signals to GPS devices. Since each GPS device communicates with multiple satellites simultaneously and their pairwise distances (device to satellite, satellites to satellite) are measurable, the device's position can be estimated after a complex calculation. As communication and calculations happen periodically, the device's location is sampled and stored discretely. In general, each GPS positioning sample consists of four basic values: the $< x, y, z, t >$, where $< x, y, z >$ represent the geographical coordinates and t marks the timestamp when the position is sampled. However, the altitude z is usually omitted due to the lack of a three-dimensional underlying map and the hardness of calculating point-wise distance on 3D terrain surface [138]. In addition to the coordinates, most of the current GPS devices also contain speed and heading information for each sample, which is obtained from the gyroscope embedded alongside the GPS module. However, to sample the object's location, the GPS device must receive position measures from at least four satellites at the same time [87]. More importantly, the accuracy of the sample is proportionate to the number of satellites connected simultaneously, which is expounded in [76]. Since the signal between GPS devices and satellites can be blocked or reflected by the object's surrounding environment, the quality of the sampled trajectory points is quite unstable. In general, the current GPS data contains two types of errors [68, 126]:

- **Measurement Error**: In a GPS, the location of a device is calculated by its relative distance to multiple satellites. However, since the number and the stability of satellite connections oscillate from time to time, the sampled trajectory point usually lands in an arbitrary place near the actual location. According to the study from Leick et. al [76], although the accuracy of GPS positioning can be affected significantly in certain situations, in most cases, the error is deemed to follow the Gaussian distribution [126] $N(\mu, \sigma^2)$, whose $\mu = 0$ and $\sigma \in (0, 50]$ are determined by the accuracy of positioning system and the operational environment (signal blocked/reflected by complex road infrastructure and surrounding buildings).

- **Sampling Error**: Although an object is moving continuously, its position can only be sampled periodically by the GPS device. Hence, the frequency of GPS position sampling, which is called the sampling rate, is an influential factor in the accuracy of GPS trajectory representation. For example, considering a vehicle runs at a speed less than 50km/h and is sampled every 30 seconds, the maximum travelled distance between two consecutive samples p_1 and p_2 is already 417 meters. Therefore, instead of the straight-line representation $\overline{p_1p_2}$, the actual trajectory over the past 30 seconds is missing. The sampling frequency is measured by its sampling rate, denoted by Δt, which is equivalent to the maximum time interval between two consecutive samples. Usually, the GPS trajectories are sampled periodically, and we regard the trajectory whose $\Delta t \leq 30$ as a high-sampling-rate trajectory while $\Delta t > 30$ as a low sampling rate. However, it is also common in practice when the sampling interval is not fixed due to signal loss or trajectory compression.

As shown in Fig. 2.1, these two types of errors combined make the trajectory (red) much different from its actual route (cyan). Apart from GPS trajectories, other positioning systems also generate similar types of errors with different scales. For instance, with the help of a ground-based calibration system, the DGPS has a much smaller measurement error of roughly 1-3 meters but the same sampling error. The measurement error of WiFi is around the same level as DGPS but with a much lower sampling rate due to the rarity of outdoor WiFi devices. On the contrary, a cellular network has more significant measurement error (≈ 100 meter) but a much higher sampling rate [108].

2.1.2 Maps and Map Quality

In general, a digital map is a virtual image that contains a graph structure representing the road network, including the major road arteries and other points of interest. In our study, we only focus on the road layout on the map, so it is regarded as a road network. The road network is a highly geographical graph, storing actual locations of intersections, shapes of road segments, the length of paths, etc. Different from general graphs, a map used in practice can have multiple variations:

- **Planar/Non-planar**: A planar map is defined as a graph that can be mapped on a plane such that a point crossed by at least two edges are guaranteed to be a vertex. Inversely, graphs that cannot

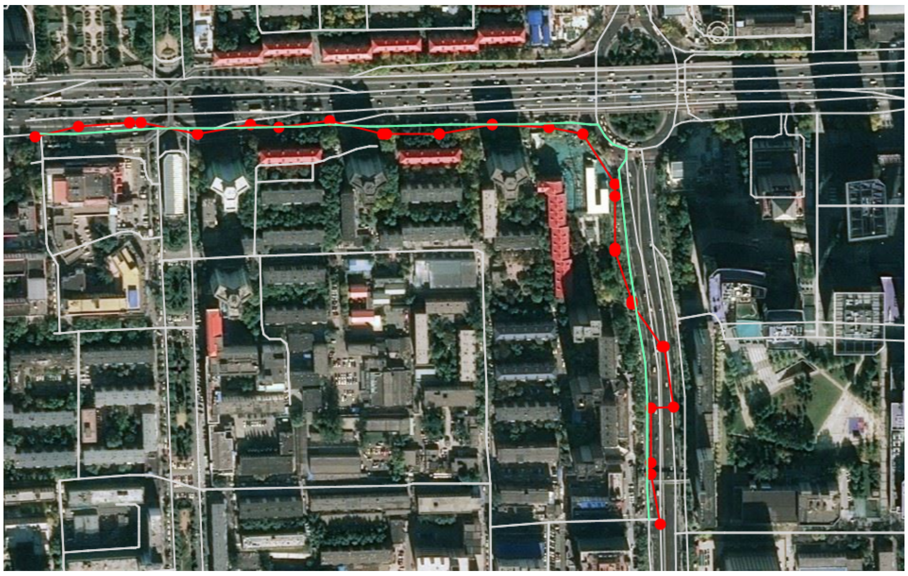


FIGURE 2.1: Example of travel route and sampled trajectory

be mapped on a plane are called non-planar graphs. Intuitively, a road network should be a non-planar graph since plenty of tunnels and interchanges allow one road to underpass another. Also, the experiment results from [46] show that this type of edge crossing happens frequently. However, it is quite challenging to detect such uncrossed roads in some of the map inference methods, especially the grid-based methods [18, 37], which will be introduced in Chapter 3.

- **Weighted/Unweighted Graph**: A weighted graph is a graph whose elements are given numerical weight. In a road network, both vertices and edges can be weighted. Ordinarily, the road length can be calculated for each road so that the distance between two vertices can be calculated. Besides, the weights of graph elements can be other values, like the capacity of traffic flow, travel time and traffic loads, to characterise specific topological features of a transport system.

- **Directed/Undirected Graph**: A graph is directed when edges are oriented from one vertex to another, while the edges in an undirected graph are bidirectional. Intuitively, most of the road networks are naturally directed graphs as plenty of roads are shown as one-way roads. However, a directed graph is more complicated than an undirected graph in terms of solving various of graph problems, like diameter calculation [35], graph partitioning [9], etc. For simplicity, some works use an undirected graph instead when unidirectional edges only take up a small portion of the network. Additionally, in map inference problem, it is difficult for some of the inference methods [18, 37] to identify the direction of each road. Therefore, they only generate an undirected road network instead.

Similar to the trajectory, the map data is also facing different quality issues. Optimally, a correct map should satisfy the following three features:

- **Accuracy**: The map is required to be accurate, which means the map should represent the real-world topological structure and geographical location precisely. In other words, for a road network G, all the vertices in G should have the correct location information; moreover, all edges in G should have the corresponding real-world links between vertices, and the shape of the links should remain the same.
- **Completeness**: The map is required to have accurate and complete attributes. The elements in a map require not only the geographical information but also other attributes, like the speed limit, road width, turning restriction of an intersection, etc. All these attributes are useful in many map-based applications so that their accuracy and completeness is crucial.
- **Recency**: The map is required to be up-to-date. In fact, change to the roads happens frequently due to rapid construction of roads, road maintenance and so on. Therefore, the update of maps should be done accordingly with a short delay so that the analyses can reflect the real-time scenario.

Amongst these three issues, the recency issue is the most difficult one due to the sophisticated process and expensive cost of map updates. Therefore, most of the real-world road map is not fully precise due to the low map update frequency. According to an online report [93], although some open maps, like the OpenCycleMap (the cyclist map in OpenStreetMap [94]), updates every few days, most of the updates for commercial maps are not that frequent, ranging from weekly and monthly to quarterly and yearly. The update frequency is tightly related to the cost/benefit efficiency of the update, which varies depending on what sources are used:

- **Ground Survey**: The ground survey is the typical way of updating maps, which is performed by a mapper, on foot, on bicycle or in a car or other vehicles carrying a GPS unit. It is the most reliable way of updating maps and is commonly used in commercial map maintenance due to their high map quality requirement. For instance, Google Maps and Google Street View send camera cars to capture the road feature and take images for visualisation and measurement purposes. However, sending vehicles to scan all road segments is an expensive and time-consuming process. Thus the update frequency is very low.
- **Satellite Imagery**: The images taken by satellites can be the auxiliary resources for map calibration. The high-resolution images can depict the road width, centre line and other road

features. However, processing the high-resolution image is time-consuming, and the weather conditions can affect the image quality, so it is not feasible to be the major source for a map update.

- **GPS Trace Data**: The GPS trajectory data is another main source for the map update. As the vehicles are running on the road equipped with the GPS devices every day, a huge number of GPS trajectories are captured, and they cover most of the existing roads. Therefore it is viable to calibrate the map accordingly. For instance, the OpenStreetMap [94], which is the world's largest open-licensed map, is updated based on users' GPS trajectories. Initially, the map was generated from scratch by volunteers performing systematic ground surveys, then it became editable to end-users who can modify the map by providing their own GPS records as evidence. Although with the help of end-users, this update approach is usually fast and frequent. However, the quality of the updates is not guaranteed as the GPS data itself contains lots of noise caused by the systematic deviation of GPS devices and other factors, like weather conditions, tall buildings nearby, etc.

2.2 Map-Matching

Map-matching algorithms aim to find the object's moving trace by aligning its trajectory to the underlying road network. Hence, the input of a map-matching process consists of both the trajectory and the road network, while the output is a sequence of road edges representing the object's actual travel path. As defined above, a trajectory Tr is regarded as a sequence of 2-dimensional points (optionally with *speed* and *heading* attributes), while the road network $G = (V, E)$ is stored as a weighted graph whose edge weights are measured by their length. The output of map-matching is usually represented as a *route*, which is a sequence of connected edges representing the actual path the object takes.

According to the problem statement, the main goal of the map-matching process is to find the object's actual locations given its trajectory samples, which is unnecessary if the object is always sampled accurately and the map is correct. However, unless the map makes an error that happens less frequently and can be addressed by map inference and update process [5, 26, 27, 134], the quality issues in trajectories are pervasive, which makes map-matching a challenging task.

The research on the map-matching problem started in the early 1990's [14]. According to the applications, the existing map-matching solutions run on either online or offline scenarios. In online map-matching, the vehicle positions are sampled continuously and are processed in a streaming fashion, which means each time the map-matching is only performed on the current sample, optionally with a limited number of preceding or succeeding (which incurs latency [50, 129, 156]) samples as references. Besides, the process is done quickly to ensure interactive performance. On the contrary, the offline map-matching is performed on the entire trajectory to find the globally optimal matching result with no constraint on the efficiency. Therefore, the online/offline map-matching is also termed as incremental/global map-matching in some papers [50, 74, 135]. Table 2.1 summarises the major differences:

TABLE 2.1: Differences between online and offline map-matching scenario

Matching mode	**Online mode**	**Offline mode**
Objective	Find the matching road of the current location	Find the matching route of a given trajectory
Applications	Navigation, location sharing autonomous driving	Vehicle tracking, traffic management map update
Trajectory feature	(x, y, t) + additional features	Usually (x, y, t)
Sampling rate	Very High ($< 10s$)	High ($< 30s$) or low (1 $5min$)
Reference	Some or all preceding points	Entire trajectory
Matching quality	Fast but less accurate	Slow but more accurate
Route continuity	Not guaranteed	Guaranteed unless break occurred
Major challenges	Initial point matching Road change detection	Break point process Uncertainty modelling

According to Table 2.1, since the online map-matching cannot foresee the succeeding samples, it usually requires additional features, such as altitude z, speed spd, and heading θ, for better matching quality. However, regardless of online and offline scenarios, the difficulty of map-matching problems mainly depends on the quality of the input trajectories. For example, the map-matching on a high sampling-rate DGPS trajectory (less than 3m error radius and less than 5s sample interval) can be trivially achieved by matching each point to its closest road. In fact, many early solutions that claimed to have outstanding map-matching accuracy ($> 95\%$ [92, 102] correct road identification) are mostly done on high accuracy trajectories (DGPS [21, 41]) and/or high-sampling-rate trajectories [92, 102], but their performance deteriorates quickly when lowering the accuracy or sampling frequency. Hence, the current research on map-matching problems focusses on more challenging tasks: (1) achieving high performance on low quality inputs (low sampling rate, less trajectory features and low measurement accuracy), (2) enabling map-matching on low-quality data types (WiFi, cellular network and Bluetooth [88]) and (3) performing finer-grained map-matching (lane-level [41, 71, 86]) on trajectories.

Overall, several papers have been proposed to survey the existing map-matching algorithms, shown in Table 2.2. Quddus et al. [103] give the first comprehensive review in this field. The paper classifies the methods according to their matching principle (*geometric*, *topology*) and technical framework (*probabilistic*, *advanced*). Besides, the paper enumerates the performance claimed in those papers and further discusses the remaining challenges in this field, like map quality, matching at Y-junctions and parameter tuning. Although those challenges became the main focus of the subsequent works, the proposed categorisation fails to classify new methods due to the lack of completeness and has become out-dated. A few other surveys proposed afterwards review the methods from different perspectives. Hashemi et al. [56] give a very detailed overview of online map-matching algorithms. Considering their simplicity and poor performance, the paper merges the *geometry* and *topology* categories into *simple* solutions. It also proposes a *weight-based* category, which is equivalent to *probability* in [103], and adds various recent solutions to the *advanced* class. Kubička et al. classify the solutions based on their applications [74], namely *navigation*, *tracking* and *mapping*. However, none of the surveys categorise existing methods in a comprehensive and informative way. They either propose categories that are obsolete due to underwhelming performance, like *geometric* [103], or provide categories that are not distinctive to each other (many methods are used in both *navigation* and *tracking* [74] and *advanced* category [56, 103] cannot be clearly defined), which is the motivation of our work introduced in Chapter 4

TABLE 2.2: Existing map-matching surveys

Author	**Year**	**Main focus/features**	**Weakness**
Quddus et. al. [103]	2007	• First comprehensive review • Algorithm categorisation • Main challenges analysis	• Obsolete categorisation • No evaluation
Hashemi et al. [56]	2014	• Online map-matching only • Adjusted categorisation	• Missing offline scenario • Categorisation too simple
Kubička et al. [74]	2018	• Application-based categorisation	• No technical differences between categories

2.3 Map Inference

Map-matching algorithms not only identify the vehicle trajectories to the corresponding road network but also improves the accuracy of those trajectory points, provided the road maps are correct [92]. This means the trajectory matching heavily relies on the quality of the road map regardless of what map-matching algorithm is used. Unfortunately, as aforementioned, the quality of road map data is not satisfactory due to the low update frequency, not to mention that many regions around the world are still lacking a usable map. Since map construction/updates were previously done by extensive ground surveys and labour-intensive post-processing, which were expensive and less efficient, many researchers are looking for an effective and efficient algorithm that automatically constructs and/or updates maps.

The current automatic map inference algorithms are mainly built upon two types of data sources:

the aerial imagery and the GPS trajectory. The map inference from aerial imagery tries to identify the roads in the input aerial images using the techniques of image processing and pattern recognition. It has been an active research topic and recent trend leads to deep learning solutions, like Convolutional Neural Network (CNN) [31], for better detection performance. The main challenge for image-based map inference is that many of the roads are partially or completely occluded by tree canopies, the shade of the tall buildings, bridges and interchanges. Also, the images are not updated frequently, making timely map updates impossible. On the contrary, GPS trajectories do not have the continuity issue. Moreover, as plenty of trajectories are generated every day, they enable timely new road detection. Therefore, some recent works combine both aerial imagery and GPS trajectory data for map inference. One simple but effective solution is to add trajectory points as a feature layer to the existing deep learning model for aerial imagery inference.

On the other hand, map inference on GPS trajectories has been studied for 20 years and is still a hot topic [24, 78, 161]. Intuitively, as all vehicles are running on the actual roads, their GPS records combined can somehow sketch the road network both topologically and geographically. Ideally, it can eliminate the cost of ground surveys and manual checks. However, the quality of map inference is strongly affected by the quality of GPS trajectories. As mentioned above, the GPS trajectory data suffer from measurement errors and sampling errors. In addition to the GPS errors, another problem of trajectory data, **trajectory disparity** (also known as **trajectory sparsity**), affects the performance of the map inference process as well, which is caused by different popularities of roads. The trajectory disparity problem is a common challenge for many different trajectory applications, like destination prediction [148, 149, 150, 154] and travel cost estimation [34, 151] Regarding the map inference problem, a road that is never travelled is impossible to be inferred as it does not appear in the trajectory dataset. Hence, the trajectory disparity makes the rarely travelled road harder to be inferred.

To the best of our knowledge, currently, three papers [4, 17, 55] and a book [5] are published surveying the existing methods. Regarding the surveys, Biagioni et al. [17] first reviewed the map inference methods in 2012, and the author classifies the existing methods into three categories, namely *k-means clustering* [43, 54], *Kernel Density Estimation (KDE)* [1, 37] and *trace merging* [25, 91]. The authors choose three representative solutions [25, 37, 43] from the categories and compare their inference results through both visualisation and a new quantitative measure proposed in the paper. Results show that the KDE-based methods have overall better accuracy and much lower running time, which is one of the motivations of their follow-up work on the KDE method [18]. [4] and [5] combine the k-means and KDE into clustering-based method category. They add another recent work [6] to the trace merging and propose the intersection linking [69] category. They also summarise the existing quantitative measures, including the *graph sampling* proposed in Biagioni's survey [17], direct Hausdorff distance and path [3]/shortest-path-based [69] distances. Seven representative methods are introduced in detail and compared visually, which is by far the most comprehensive work in surveying the map inference methods. Also, as they share their implementation publicly, it enables subsequent authors to easily compare their solution with the representatives. However, since they focus more on the visual comparison, and the quantitative measures [3, 17, 69] are only conducted with three of the algorithms [6, 18, 69] on one small dataset. There is a lack of evidence to draw any conclusion on those candidates and methodologies. Instead of conducting extensive evaluations, [55] briefly summarise the main features of existing methods and publish plenty of datasets, with detailed specifications, for future map inference evaluation. As a great number of new methods [24,38,39,59,70,78,100,116,132,160,161] and new directions, like parallel processing [39,44], online map inference [116] and certain map types [19], have been proposed after the last comprehensive survey, it is worth revisiting the existing categorisation and conducting a more comprehensive experimental study on both new and classical methods to comprehensively contrast their performance in more dimensions.

2.4 Map Update

Unlike map inference, which constructs a map from scratch, map updates aim to modify a given map using the trajectories provided. In general, most of the state-of-the-art map update solutions follow the same pattern [134,143], which consists of three steps: (1) extract unmatched trajectories via map-matching; (2) conduct map inference to generate new road edges; (3) and merge road edges to the existing map. Besides some early algorithms that focussed on map inference but also claim their viability for map updates [43], CrowdAtlas [134] was the first work that dedicated on map update in the recent years. Other methods, including Glue [143], COBWEB [111] and HyMU [133], follow the same idea with slight changes to the map influence and map merge solutions. In general, two main issues remain in map update methods: (1) they all require a time-consuming map-matching step (95% of the total algorithm time as shown in [134]) to extract unmatched parts of the trajectories (account for only 1% of the original trajectory set); however, the matching result is completely wasted; and (2) due to the performance issue in map inference phase, the quality of the updated map components is also questionable, which might bring more spurious road nodes/edges than the number of errors it corrects. Besides some general parameter tuning or threshold-based filtering methods, there is still a lack of road correctness evaluation in existing methods to further filter the noisy output, which is the main reasons for us to propose the map-trajectory co-optimisation method.

2.5 Summary

In this chapter, we introduce the data quality issues occurring in map and GPS trajectory data, the respective characteristics of each error type and the existing solutions addressing them, namely map-matching, map inference and map updates. We explain the main idea of each class of solution and the current research progress. Note that we do not provide detailed summarisation of the algorithms for map inference and map-matching problems as they will be introduced and compared in Chapters 3 and 4, respectively. Instead, we review the existing surveys in those fields and discuss their contributions/weaknesses, which are the motivation of our work.

Chapter 3

Map Inference Algorithms

3.1 Introduction

Nowadays, digital maps have been used as the foundation of various map services, such as navigation, vehicle tracking and location-based advertising. Previously, digital maps were constructed by labour-intensive field survey, which required long processing times and huge labour investment. Hence, most of the map data vendors, like TomTom [124] and Google [51], only focus on constructing and updating maps of urban areas for maximum commercial profit. The recent emergence of Volunteered Geographic Information (VGI) complements maps with the contribution from voluntary users. As a successful example, OpenStreetMap [94] has been the largest free editable map created by VGI. However, since volunteered information may contain inaccurate input or even intentional errors, dedicated editors are still necessary to manually validate the map. Moreover, as the road network changes frequently over time, neither field survey nor volunteered information can update the map timely, which motivates the research of automatic map inference (also known as map construction [5]) and map updates [134].

Currently, two major resources are being used for automatic map inference: aerial imagery [31] and GPS trajectories. Different from aerial imagery inference, which detects the map layout from static satellite images, the map inference on GPS trajectories utilises the users' vehicular traces obtained from GPS-enabled devices. As GPS trajectory datasets have a larger volume than aerial images and contain more information, such as road connectivity, turning restriction and road speed, they help improve the quality of the constructed map. However, since the trajectories obtained from GPS devices are intrinsically inaccurate, inferring maps from the noisy trajectories is still challenging nowadays and has been an ongoing research task.

Several survey papers [4, 17, 55] and a book [5] have been published summarising existing map inference algorithms from different aspects. In particular, Ahmed et al. [4, 5] present a well-rounded introduction to previous algorithms based on their categorisation, together with evaluation measures, experiments and comparison. These works provide us with a great overview of the existing solutions and help new researchers quickly dive into this area with some available codes [5]. However, as they mainly focus on delivering the idea of map inference, most of their experiments are conducted through visualisation. Although some surveys compare the solutions via quantitative measures, they all lack enough candidates [5, 17, 55] and proper reasoning for their measures. Overall, none of the existing work clearly illustrates the advantages of each type of solution, and there is no research on how GPS data quality affects their performance. As plenty of new solutions [44, 78, 100, 131, 132, 147, 161] have emerged since the last survey [5], in this chapter, we briefly review the existing map inference methods and propose a new categorisation. More importantly, we discuss the main factors that affect map inference quality and address the issues in the current quantitative evaluation system. We conduct a comprehensive experimental study on representative algorithms to quantify the advantages of each type of method under different datasets and application scenarios. In general, our contributions over the existing surveys are as follows:

- We conduct a comprehensive experimental study on the existing map inference algorithms. The chosen candidates, including both up-to-date and classical solutions, are more diverse in terms of their methodology. Moreover, by introducing datasets with different scales and characteristics and deploying the solutions under the same platform, we are able to evaluate both their effectiveness and efficiency under different input features, which has never been done by any previous work.

- We study how the input GPS data quality affects the performance of various inference algorithms. To this end, we propose a synthetic dataset generator to simulate different types of GPS errors. We evaluate the candidate algorithms on the synthetic datasets to test the influence of noisy data to the constructed map quality and draw conclusions on what type of noisy trajectories should be removed from the input to achieve better map quality.

- We discuss the difficulty of map inference evaluation and the weaknesses of the current map quality evaluation system. To address the issues of existing quality measures, we elaborate on the possible map quality issues of constructed maps and propose an artificial map generator which produces maps, each of which contains a certain type of errors. Then, the synthetic maps are used to evaluate the effectiveness of the current quantitative measures in identifying different map problems.

3.2 Algorithm Survey

In this section, we summarise and classify the existing map inference methods according to our categorisation. Prior to this, we first introduce the related data formats.

3.2.1 Problem Statement

As introduced in Section 2.3, the map inference is the process of inferring road network using GPS trajectories. Therefore, similar to the definitions in Section 1.2, the input of map inference problem is the trajectory dataset, while the output is a road network.

It is worth noting that, in addition to the basic trajectory definition, many trajectories used in map inference also contains the speed (spd_i) and heading (θ_i) information for each trajectory point p_i. Those attributes are utilised in some of the map inference solutions [70, 116] for better inference accuracy. Regarding the road network, networks generated by different inference algorithms may contain various features. Most of the algorithms [18, 37] and evaluation measures [3] are only able to construct/measure **planar** maps, which require that any two roads traversing each other must generate a vertex. The issue of planar maps is that they fail to represent bridges and tunnels where two roads overlap without connection. In addition, some of the map inference algorithms can only generate **undirected** maps and are unable to infer features like turning restriction and speed limit, which should be remedied by a post-processing step such as trajectory map-matching [103].

3.2.2 Survey and Categorisation

The recent upsurge of map inference solutions has introduced new techniques to this field, which forces a review of the current categorisation. We observe that some solutions improve the existing methods by providing new alternative sub-modules [30, 38], while others belong to neither of the existing categories [59]. Therefore, we propose a new categorisation, which consists of three classes: *road abstraction*, *intersection linking* and *incremental branching*. The major difference between these classes is their map inference procedures. Both *road abstraction* and *intersection linking* take the entire trajectory dataset as input and construct the map in one shot, whereas the *incremental branching* starts from an empty map and incrementally expands the map by inserting one or multiple trajectories at a time. Meanwhile, *road abstraction* aims to directly extract the map skeleton by finding representative points/edges from the trajectories, while the *intersection linking* first identifies the intersections and then connects them based on the knowledge from trajectories. In summary, we present most of the existing map inference algorithms in Table 3.1 based on our classification. In this table, we summarise

TABLE 3.1: Summary of map inference methods

Category	Name	Input Features	Sampling Rate	Map Feature	Evaluation	Evaluated in
Road Abstraction						
Point-based, k-means	Edelkamp03 [43]		high(1-15s)	directed	V	[18, 82, 116]
	Elleuch15 [44]			undirected	V	
	Qiu16 [100]	heading	low(45s)	directed	V, GM	
	Stanojevic2018 [116]	heading	high(1-5s)	directed,planar	V, GS	[59]
Point-based, mean-shift	Chen16 [30]	heading,speed	high(1-5s)	directed,non-planar	V, GS, PD	[116]
	Dorum17 [39]	heading		directed	V, GM	
Point-based, others	Li16 [78]		high(15s)	directed	V, GM	
	Agamennoni11 [1]	heading	high(<3s)	directed	V	
Line-based, KDE	Davies06 [37]		high(1s)	undirected	V	[18, 78, 146]
	Biagioni12 [18]		high(2-6s)	undirected	V, GS, GM	[4, 59, 78, 82, 100, 116, 161]
	Ahmed15 [2]		low(60-180s)	undirected,planar	V	
	Wang15&Dey18 [38, 132]		low(40s)	undirected,planar	V	
Line-based, LDA	Zheng17 [161]		high(1-15s)	undirected	V, GS	
Line-based, others	Liu12 [82, 83]	heading	low(16-60s)	directed	V, GS	
Incremental Branching						
Trace merging	Cao09 [25]			directed	V	[18, 82, 116, 146]
	Niehoefer09 [91]		high(1s)	undirected	V	
	Li12 [79]		low(30s)	directed	V	
	Ahmed12 [6]		low(30s)	undirected	V, GM	[161]
	Buchin [24]			undirected	V	
	Zheng18 [160]		low(10-300s)	undirected	V	
Map expansion	He18 [59]	heading		directed,non-planar	V, GS, PD	
Intersection Linking						
	Fathi10 [48]		high(1-5s)	directed	V	
	Karagiorgou17 [69, 70]	heading,speed	low(15-90s)	directed	V, GM, PD	[161]
	Wu13 [144]	heading	low(30-120s)	directed	V	
	Xie141516 [146, 147]	heading,speed	high(1-5s)	directed	V, GM	
	Wang15Novel [131]	heading	high(1s)	directed	V	

the solutions and their details in terms of input trajectory feature (location only/heading/speed), trajectory sampling rate, output map type, evaluation method (V=visual, GM=Graph Item Matching, GS=Graph Sampling, PD=Path/shortest-path-based Distance; details in Section 3.3) and if they are evaluated or compared in other works. Note that, the input trajectory feature does not include those used in pre-processing or post-processing.

In the following, we will summarise the major differences between these categories. We will also introduce the representative methods in each category and highlight some of them (with aliases) as the candidate in our experiments.

3.2.3 Road Abstraction

The road abstraction regards the regions where more trajectories passing through to be more likely roads or intersections. Given that trajectory measurement errors are usually modelled by 2D Gaussian distribution, there is a higher probability that the trajectory sample is closer to the road centreline it travels. Therefore, the main process of road abstraction is to find the densest areas on the map and extract a road network from them. The clustering technique, which divides the input points/segments into groups and finds the representative point/segment for each group, is the major technique used in this category. Due to its simple idea, plenty of solutions have been proposed under this category with various types of clustering algorithms. Most of the work from k-means clustering [43, 44, 54, 82, 100] and KDE-based clustering [2, 17, 18, 37] mentioned by previous surveys fall into this category, together with some new solutions [30, 39, 161]. The existing road abstraction methods can be further classified based on their input data types (trajectory point/segment) and clustering methods (k-means/KDE/mean-shift/etc.). In general, the road abstraction method mainly contains the following three steps.

Data preparation: An input trajectory can be regarded as either a set of points or a set of trajectory segments. For point clustering, besides the point coordinates, the heading of the point is also utilised in most clustering algorithms [30, 39, 100, 116] to distinguish roads with opposite directions and the intersections. For segment clustering, one of the major concerns is that the trajectory segment cannot correctly depict the object's movement trace when the sampling rate is low, which gets even worse when the object makes a turn around intersections. Hence, some line-based methods [83] remove the segments around intersections by their heading changes while others, especially the KDE-based methods [2, 17, 37], filter out such errors when generating the skeleton map.

Clustering: The purpose of the clustering is to find the representative points/segments from the input points/segments so that the final map can be obtained by linking them. K-means is the most widely used algorithm for point-based clustering [43, 44, 54, 100] due to its simplicity and effectiveness. Starting from a set of random seeds on the map, the algorithm iteratively adds GPS points to "closest" clusters, relocates cluster centres, and removes distant points until no more adding or removing happens. In fact, the performance of the k-means clustering is sensitive to the number of initial seeds and their locations, so [100] first identified the straight lines in the map by clustering the trajectory points using the DBSCAN algorithm and continuously sampled points as seeds alongside the straight lines, which ensures the k-means clusters are close to the road centreline. Mean-shift clustering works in a similar fashion as k-means but is claimed to be less vulnerable to low sampling trajectories [30, 39]. Most of the line-based clustering utilises the idea of image processing. It first rasterises the map region into a grid whose grid cells correspond to pixels in an image. The trajectory segments are then drawn on the map and each cell receives a count of trajectories passing it. As the trajectory segments may distort around intersections due to low sampling rate, a smoothing process, the Kernel Density Estimator (KDE) [2,18,37,38,132], is usually applied to reduce the significance of noisy cells and emphasise the road centreline. Finally, the dense area on the map is regarded as roads and intersections. The simplest way of extracting the road areas is to binarise each cell using a global threshold [37]. However, it does not work well as the trajectories are not evenly distributed on the map. Hence, other works improve the threshold by providing multiple thresholds for different road popularities [18] and road widths [2]. Alternatively, the discrete Morse theory is introduced for better centreline extraction [38, 132] especially for less popular roads. Note that although the KDE-based methods are proven to be effective [78, 82, 100, 161], their output map does not have direction and other map features apart from the road skeleton since only the trajectory locations are preserved after converting the map region into an image.

Road generation: After the representative points/segments are found, the final step is to connect them and form the output map. For point-based clusters, one simple solution utilises the trajectory continuity and connects clusters with consecutive GPS points from one trajectory [43]. Li et al. [78] further improve this idea by calculating a representative point sequence from trajectories that connect the same sets of clusters as a road. Other works directly create roads without the help of trajectory information. The principal curve method, which creates a representative curve from a set of points, is perfect for this scenario [1, 39]. Qiu et al. [100] assumes that most of the real-world roads are near straight, so it connects the nearby clusters with a similar direction and extends the road until the direction changes. For line-based clustering, the density-based image obtained from KDE methods can be further thinned to extract road centrelines using the techniques like spline fitting [37] and skeletonisation [18]. After the map skeleton is obtained, a scan of the image is needed to identify intersections based on the value of nearby pixels and edges to be obtained subsequently.

Overall, the road abstraction algorithms regard trajectory as a set of points or segments. One advantage is its ability for parallel processing [44] as most of the operations are localised. However, one major problem is the potential connectivity issue. As exemplified in Fig. 3.1, two parallel roads can be incorrectly connected if the trajectories are broken down into points/segments. Although some effort has been devoted to rectifying such errors using the knowledge from typical intersection design [30] or additional map-matching in post-processing [18], it is still challenging for such methods to infer non-planar features in the map.

Six methods are introduced as representatives and three of them are included in our experimental study as follows.

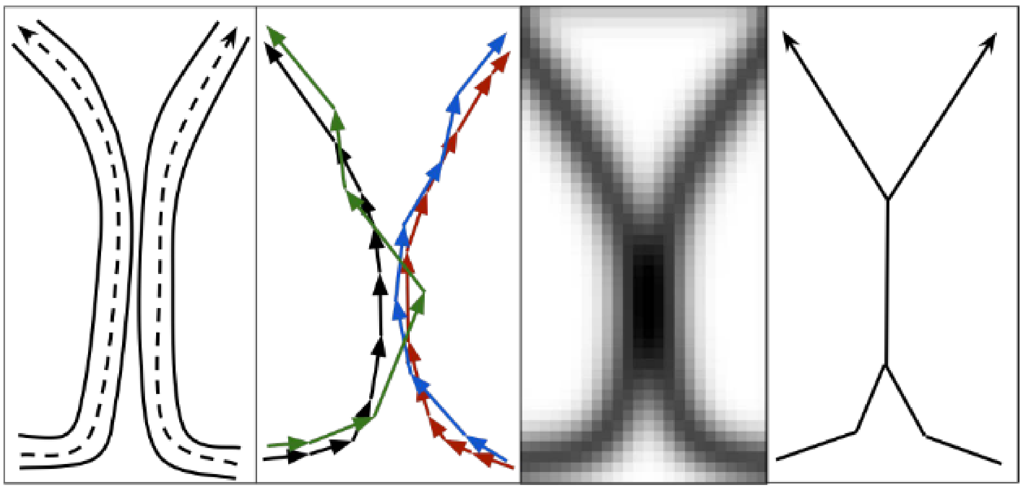


FIGURE 3.1: Example of a wrong connection

Stanojevic and Abbar 2018 (RA-K-MEANS)

The RA-K-MEANS [116] is the most recent k-means solution with a bunch of optimisations. It regards map inference as a multiple network alignment problem (MNAP), which aims to infer the underlying graph given a set of observable sub-graphs (GPS trajectories). For the k-means clustering, the pair-wise point distance considers both geographical distance and heading difference, and each initial seed is selected along the trajectories with no less than a certain distance away from existing seeds. Moreover, to further reduce the inference of spurious roads, a graph spanner is applied during the road generation to remove roads that are potentially generated by noisy or low-sampling-rate trajectories. Besides, it also provides an online model for stream-based map inference and map updates. In general, as the most recent k-means solution, this paper adopts several optimisations on top of the classical k-means clustering [43, 100]. Hence, we use it as a representative of the k-means category.

Li and Kulik 2016

As a point-based method, instead of using *k-means* for clustering, Li et al. [78] proposes a spatial-linear clustering based on the idea that most of the real-world roads are close to a straight line or a combination of several straight lines. Initially, the algorithm starts by randomly selecting a point, named as the *anchor point*. The anchor point gathers all points that are within a certain distance and similar direction and forms a cluster; then, in the current cluster, it finds the furthest points along the cluster's average direction (both forward and backward) to form new *anchor points*. By incrementally applying this process until no *anchor point* is discovered and starts another seed randomly, all the straight roads should be covered by the clusters. Therefore, it connects the clusters according to the trajectories crossing it. Besides, this method claims its high performance and its ability to identify parallel roads.

Davie and Beresford 2006

[37] is the first work proposed in this category. As a density-based map inference algorithm, it first obtains the density for each grid cell. Instead of directly counting the number of passing trajectories, the grid density in this work is defined based on the gross length of the trajectories segments that cross the grid. After blurring the grid values, the algorithm binarises the grids into bit values by a global threshold so that only grids with high density are set to 1. Then, the contour of the bit map is calculated through the contour follower algorithm [157], and the centrelines are extracted by the Voronoi diagram algorithm. Finally, a post-processing step is performed to remove the unconnected centrelines and assign road directions by analysing the direction trend in each grid cell separately. Compared with the follow-up solutions in this category [2, 18, 29, 113], Davie's algorithm filters the grid density by a simple threshold, which can potentially cause a great loss of road coverage or introduce a large number of spurious roads, depending on the threshold chosen. However, although improved by the later methods [17], we still find that some recent experimental results show that it is not outperformed by its follower in some cases, which is the main reason why we choose it as our candidate. Moreover, due to the contour follower and Voronoi diagram algorithms used in centreline generation, roads generated by this algorithm have many zigzags [82], which helps evaluate the performance of the quantitative measures in identifying such road shape problems.

Biagioni and Eriksson 2012 (RA-KDE)

The RA-KDE [18] is a typical KDE-based method with a focus on tackling trajectory disparity problems. Instead of setting a global threshold when binarising the grid density, it proposes a grey-scale multilayer solution so that grids with different densities fall into different density layers. By doing so, roads with fewer visits can be preserved in low-density layers. After smoothing the map by the kernel density estimator, the algorithm generates centrelines from each layer using the skeletonisation method and merges them to form a final map. The algorithm also further post-processes the map to merge unnecessary intersections, remove unmatched roads, simplify road shapes and assign road directions. Currently, the RA-KDE [18] is regarded as the representative method in this category. As shown in Table 3.1, it has been compared the most times with various methods in different categories and performs well in both effectiveness and efficiency.

Zheng and Liu 2017 (RA-LDA)

Although being rasterised into grids, the paper does not leverage the KDE method. Instead, it utilises the grid-trajectory relationship generated during the trajectory projection. Therefore, each trajectory is represented by a set of grid cells. Since the constructed roads are also derived from a set of grids, such a relationship is equivalent to the topic model in NLP, where a document (trajectory) is built up by a set of words (cell) and each word (cell) is related to some topic (road segment). After converting the problem, the paper utilises LDA [22] and pLDA [61], which are two famous models in topic extraction, to extract topics (road segments) from documents (trajectories). In this paper, the idea of solving the map inference problem is quite novel, and the experiments show that it outperforms various map inference solutions, including trace merging [6], density-based algorithm [17] and intersection linking [69] in terms of both accuracy (slightly) and performance (significantly). However, the experiment results are insufficient for a solid conclusion, and there is a big concern about the space complexity of the algorithm as the cell-trajectory Boolean matrix may become very large when the map is huge especially when the cell size is set to a small value (less than $10m^2$). Hence, as a new method inspired by a completely different topic, we plan to conduct more experiments to better compare with the traditional methods.

Liu and Zhu 2012

[83] is the only line-based clustering method that does not rasterise into an image. The algorithm first filters out segments that have significant heading changes between start and end points. Remaining segments are then clustered using single-linkage clustering, which merges nearby segments with similar orientation. It further proposes a Y-split separation method to detect Y-splits (a forked branch from the main road with similar orientation). Eventually, a B-spline fitting is applied to generate centrelines from the clusters and form the final map. In [83], the algorithm is empirically verified to be a good candidate for line-based clustering with low-sampling-rate trajectories.

3.2.4 Incremental Branching

The idea of incremental branching is to incrementally insert new roads to an empty map until all trajectories are examined. Overall, a map can be constructed in two incremental ways:

Trace merging: The trace merging method incrementally inserts trajectories into the map. For a new trajectory, the algorithm first tries to merge it to existing roads by map-matching or distance measure calculation, such as the simple pairwise distance [25,79] or Fréchet distance [6,24,160]. Successfully matched trajectory segments are utilised to adjust road shape by introducing physical attractions [25].

Meanwhile, the unmatched portion either forms a new road from the existing map (partially matched) or creates a completely new road (completely isolated). As one of the early works in this field, Cao et al. [25] merges new trajectories by only considering their segment-based distance and direction difference. Once the merge process is completed, the algorithm defines the physical attraction force between pair-wise points so that the matched roads run towards the newly merged trajectory whereas the unmatched roads run against it. Considering that roads have different lengths in the real world, Buchin et al. [24] proposes a method that predefines multiple road lengths and generates roads with different scales accordingly. This helps to infer small roads that are dominated by nearby main roads.

Map expansion: Map expansion is a new type of method initiated by He et al. [59]. Instead of iterating on the trajectories, it starts from a map node. The major process of map expansion is to incrementally infer new roads from the starting node based on the trajectories passing through it. After the current node is visited, the map has branched out and generated multiple new starting nodes, which are the starting points of new iterations. Different from other incremental solutions, since all the trajectories passing the same node are processed once, it is much easier to infer the outgoing directions correctly. Therefore, it performs better when inferring complex map elements, such as complex intersections, non-planar tunnels and interchanges.

Overall, different from road abstraction which treats each trajectory point/segment separately, the incremental branching methods fully utilise the continuity of an entire trajectory so that neighbouring roads that lead to different destinations can be better separated. However, it is hard for such incremental methods to run in parallel. In addition, the scalability of these iterative methods, especially the trace merging, can be really poor as the size of the input dataset increases. In our experimental study, we choose the following two methods as the representative of this category.

Ahmed and Wenk 2012 (IB-TM)

The IB-TM [6] is the only trace merging solution whose performance has been quantitatively evaluated [5]. The algorithm utilises the Fréchet distance to merge a new trajectory with existing roads. It can efficiently extract matched trajectory parts via a maximum distance threshold ϵ, and generate new representative roads using minimum-link paths algorithm. Unmatched portions are inserted into the map as new roads and connected to the matched road to form a new branch and intersection. This method has a theoretical quality guarantee that the inferred roads are well-separated.

He and Bastani 2018 (IB-ME)

As the only candidate in map expansion, He et al.'s solution [59] mainly focus on high inference precision and non-planar map features, which are the two main weakness of existing methods. Instead of generating a low-accuracy initial graph and refininge it to improve performance, this method aims to infer a partial map correctly without worrying about the recall. Instead, it achieves a high recall by merging the constructed map portion to another map inferred by other high recall methods [17, 116]. As introduced above, the method starts from an initial map node and grows until covering the entire map region. For trajectories that pass a certain active node, the algorithm matches them to the node and its preceding edges, then the node expands according to the directions appearing in the trajectory set. As shown in their results, the solution performs much better than the existing methods in inferring complex regions like overpasses/underpasses, stacked roads, parallel roads and complex intersections as it fully utilises the long-term travel information from the trajectories. Since the inference of complex intersection has been a challenging problem for most of the solutions, we choose it as our candidate for more experimental study.

3.2.5 Intersection Linking

The intersection linking approaches emphasise the correct detection of intersections. Once the intersections are inferred, the remaining step is to link intersections using the information from trajectories. Different from the points on roads, trajectories passing intersections usually have different heading or speed, and more importantly, their headings or speeds may change significantly during a short period. Therefore, the intersections can be identified based on either trajectory movement characteristics (speed, direction) [69, 70, 146] or point density [144]. The main steps include the following:
(1) Scan the input trajectory points and extract the point sequences around intersections, which satisfy the heading and speed requirements (for example, the heading changes more than $15°$ and average speeds under $40km/h$ [69]).
(2) Cluster the intersection points based on their proximity using k-means [131]/random sampling [146] or based on turn similarity [48, 69] to extract the intersection region.
(3) Link intersections connected directly by trajectories. In addition, since each intersection covers a region instead of a point, the actual type of the intersection (crossing, T-junction, roundabout, etc.) is also considered when connecting the edges inside the intersection region.

As the intersections are usually very complex in real life, a correct inference of intersection, especially its turning pattern, can significantly improve the usefulness of the constructed map. Moreover, this can be conducted concurrently, and the machine learning technique, which has been used in road abstraction methods [1, 30], is also applicable. We consider the following algorithm in our experiments.

Karagiorgou and Pfoser 2017 (IL-TURN)

The IL-TURN [69] is a typical intersection linking approach. Intuitively, a vehicle is making a turn at an intersection when its trajectory sees a significant direction change at a low speed. Hence, this method first calculates the direction change for each trajectory point and finds all points likely to be near an intersection (significant direction change and low speed). The turning points are then clustered according to their proximity and a given maximum intersection radius. One intersection is generated from one cluster, and the intersections are finally linked based on the trajectories connecting them. To avoid large intersections, the algorithm is further improved [70] by sorting turning points according to proximity and adaptively deciding the intersection size. Moreover, in the preprocessing step, it categorises the input trajectories according to their speed to infer three maps for different speed level, ranging from highways to low-level lanes, which are conflated to form a final map.

3.3 Quantitative Measure Evaluation

As the performance of map inference algorithms is evaluated by the quality of its constructed map, measuring the map quality is crucial especially when comparing the performance of multiple inference methods. Note that most map inference datasets come with a ground-truth map (usually obtained from public map providers like Google Map and OpenStreetMap) for validation, so the core of the evaluation is to quantify the similarity between the constructed map and the ground-truth. Previously, most of the map inference algorithms demonstrate their performance by visually overlaying their maps with the ground-truth. Despite its simplicity, intuition and usefulness in troubleshooting, visual comparison cannot quantify the map difference, as illustrated in Fig. 3.2, making it hard to compare the performance of different algorithms. Moreover, many road features, such as road direction, turning restriction and road connectivity, are not displayable. Therefore, there is a strong need for quantitative measures in the map inference field.

However, the design of the quantitative measure is challenging. Intuitively, since the ground-truth map is usually obtainable, the quality of a constructed map should be measured based on its differences to the ground-truth. It is obvious that a map with better quality should be more *similar* to the ground-truth. However, the definition of *similarity* is ambiguous. Note that a map is usually represented as a graph, this problem is related to the graph matching problem [45] in graph theory, which is associated with some difficult problems, like subgraph isomorphism (NP-complete), graph

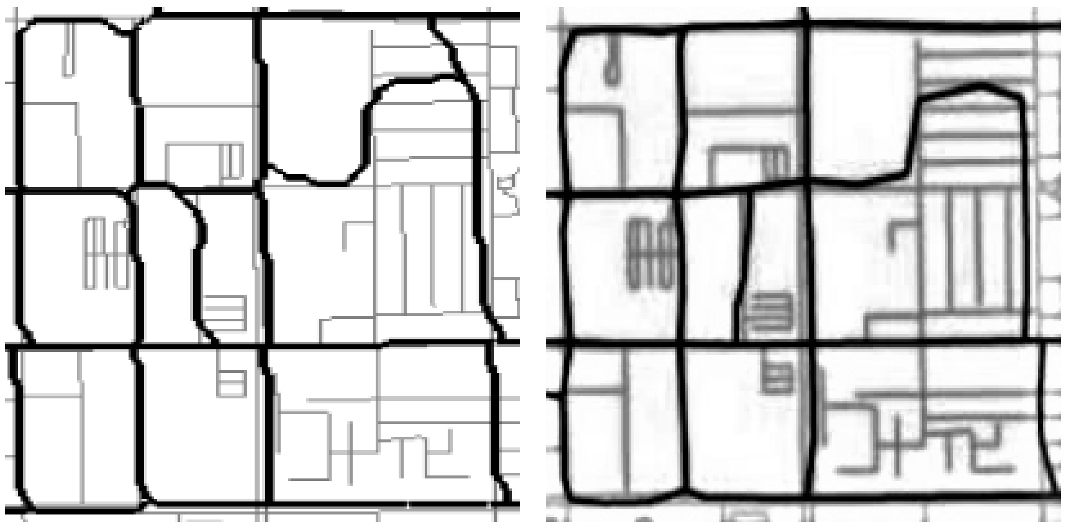


FIGURE 3.2: Visually compare two constructed maps, no clear winner

edit distance (NP-hard) and network alignment. Moreover, Cheong et al. [32] proved that the problem is still NP-hard even with the geometric information embedded. Therefore, all the existing map quality measures evaluate the map similarity heuristically based on various intuitions, such as the vertex/edge closeness [17] and the navigation correctness [3, 69]. However, since each measure only focuses on one or few map features, there is a lack of discussion that treats the map quality as a compounded problem that is determined by multiple factors. In this section, we will first elaborate on the current map quality issues, introduce the state-of-the-art measures, and then propose our design to experimentally evaluate the existing measures.

3.3.1 Map Quality Issues

Note that in this section, the map quality issues we refer to are not the general issues in real-world maps; instead, we focus on the map errors that (1) can possibly be produced by inference algorithms and (2) may cause quality issues when the map is used in applications such as navigation and location-based services. The main features of the applications include the following:

Navigation: In navigation systems, a map with high quality is expected to provide a correct shortest route (or fastest route according to user preference) given a pair of origin-destination locations on the map. One of the fundamental techniques used in navigation is map-matching [74], which aligns a trajectory to the underlying map. In order to find the correct matching route, the map components (intersections/roads) should be close to their actual locations, and their connectivity should also be preserved. Besides, different roads/intersections are not equally important. Roads that are irreplaceable, like the only bridge across the river, are more important to navigational accuracy.

Location-based Service: Location-based services require the map to be more accurate in terms of the location proximity. Since location-based services heavily rely on the spatial queries, such as the range queries and nearest neighbour queries, the location accuracy of the map component is crucial. Unlike the navigation system, map components usually have equal importance in location-based services.

In general, considering the above requirements from the applications and possible errors from existing map inference algorithms, the map quality issues can be categorised as follows:

Topological Error: Topological correctness is crucial for navigation systems. However, it happens quite often that two roads connected in the ground-truth map happen to be disconnected in the constructed map, or vice versa, as exemplified in Fig. 3.1. Moreover, most inference methods [18,37]/map quality measures [3] are only able to generate/evaluate a planar map, which does not match the real-world scenario where transverse roads do not necessarily generate intersections, such as underpass tunnels and interchanges.

Geographical Error: Since the input trajectories suffer from GPS errors, it is common that the inferred road nodes/edges deviate from their actual locations. These errors can affect the correctness of both navigation applications and location-based queries. Besides, the severity of a geographical error is not only determined by its absolute deviation, but also relative distance, which considers the map density of the surrounding area.

Road Loss and Spurious Road: Due to the trajectory disparity [18], many of the roads in ground-truth map are not travelled by any vehicle. Hence, there is no chance for them to be constructed. Also, since the input trajectories contain outliers, many inference algorithms remove the roads constructed from fewer trajectories to avoid generating spurious roads. Therefore, the constructed map usually contains only a subset of roads in the ground-truth map, while the road coverage is an important indicator of algorithm effectiveness. Meanwhile, even with the noise reduction, it is still common to have spurious roads constructed in the map especially around the intersections. Those roads are usually caused by incorrect linking between intersections or non-existent shortcuts. Both road loss and spurious road insertion can affect the navigation and location-based services significantly.

Other Errors: These errors may not affect the quality of map navigation and location-based services considerably, but it can still cause confusion and misunderstanding to map users. For example, many inference algorithms generate roads and intersections with odd shapes, like the zigzag road pattern occurring frequently in the KDE-based method [37] or wrong intersection layout. Besides, it is challenging to infer/measure parallel roads as their distance is usually similar to the GPS error radius. Overall, these errors should also be measured.

3.3.2 Quantitative Measures

Several quantitative measures have been proposed recently for map comparison, which falls into three categories according to their methodologies: graph item matching, graph sampling and path-based distance.

Graph Item Matching (GM)

In this category, the constructed map $G = (V, E)$ is regarded as a set of nodes V and a set of edges E, and the ground-truth map is $G^* = (V^*, E^*)$. Hence, the quantitative measure can be converted to a set similarity problem and answered using the well-known measures of *precision/recall/F-score* as long as the match of nodes ($match(v, v^*)$) and edges ($match(e, e^*)$) is properly defined. In fact, $match(v, v^*)$ and $match(e, e^*)$ are defined differently in various papers. We present a basic definition [70] as follows: for $match(v, v^*)$, Eq. 3.1 defines that two points are matched only when their distance is less than a threshold ϵ; meanwhile, the $match(e, e^*)$ in Eq. 3.2 defines that two edges are matched only when both endpoints are matched, respectively. Based on these definitions, the *precision/recall/F-score* can be evaluated separately for nodes and edges [70] or combined.

$$match(v, v^*) = \begin{cases} true, & \textit{if } dist(v, v^*) \leq \epsilon \\ false, & otherwise, \end{cases} \tag{3.1}$$

$$match(e, e^*) = \begin{cases} true, & \textit{if } match(v_i, v_i^*) \wedge match(v_j, v_j^*) \\ false, & otherwise, \end{cases} \tag{3.2}$$

In general, the graph item matching is the simplest way of evaluating the map quality in terms of road loss and spurious road insertion. However, it also faces several issues: (1) Nodes are not one-to-one matched, which means multiple nodes can be mapped to the same ground-truth node. It can potentially lead to some undetectable connectivity issues. (2) The value of threshold ϵ used for defining node matching may significantly affect the evaluation results. (3) Since the edges are defined by polylines, it is possible that two edges with extremely different shapes may share the same endpoints, which is undetectable in this metric. Although some distance metrics, like Fréchet distance [6] and Hausdorff distance [83], were introduced to further restrict the edge matching to solve the issue (3), it is still hard to measure both the topological and geographical features of the map simultaneously.

Graph Sampling (GS)

This method was first proposed in [17] and adopted in [4, 5, 59, 82, 116, 161]. The main idea is to randomly extract a set of subgraphs from both the constructed and ground-truth maps and compare their similarity. The whole process consists of four steps: (1) randomly select a set of points on the constructed map as seeds and find the corresponding seed points on the ground-truth map, each of which is the closest point on road to a seed location; (2) Traverse the constructed map starting from each seed point and generate a new point, named as *marble*, each time a certain distance is passed. Stop the traversal until a given maximum distance is reached; (3) do the same process for seed points in the ground-truth map, and name the generated points as *holes*; and (4) match the marbles with holes to generate the matching ratio. The graph sampling method is able to detect topological errors, especially its variation (*TOPO* method [116]), as the disconnect edges are not traversed when generating the marbles/holes. Meanwhile, it is also good at evaluating geographical precision through the matching ratio. However, as pointed by Hashemi [55], the performance relies on the number of samples and the traverse distance. It may overestimate the connectivity and incur a huge computational cost if the extracted subgraphs overlap a lot. In addition, since a marble and a hole is a defined match if their pair-wise distance is under the threshold, it cannot detect the road shape errors as long as it is close enough to the ground-truth. The spurious roads may also be hard to be detected through such a process.

Path-based Distance (PD)

This category [3, 59, 69] assesses the map correctness by its navigational performance. In the path-based evaluation, the algorithm first picks up a set of source and destination location pairs, which are randomly selected on the map [69] or from existing trajectories [59], and projects them onto the closest road on both constructed and ground-truth maps. For each pair, the shortest paths on both maps are then obtained and their pair-wise distance is calculated using Discrete Fréchet distance [59,69,70] or average vertical distance [70]. Overall, the map which has less gross pair-wise distance is deemed to be more similar to the ground-truth. This type of methods mainly targets the topology correctness of the map. The map with high correctness in path-based evaluation can better serve navigation purposes. However, since the pair-wise distance is usually dominated by the most significant difference along the path, it has weak power to detect the geographical error, spurious roads and other errors. Moreover, as the quality is measured by a distance value without a proper way of normalisation, it is hard to compare the performance over different algorithms on maps with various scale and density. Moreover, since it requires a large number of sampled pairs to evaluate the road coverage, it can be very expensive computationally considering the complexity of the shortest path calculation.

3.4 Synthetic Data Generator

To evaluate how quantitative measures react to different map quality issues and how data quality affects the map inference methods, we propose two synthetic data generators, namely the **artificial map generator** and the **synthetic trajectory generator**.

3.4.1 Artificial Map Generator

Although introduced above, we think the current study on the quantitative measures is still insufficient: (1) Given that the existing measures are designed for different purposes, no research has studied what types of map quality issues each measure can actually identify. The measures are always used by previous works blindly. (2) Some of the map errors, like the road shape error and wrong intersection layout, are not mentioned by previous works. Hence, they may not be detected by any of the existing measures. (3) Evaluation results from some measures, like the path-based evaluation, are not quite informative as the significance of the distance value does not justify the quality of the map without proper reasoning. Therefore, to evaluate the existing measures regarding their ability in identifying different types of map errors, we propose a synthetic data tool: the artificial map generator.

The idea of the map generator is to generate a set of maps, each of which contains only one specific type of error mentioned above that may occur in a constructed map. Note that the map constructed by map inference algorithms usually contains multiple types of map errors; these synthetic maps can help evaluate how the quantitative measures react to a certain type of error. Given a ground-truth map $G* = (V^*, E^*)$ and a set of trajectories running on it, the map generator produces an artificial map for each error by modifying the ground-truth map respectively as follows.

Topological Error (TE)

The topological error implies the case that an edge exists at the right location without connecting to the right node. To simulate such error, we randomly select a $pct\%$ of roads $E_{te} = e_1, e_2, ..., e_n$ from E^* to be the candidates. For each candidate $e_i \in E_{te}$, we disconnect one of its endpoints, v_a for example, and find a new endpoint $v_{a'}$ on e_i whose distance to v_a is:

$$dist(v_{a'}, v_a) = min(\epsilon, \mu e_i.length) \tag{3.3}$$

where ϵ refers to the GPS measurement error threshold, which is used in various map inference and quantitative measures. $\mu \in (0, 1)$ is a weight that ensures the distance is too long compared with the original road length. Then, we set $v_{a'}$ as the new endpoint, add $v_{a'}$ to V^*, remove the adjacency information in v_a and eventually remove v_a from V^* if it is no longer an intersection. By doing so, the map is barely changed geographically, while the roads are disconnected topologically.

Geographical Error (GE)

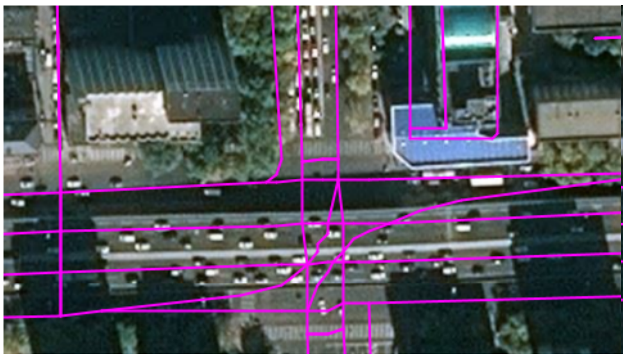
We generate such an error by introducing randomness into the intersections. Here, we first set an error radius r and select a $pct\%$ percent of vertices from V^* as V_{ge}. For each vertex $v_i \in V_{ge}$, we randomly choose a location within the error radius as its new position. Then for each edge in E^* whose endpoint has been shifted, we offset its mini nodes proportionate to the drifted amount to maximally preserve the road shape. Without changing the road connectivity, it is clear that the modified map G_{ge} is topologically isomorphic to G^*.

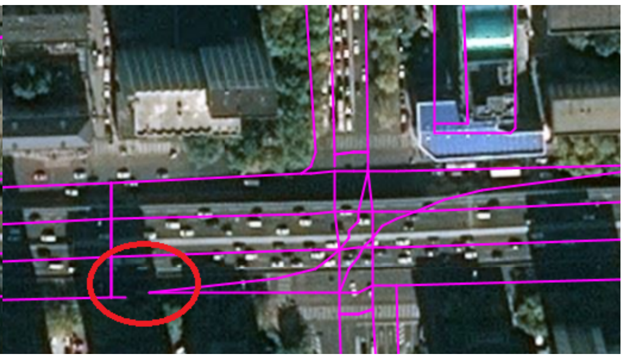
Road Loss (RL) and Spurious Road (SR)

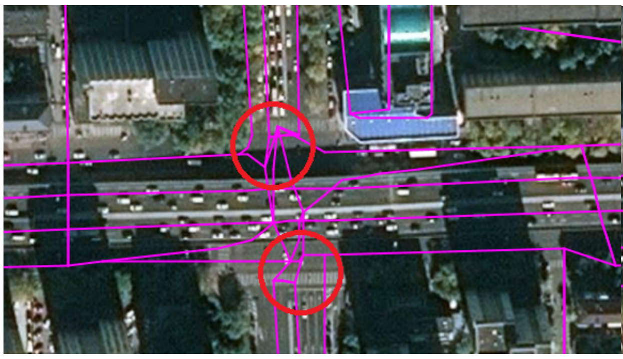
The introduction of the spurious road (SR) requires the use of trajectories. Following the idea that most of the spurious roads are inferred from some noisy trajectories, we aim to find the noisy trajectories by performing the following two steps: (1) we perform a map-matching algorithm [90] on the trajectory dataset and select the trajectories whose matching result contains breakpoints; and (2) for each consecutive breakpoint sequence, we extract the matching roads of its preceding and succeeding points and create a spurious road to connect them directly. We then randomly select a set of spurious roads and insert them into the ground-truth map. The rate of outliers is controlled by the total length of the spurious road inserted. Regarding the road loss (RL) error, we remove a $pct\%$ of the roads according to their total length over the total map length.

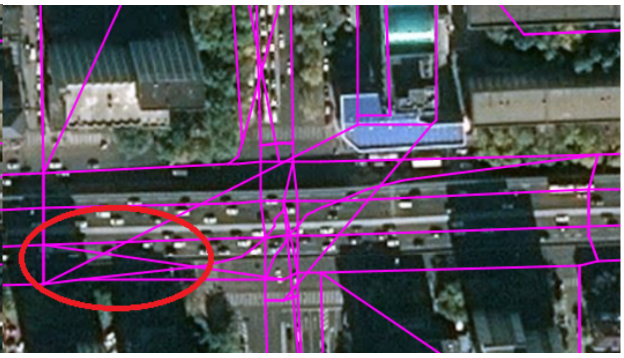
Other errors

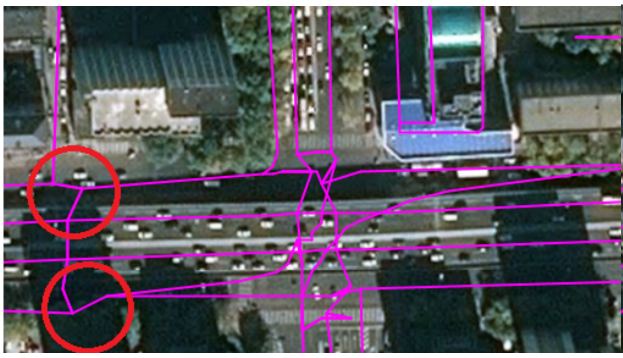
We simulate the zigzag road shape error (RSE) by interpolating mini nodes on the roads. For each road in the ground-truth map, we interpolate a new point p' for every θ meters along the road. In fact, the actual interpolated mini point p is not at p'. Instead, it is chosen randomly with two constraints: (1) $\overline{pp'}$ is perpendicular to the road direction at p'; (2) $\overline{pp'}.length = \theta$ where θ is a distance threshold that is very small to avoid introducing too many geographical errors. Therefore, the severity of the road shape problem is determined by the distance θ, since more interpolated mini points lead to a more serrated road shape. For the intersection layout error (ILE), we create the spurious intersections by splitting the existing intersections. Specifically, we select a $pct\%$ of vertices V_{il} from V^* whose degree is no less than four. For each $v_i \in V_{il}$, we split it into two points v_{i1} and v_{i2} which are symmetric, centred at v_i, and satisfy that $dist(v_{i1}, v_i) = dist(v_{i2}, v_i) \leq \theta$. Lastly, the previously connected edges are redirected to the vertex that is closer and v_{i1} and v_{i2} are connected to each other eventually.


(a) Original intersection


(b) Topological error ($pct = 30\%$)


(c) Geographical error ($pct = 30\%$)


(d) Spurious road ($pct = 30\%$)


(e) Intersection error ($pct = 30\%$)

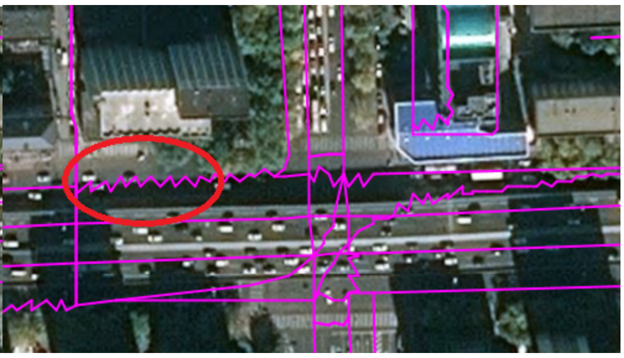

(f) Zigzag road shape ($\theta = 2m$)

FIGURE 3.3: Various map errors generated by synthetic map generator

In Fig. 3.3, we visualise different types of map errors generated by our artificial map generator. Besides, all the $pct\%$ used in the aforementioned road-related cases (TE, RL) requires a random selection, which can be achieved in two modes: complete random (CR) and weighted random (WR). Different from the complete randomness, the weighted random assigns a weight to each edge/vertex so that higher weight items are more likely to be selected. Here, the weight for each edge/vertex is calculated by the following steps: (1) we perform map-matching on the trajectory dataset to the ground-truth map, and each trajectory is converted to a sequence of road edges; (2) for each road edge, we count the total number of appearance in the map-matching results as its weight; (3) the weight of each vertex is the sum of all the weights of its adjacent edges. Therefore, the weighted random considers the popularity of road. Such randomness can help test if the measures are sensitive to road importance.

3.4.2 Synthetic Trajectory Generator

Since GPS trajectories usually contain multiple types of GPS errors, it is impossible to evaluate how a certain type of error affects map inference algorithms. Therefore, we propose a synthetic trajectory generator to simulate different types of GPS trajectory errors, respectively, while preserving the statistical features of the trajectory dataset. To make sure the synthetic datasets follow the same pattern as the actual vehicle movements, instead of generating trajectories randomly, our generator takes the actual road sequence that a vehicle travelled as input to generate a synthetic trajectory. To achieve this, we first conduct map-matching [90] on a real trajectory dataset to obtain the actual travel route of each trajectory (this step can be skipped if the ground-truth map-matching result is available). Then, for each trajectory Tr and its matching result $M(Tr)$, we perform the following process:
(1) For each point $p_i \in Tr$, we find its matching point $M(p_i)$ on $M(Tr)$. $M(p_i)$ is regarded as the estimated location of the vehicle at time t_i.
(2) For each pair of matching points $M(p_i), M(p_{i+1})$ of two consecutive points p_i, p_{i+1}, we find $\Delta t - 1$ intermediate points along the route between $M(p_i)$ and $M(p_{i+1})$ ($\Delta t = t_{i+1} - t_i$) which evenly divide the route into Δt pieces. Therefore, each intermediate point represents the estimated location of the vehicle at a certain timestamp.
(3) We collect all points of $M(p_i)$ and intermediate points to form a point sequence. The sequence is sorted chronologically and form a trajectory, called *primitive trajectory*.

Here, the primitive trajectory is the estimated vehicle locations at every moment during the trip. According to Section 2.3, map inference algorithms could encounter the following types of data quality issues: (1) *inaccurate trajectory point* caused by measurement errors; (2) *low sampling rate* which makes the shape of a trajectory very different from the actual vehicle movement trace; (3) *trajectory disparity* which causes unpopular roads harder to be inferred. To evaluate the impact of these issues on the performance of map inference, our generator produces corresponding synthetic trajectory datasets by manipulating primitive trajectories in certain ways:

- To simulate the GPS measurement errors, we follow a bivariate normal distribution with $N(0, \sigma^2)$

on both axes. By varying the standard deviation σ, we are able to generate trajectories with different measurement accuracies. Hence, for each point in primitive trajectory, we randomly choose a new location according to the distribution and finally form a synthetic trajectory.

- For sampling errors, we re-sample the primitive trajectory by sampling periodically based on the sampling rate Δt.
- As each primitive trajectory is derived from a set of roads, to achieve a certain percentage $pct\%$ of road coverage, we incrementally add primitive trajectory that passes new roads until a certain $pct\%$ of the total map's roads are travelled.

These synthetic datasets will be used in our experiments to evaluate the robustness of the inference algorithms when facing different types of errors.

3.5 Experiments

Our experimental study consists of two main components: (1) evaluate the ability of the current quantitative measures to identify various types of map errors; (2) conduct comprehensive experiments on the representative map inference algorithms and compare their performance under both synthetic and real GPS data.

3.5.1 Experimental Settings

Datasets

There are a few public datasets available, like the *Chicago* [4, 17, 70, 100, 116, 132, 161] and *Berlin* [38, 132] which are widely adopted in existing map inference algorithms. However, they both suffer from severe trajectory disparity issues, as shown in Table 3.2, where the *Chicago* and *Berlin* dataset both have more than 75% of roads that are rarely travelled (96.81% in *Chicago* and 78.60% in *Berlin*). Such datasets are good for visual comparison, because you only need to focus on a certain region of the map, which is how previous surveys conduct their experiments. However, they are not applicable to quantitative experiments as more than 75% of the recall is already lost. Since we mainly focus on quantitative evaluation, in our experiments, we mainly use the *Beijing* dataset, which is a commercial dataset collected by our industry collaborator. It contains trajectories of 5,000 taxis in Beijing for 5 days. From the original map *Beijing-L*, we further extract two sub-areas, namely *Beijing-S* and *Beijing-M*, which represent two urban areas with different scales. Table 3.2 compares these datasets and shows their specifications. In addition to basic information, we also provide three additional statistics for each dataset: (1) *map density*: the average length of roads (km) in a unit of map area (km^2); (2) *trajectory density*: the average number of trajectory points in a unit of map area (km^2); and (3) *trajectory disparity*: the percentage of roads that are never or rarely ($\leq$ 5 times) visited by trajectories. Overall, compared with *Chicago* and *Berlin*, the *Beijing* datasets contain multiple advantages: (1) the variety of map scale and map/trajectory density enables testing algorithms' efficiency and scalability; (2) most of the roads ($> 63\%$) in the urban area (*Beijing-S* and *Beijing-M*) are travelled by at least one trajectory, which is much higher than *Chicago* (3.2%) and Berlin (21.4%); and (3) they provide heading and speed information for each trajectory point, which is required by some of our candidate algorithms.

TABLE 3.2: Comparison of public and private datasets

Name	Input Trajectory			Road Network			Statistics			
	Trajectory Count	*Trajectory Point Count*	*Sampling rate(sec)*	*# of vertices + mini nodes*	*# of edges*	*Area Size(km^2)*	*Map Density (km/km^2)*	*Trajectory Density(pts/km^2)*	*Visit Count $= 0(\%)$*	*Visit Count $\leq 5(\%)$*
Chicago	889	118,360	3.62	9,391	23,602	31.48	38.47	3759.34	94.63%	2.18%
Berlin	27,185	192,194	41.65	5,894	13,678	31.16	22.96	6,168.74	39.70%	38.90%
Beijing-S	55,215	1,329,606	9.67	7,672	4,484	9.92	27.26	134,058.86	9.34%	27.55%
Beijing-M	268,492	8,837,029	11.20	41,353	22,580	56.99	24.18	155,075.62	8.11%	21.50%
Beijing-L	1,994,770	126,929,547	7.66	2,459,768	602,455	33404.56	2.58	3799.77	39.37%	23.69%

Environment

We perform our experiment on a single server, which consists of two Intel(R) Xeon(R) CPU E5-2630 with 10 cores/20 threads at 2.2GHz each, 378GB memory and Ubuntu 16.04. The server has large enough memory to ensure the algorithms can be fully processed in memory. Algorithms run on Python-2.7 (*RA-K-MEANS*, *RA-KDE*, *RA-TOPIC* and *IB-ME*), Java-1.11 (*IB-TM* and *IL-TURN*) and Go-1.12.5 (*IB-ME*)). The reason for algorithms being implemented in different languages is mainly due to their respective dependency requirements. However, it does not affect our experiments as we mainly focus on their self scalability rather than direct efficiency comparison.

3.5.2 Evaluation of Measures

Before comparing the map inference algorithms, we first evaluate the existing quantitative measures in identifying map errors discussed in Section 3.3. Our candidate measures include graph item matching [69] (GM), graph sampling [17] (GS) and path-based distance [3] (PD). We test them on six types of generated maps (GE, TE, RL, SR, RSE, ILE) with two of them using both complete-random and weighted-random sampling strategies (TE-CR, TE-WR, RL-CR, RL-WR). Tables 3.3 and 3.4 show the evaluation results under different error types. Due to the space limit, we only list the F-score for vertex graph item matching (GMV), edge graph item matching (GME) and graph sampling (GS), and we calculate the average Fréchet distance for path-based distance (PD). In general, it is clear that each measure has its weakness in identifying certain types of errors. For instance, it is expected that the vertex-based graph matching is not able to identify the topological error, road loss, spurious roads and road shape errors as the intersections remain unchanged in those maps. However, it is also less sensitive to the cases where the intersections shift slightly (GE), or are duplicated (ILE) since most of those noises do not escape the radius of vertex match and duplicate intersections can be matched to the same ground-truth without penalty. On the other hand, edge-based graph matching and graph sampling are capable of detecting topological changes. In particular, both measures decrease considerably when more topological errors (TE) or road removal (RL) takes place. Interestingly, the decline becomes less significant when the errors are introduced based on road popularity. The reason is that in the weighted random, we tend to remove a fewer number of roads but with higher importance. However, both GME and GS scores only reflect the reduced number of removals; in other words, these measures do not take into account the importance of different roads.

Overall, the path-based measure is underwhelming in most scenarios, except SR and GE, due to multiple reasons: (1) different from a percentage, an absolute value of distance sometimes cannot measure the difference between two maps, especially when comparing the results between maps with different scaled and road densities; (2) since the measure tries to find the matching ground-truth to each path in a constructed map, it can hardly detect the road coverage and connectivity problems, like RL and TE. Instead, it can better detect spurious roads (SR) rather than other measures as those roads are usually hard to match to ground-truth; and (3) as it compares each generated link to all possible links on the map to find the one with minimum Fréchet/Hausdorff distance, the complexity is at the level of $O(V^3)$ [3]. Due to the high complexity of the Fréchet distance calculation and candidate generation, the measure runs extremely slow even with a medium map size.

In general, the graph sampling and edge-based graph item matching have better overall performance in most error types. Compared with graph item matching, graph sampling is slightly more sensitive to the noise and is very sensitive to low road coverage. However, none of these measures takes into consideration of road importance, which defines how crucial a road is with respect to the road network. It is sometimes given as the road level in road network datasets or calculated based on the centrality indices (degree centrality, betweenness centrality, etc.) in graph analysis. As an incorrect inference of an arterial road can lead to more serious problems than a missing rural road, especially in the navigation system, the lack of sensitivity to road importance can be a critical problem to map quality evaluation. Moreover, they have a hard time identifying intersection layout errors (ILE) and are unable to detect road shape changes. Since both errors happen quite often in various inference algorithms [17, 37], it is clear that the current measurement system still requires more attention and further development.

TABLE 3.3: Measurement results under different topological error (TE) and road loss (RL) ratios

Pct	TE-CR				TE-WR				RL-CR				RL-WR			
(%)	GMV (%)	GME (%)	GS(%)	PD (m)	GMV	GME	GS	PD	GMV	GME	GS	PD	GMV	GME	GS	PD
10	99.35	96.94	91.54	1.07	99.49	97.29	94.88	1.03	98.93	95.76	93.13	1.02	99.54	96.72	96.69	1
20	98.77	94.16	83.31	1.16	99.27	95.20	90.86	1.05	97.97	91.01	86.71	1.01	99.40	93.78	93.78	1
30	98.39	90.89	73.58	1.39	99.22	92.47	85.69	1.09	97.27	86.19	81.88	1.01	99.06	88.56	91.23	1
40	97.91	88.40	65.41	1.60	99.04	89.74	80.20	1.17	96.61	81.25	77.94	1.01	98.64	82.38	88.14	1
50	97.71	85.98	60.92	1.81	98.85	87.03	74.66	1.32	95.92	74.55	76.32	1.01	98.21	74.77	85.27	1
60	97.39	82.66	57.11	2.00	98.64	84.14	68.34	1.54	95.78	68.37	75.06	1.01	97.96	65.35	81.61	1

TABLE 3.4: Measurement results under various other error types

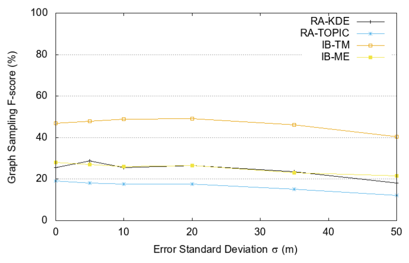
Pct	SR				ILE				σ	GE				θ	RSE			
(%)	GMV	GME	GS	PD	GMV	GME	GS	PD	(m)	GMV	GME	GS	PD	(m)	GMV	GME	GS	PD
5	99.09	97.43	98.42	1.41	1.0	99.77	97.84	1.45	5	1.0	1.0	95.01	6.80	2	1.0	1.0	86.95	13.91
10	98.16	94.95	97.02	2.32	1.0	99.46	95.11	1.88	10	1.0	1.0	91.84	8.12	4	1.0	1.0	87.74	13.37
15	97.75	93.20	95.29	3.33	1.0	99.20	92.27	2.31	15	1.0	1.0	90.74	9.37	6	1.0	1.0	87.75	13.03
20	97.51	91.81	93.80	3.82	1.0	99.14	90.38	2.63	20	1.0	1.0	89.08	10.74	8	1.0	1.0	88.20	12.92
25	97.24	90.49	92.04	4.61	1.0	98.94	87.87	2.88	30	1.0	1.0	85.71	13.48	10	1.0	1.0	88.47	12.93
30	97.00	89.57	90.38	5.50	1.0	98.80	84.53	3.11	40	1.0	1.0	83.09	16.33	12	1.0	1.0	88.70	13.06

3.5.3 Evaluation of Inference Algorithms

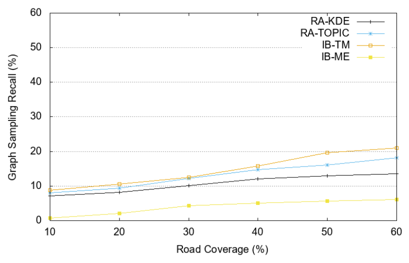
We evaluate our candidate inference algorithms on the *Beijing* dataset using the above quantitative measures. Note that the parameters used in the algorithms are mostly set to default value according to their original paper unless it is related to the expected GPS measurement error range. As all the algorithms assume the input is always inaccurate, they always set a threshold for tolerable error radius which is usually between 0 50 meters. In this case, we set it as 20 meters after we analyse our *Beijing* dataset. Overall, we conduct the experiments from three perspectives: robustness, scalability and overall performance comparison.

Robustness

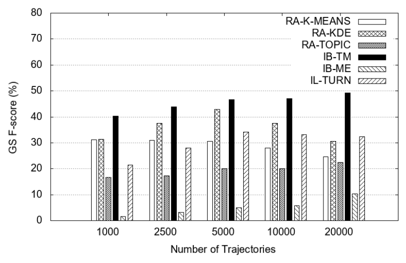
We test the robustness of the algorithms based on how they perform under different qualities of trajectories. Therefore, we generate synthetic trajectory datasets through our generator on *Beijing-S* map, which only contains measurement error, sampling error and trajectory disparity issues, respectively. Note that we do not evaluate RA-K-MEANS and IL-TURN as they require speed and heading as inputs, which is not available in synthetic datasets. Fig. 3.4a shows the inference quality of the candidate algorithms under different GPS measurement accuracies. It is shown that the increase of



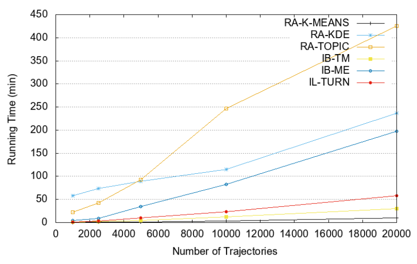
(a) Synthetic measurement error



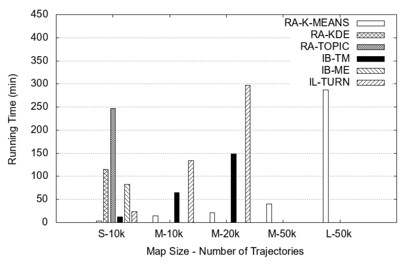
(b) Synthetic road coverage



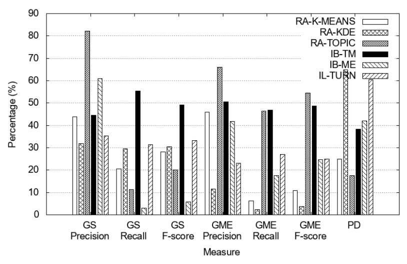
(c) Map quality when varying input size



(d) Running time when varying input size



(e) Running time on different data scales



(f) Map quality comparison (Beijing-S)

FIGURE 3.4: Summary of experimental results of map inference algorithms

measurement error does not affect the overall performance significantly until the error exceeds the preset error radius (20m). The reason is that when a GPS sample exceeds the tolerable error range, it is likely to be treated as a new road element, so its precision drops quickly as the graph sampling measure is sensitive to spurious roads. However, in another experiment we have done, we set the error radius to a higher value (50m) and found that the overall F-score of most algorithms dropped by a visible amount due to a significant decrease in recall. The combined results show that a proper preset error radius that matches the input is crucial. A strict preset value on a low-quality input can cause low inference accuracy, while a loose preset range guarantees an incomplete map, even with high-quality input. Nowadays, as the accuracy of GPS positioning devices keeps improving, the performance of map inference algorithms can also benefit from high accuracy in the future. On the other hand, we also vary the sampling rate while removing the measurement errors entirely. It is clear that the map quality gradually decreases when the sampling rate drops. This happens more seriously to line-based methods, namely RA-KDE and RA-TOPIC, as the lines can no longer represent the road shape when the sampling rate is very low. In terms of the trajectory disparity, we test the algorithms by providing various percentages of the road coverage which contains neither measurement nor sampling error, as shown in Fig. 3.4b. Note that although the road coverage reaches up to 60% in our experiment, most of the roads are travelled on a few times since the input trajectory set is chosen by maximising the road coverage while minimising the input size. As shown in the figure, although we can clearly see the improvement of map recall, only less than half of the roads can be inferred, which means that finding the rarely travelled roads is still a challenging task. Many of them are either removed as noise or merged with neighbouring roads. Amongst all the candidates, the trace merging method has the best detection rate. Since it processes every trajectory individually, it is more likely to find new road segments as long as the trajectory is partially different from the existing roads.

Scalability

The main objective of the scalability test is to answer three questions: (1) how the quality of the constructed map improves as the size of the input trajectory increases; (2) how the efficiency of algorithms scales with the size of trajectory dataset; and (3) how the map size affects the efficiency of algorithms. Figs. 3.4c, 3.4d and 3.4e answer these questions, respectively. In Fig. 3.4c, we use graph sampling F-score to evaluate the map quality. Interestingly, not all algorithms' performance benefits from increasing the trajectory dataset, in particular, the KDE, k-means and intersection linking methods. The main reason is that although the increase of the trajectory size can slightly improve the road coverage, due to the poor denoise mechanism, more spurious roads are also introduced, which affect the precision significantly. For example, in the KDE method, the increase in input size can lead to more ambiguous road intersections, which can potentially create more incorrect branches after the skeletonisation process. Therefore, increasing input dataset is not always a good idea unless the quality of input is ensured by data preprocessing or a more robust algorithm is chosen. The quality of intersection linking suffers from low input size, especially when the trajectory sampling rate is low where the intersections are difficult to detect. Besides, although two of the algorithms have a very poor F-score, namely RA-TOPIC and IB-ME, their actual performance is not as bad as indicated. The main reason for their poor performance is the low recall. Plus, the result from graph sampling measures usually further decreases the score for lower recall solution since most of the seed points from ground-truth cannot find their correspondents on the inferred map. In fact, these two methods can achieve the best accuracy amongst all solutions, making them the perfect choices for larger input dataset thanks to strict noise control. However, during our experiments, we found that IB-ME and IB-TM can barely generate reasonable results when the trajectory size is too low. The reasons are different. For IB-ME, the algorithm always starts from some random locations on the map, and explore the whole map. When the trajectory size is too small, the starting seeds can not find neighbouring trajectories to start with, which prevent the algorithm to generate any result. Meanwhile, in IB-TM, the trajectories are well separated due to the sparsity, which means the final map becomes a set of individual edges rather than a connected graph. However, there is no clear bottom line, in terms of the trajectory size, for each algorithm to generate reasonable map as it determined by multiple factors, like the trajectory length, spatial distribution, map size, etc., but it is better to try algorithms like KDE, k-means and intersection linking when the trajectory size is considerable small.

In terms of efficiency, since our algorithms are implemented in various languages, we mainly focus on their scalability when changing the input size rather than comparing the efficiency between algorithm directly. Fig. 3.4d shows the scalability of those methods with fixed map size and increasing trajectory input. To our surprise, the trace merging algorithm has the second-best scalability amongst all candidates, which shows that the cost of the Fréchet distance calculation between every new trajectory and the map has been fully optimised. On the other hand, the topic model method has the worst scalability since topic extraction algorithms (LDA and pLDA) are all computationally-extensive operations, which rely heavily on iterations. On the contrary, the k-means-based method is very efficient and also scales perfectly. The same applies to Fig. 3.4e where the map size increases to *Beijing-M* and *Beijing-L* with the same or larger trajectory size. Most of the solutions struggle to finish the task within a reasonable time (< 5 hours) even on the *Beijing-M* dataset, but we can see that the RA-K-MEANS methods finish the task on *Beijing-L* efficiently, not to mention its ability for parallel processing which further accelerates the process. From the results in Fig. 3.4e, we can clearly see that even with the same input trajectory size, the map size can significantly affect the performance of all algorithms.

Overall performance

We evaluate the overall performance on 10,000 trajectories in the *Beijing-S* map, shown in Fig. 3.4f. In general, the classical methods, like KDE and K-means methods, has been largely outperformed by recent solutions. However, there is nothing wrong with the clustering methods. Although using the same rasterisation technique, the topic model-based methods achieve much better performance than the KDE method in terms of both accuracy and road coverage, which indicates that the current way of road abstraction from the cluster is still under development. Despite the outstanding efficiency, the accuracy of the K-means method is unsatisfactory. Moreover, since its accuracy does not always improve as the input data grows, its potential for large scale map construction cannot be fully utilised unless some pruning step is introduced. The map expansion method has decent inference accuracy; however, its road coverage is underwhelming. Considering its ability of road feature inference (non-planar map, more accurate road shape), it is an ideal method for map updates. Overall, despite its relatively low recall, the topic model-based method has the best accuracy and overall performance amongst all other methods. Since the idea and solutions of topic model come from another research field, it shows that map inference problems can potentially be solved by ideas from other research areas or through new techniques, like machine learning.

3.5.4 Experiment Findings

As one of the main contribution of our study, we simulate different types of map errors, and evaluate how the current quantitative measurement systems are able to identify them, which were never studied in previous research. In our experiments, we find that despite various quantitative measures are proposed, no measure is capable of identifying all types of errors appeared in an inferred map. Moreover, some measures (path-based measures) are completely dominated by others, and several types of errors are unable to be identified by any of existing measures. Such findings show that the study on map inference evaluation is still much needed in the future.

In terms of the map inference algorithm performance, our experiments show two interesting observations: (1) The increase of input trajectory size does not always lead to better inference accuracy, which is mainly caused by the poor error control mechanism. (2) The size of the map can solely affect the performance of most inference algorithms, and most of the advanced map inference algorithms fail to handle large trajectory dataset. Both insights show that the current map inference algorithms do not benefit from the big data era. Better methods for error control and big data processing are desired, which can be achieved by either better data preprocessing procedures or better inference algorithms in the future.

3.6 Summary

In this chapter, we present a comprehensive survey and experimental study of existing map inference algorithms. Specifically, we propose a new categorisation method, compare the representative algorithms experimentally and evaluate the existing quantitative measures. Besides, to test the robustness of algorithms and quantitative measures, we introduce a synthetic trajectory generator and an artificial map generator to simulate different trajectory errors and map quality issues, respectively. According to our experiments, we observe that besides their respective weakness, the existing quantitative measures are unable to identify several map issues and do not consider road importance. Regarding the candidate algorithms, the existing algorithms struggle to guarantee performance when the GPS errors exceed their expected threshold, and they still find a hard time identifying roads that are rarely travelled. Moreover, more input trajectories do not always lead to better inference results. Overall, we identify the method that has the best scalability (RA-K-MEANS), the best accuracy (RA-TOPIC), and the best suitability for map updates (IB-ME), respectively, and meanwhile point out potential future research directions.

Chapter 4

Map-Matching Algorithms

4.1 Introduction

The recent popularity of GPS-equipped devices provides abundant user/vehicle trajectories. However, as the backbone of most location-based services, the trajectory data suffer from various data quality issues, which lead to false representation of user/vehicle travel history. Hence, in addition to some generic data cleaning techniques, the trajectory map-matching was proposed to correct trajectory errors. By aligning it to the road network, each trajectory is converted to a sequence of roads representing the actual route travelled, which serves as the input of many downstream applications, such as navigation [51], traffic monitoring and map construction/update [5, 27, 134],

The map-matching problem was first studied in late 1990s [14, 75], in which the data from Global Positioning System (GPS) were map-matched to support navigation applications. Since then, plenty of solutions have been proposed during the last two decades. On the one hand, driven by the continuous evolution of positioning systems and devices (GPS [33, 117], Wi-Fi [121], inertial sensor [130], etc.) and the emergence of new applications (autonomous driving [81], map update [134], etc.), various new algorithms have been proposed to support map-matching in different scenarios. On the other hand, as it is known that map-matching becomes more challenging as the input trajectory quality deteriorates, plenty of tuning techniques are applied to the existing models to achieve faster and more accurate map-matching over inaccurate and less-sampled trajectories.

However, to the best of our knowledge, despite the massive number of map-matching algorithms, only a few surveys are found [56, 74, 103, 114, 136] classifying or comparing them. Quddus [103] et al. first summarise the early algorithms in 2007. They categorise the methods based on the matching principles (geometrical/topological) or the computation model (probability/advanced) adopted in the algorithms. Such categorisation was a consensus agreed but is now obsolete as it fails to classify most of the new methods proposed afterwards. Nevertheless, apart from some recent topic-specific surveys [56,74], this categorisation is still widely adopted by recent papers [73,101,118,122] and surveys [114], which indicates the need of a review in this field. Furthermore, recent works bring various new matching frameworks [112,119] and tuning techniques [57,63,81,95] to solve the map-matching problem on new types of positioning data (DGPS, inertial sensor, laser scanner [86], camera [71]) and new queries (lane-level map-matching [41, 71, 86]). Hence, it is worth conducting a comprehensive survey to summarise existing works and discuss the remaining challenges and future research directions. In this chapter, we introduce the existing map-matching methods and propose a new categorisation that classifies them according to their map-matching models, working scenarios and input data features. Besides, we conduct extensive experiments on multiple representative methods with both real and synthetic datasets. In summary, the main contributions include the following:

- We decompose each map-matching solution into a map-matching model and a set of tuning techniques. In addition, we categorise the algorithms based on their matching models, working scenarios and input data features. Our proposed categorisation can better distinguish the existing methods from the technical perspective and is easily adaptable to new solutions.
- We conduct extensive experiments on several representative map-matching algorithms to compare their performance under both online/offline working scenarios. Additionally, our experiments study the problems of (1) finding the best trade-off between online matching accuracy and latency, (2) evaluating the influence of different data features to the map-matching performance and (3) testing the effectiveness of some typical tuning techniques.
- We identify and visualise the matching errors that remain unsolved by the current map-matching algorithm, which is regarded as the future research direction.

Note that our survey only focusses on vehicle trajectory map-matching in the outdoor environment, which is the foundation of a wide range of applications. Other fields, like map-matching in indoor environment [11, 145], for pedestrians [13, 108] or on different data types [88] face various unique challenges that are irrelevant to our focus. Hence, we do not include them in our discussion.

4.2 Preliminaries

Before the detailed survey, we first define the map-matching problem and related data types. The map-matching algorithm aims to find the object's moving trace by aligning its trajectory to the underlying road network. Hence, the input of a map-matching process consists of both the trajectory and the road network, which are defined in Section 1.2, while the output is a sequence of road edges representing the object's actual travel path.

Regarding the input, the format of the input trajectory and road network may have multiple variations. For each GPS trajectory point p_i, in addition to its 2D location, the *speed* spd_i and *heading* θ_i are also measured by inertial sensors which are integrated with most GPS positioning devices, like mobile phones and GPS-equipped vehicles. Other data sources, like the Differential GPS (DGPS) [41, 128], WiFi [121], cellular radio [8, 89] and visual sensors [81], also serve as either primary or auxiliary data sources for map-matching with different level of accuracy and stability. On the other hand, in addition to the basic geographical map features, other features, like road width, number of lanes, speed limit and road type, are utilised in some map-matching papers [50, 156] for better modelling of the user's driving preference or fine-grained lane-based map-matching [71, 112].

In terms of the output travel paths, we define them as routes:

Definition 3. *(Route) A route R on map G represents a sequence of connected edges, i.e. $R : e_1 \rightarrow e_2 \rightarrow ... \rightarrow e_n$, where $e_i \in G.E(1 \leq i \leq n)$ and $e_k.e = e_{k+1}.s(1 \leq k \leq n)$.*

Therefore, we are ready to formally define the map-matching problem:

Definition 4. *(Map-matching) Given a road network $G(V, E)$ and a trajectory Tr, the map-matching finds a route $\mathcal{MR}_G(Tr)$ on G representing the sequence of roads travelled by Tr and each point $p_i \in Tr$ is matched to a point on R representing its actual location at t_i.*

Note that, as the map-matching result $\mathcal{MR}_G(Tr)$ represents the object's travel route, it is usually expected to be continuous. However, it is quite often that $\mathcal{MR}_G(Tr)$ contains disconnected edges due to incorrect map-matching or map errors. Besides, as mentioned in the definition, many papers store the matching result as a set of matching points $\mathcal{MP}_G(Tr) = \{mp_1, mp_2, ..., mp_n\}$ where mp_i is the matching point (actual location on the route when p_i is sampled) of $p_i \in Tr$ and $n = |Tr|$. Note that, for simplicity, we omit the subscript G and use $\mathcal{MR}(Tr)$/$\mathcal{MP}(Tr)$ instead to represent route/point match in the following content as the map-matching of different trajectories is usually done on the same map.

4.3 Map-Matching Models

As mentioned in Section 2.2, previous surveys classify the map-matching algorithms based on the mathematical tools they utilise [103], working scenarios [56] or applications [74]. However, these categorisations fail to classify the current solutions due to three main reasons: (1) categories for some

TABLE 4.1: Categorisation of map-matching algorithms in different scenarios

Category	Model	High Δt		Low Δt		Others
		Online	Offline	Online	Offline	
Similarity	Distance-based		[162]		[137]	[96]
	Pattern-based				[159]	
State-transition	HMM	[47, 121, 129]	[36, 90, 115]	[8, 50, 67, 107]	[95]	[88]
	CRF			[65]	[65]	
	WGT			[84, 156]	[63, 64, 84, 105, 156]	
Candidate-migration	PF		[23]			[123]
	MHT	[73, 109]	[85]	[119]		
Scoring	Naïve weighting	[20, 53, 58, 80, 112, 127]		[101]		[127]

primary methods, such as *geometric* [103], are no longer the focus due to their weak performance. (2) application-based classification [56, 74] cannot fully distinguish the methods. Many of the map-matching algorithms, like the Hidden Markov Model (HMM) and Multiple Hypothesis Technique (MHT), apply to both online and offline scenarios for different applications; and (3) classifying algorithms by embedded mathematical tools is not feasible since many recent algorithms employ multiple mathematical tools. Furthermore, the same tool implemented in different algorithms may be used for different purposes; for example, an extended Kalman filter can be used to either estimate GPS biases or fuse measurements from different sources [80].

According to our observation, the existing map-matching algorithms are comprised by two main components: the core map-matching model and a set of tuning techniques. A map-matching model is a framework that coordinates other techniques to finally achieve map-matching, while the tuning techniques can be applied in various map-matching algorithms for performance improvement or data integration, and each map-matching algorithm can apply multiple tuning techniques for different purposes. In this section, we introduce the existing map-matching models and establish a new categorisation that classifies the map-matching algorithms by their core matching models.

In a map-matching algorithm, the map-matching model is the overall framework or matching principle for the map-matching process. A model usually consists of a set of computational components, like the calculation of distance, transition and user behaviour modelling, and a workflow connecting them. Those components are fixed while their definition and implementation vary amongst different methods. According to our observations, existing map-matching models can be categorised into four classes: *similarity model*, *state-transition model*, *candidate-migration model* and *scoring model*. In addition, we further divide the solutions in each category to multiple subclasses according to their working scenarios (online/offline) and input data features (high/low sampling rate or other data sources), as listed in Table 4.1. Hence, we introduce the core idea of each category and their representative solutions.

4.3.1 Similarity Model

The similarity model focusses on finding the vertices/edges that are *closest* to the trajectory geometrically and/or topologically. Intuitively, the trajectory should have a similar shape to the actual route when the data quality is decent. Therefore, the main focus in this category is how to define the *closeness* properly and find the candidate efficiently.

In this category, **Distance-based** methods measure the closeness based on spatial proximity and/or topological similarity. Apart from some old methods that only consider point-to-point distance [103], recent algorithms evaluate the closeness by distance functions that consider both spatial proximity and topological continuity, which have much better accuracy but also higher complexity, like Fréchet distance [135] and Longest Common Subsequence (LCSS) [162]. **Pattern-based** methods define the closeness as the most similar travel patterns in historical map-matched data. Following the assumption that people tend to choose the same path given a pair of origin and destination points, pattern-based methods store historical map-matched data as references and achieve map-matching of new trajectories by finding similar patterns in the trajectory history.

In summary, despite some early geometric/topological methods that run fast but with poor performance, most of the recent similarity-based methods require high computation cost, but they are sensitive to GPS errors due to their reliance on trajectory shape, which is also the main reason for not supporting online map-matching since the shape is is incomplete.

Frec2013 [135] proposes a Fréchet distance-based similarity method with attention on the alternative path problem. The algorithm initially finds paths whose Fréchet distance to the trajectory is lower than a threshold. If the result is a unique path, this path will be returned as the final match; otherwise, the algorithm ranks candidate paths using a weight function and returns the path that ranks first. However, due to the feature of Fréchet distance, the path is sensitive to trajectory outliers since the distance is determined by the most divergent point.

Lcss2017 [162] uses Longest Common Subsequence (LCSS) for similarity calculations. Specifically, the algorithm segments the trajectory and finds the shortest path on the map for each pair of segment endpoints. The obtained shortest paths are then concatenated to form a candidate route; meanwhile, their corresponding LCSS scores are summed to obtain the total LCSS score. Since increasing the segmentation granularity enhances the accuracy but lowers the efficiency of map-matching, considering the trade-off between these two factors, the algorithm returns the path requiring the lowest level of segmentation but gaining a total LCSS score higher than the predefined threshold.

Hist2012 [159] first utilises the historical data for pattern-based map-matching. The historical map-matched trajectories are processed and stored as OD pairs and corresponding routes. The map-matching of a new trajectory is achieved by finding a historical trajectory, or the concatenation of multiple historical trajectories if each point of the trajectory falls into the safe region (with temporal constraints) around the task trajectory. The algorithm finally uses a scoring function to decide the path the vehicle actually travelled. However, due to the sparsity and disparity of historical data, the query trajectory may not be fully covered by historical trajectories especially in rural regions, which leads to a direct matching process.

4.3.2 State-Transition Model

The state-transition models build a weighted topological graph which contains all possible routes the vehicle might travel. Different from the road network, the vertices in this graph represents the possible states, which is the potential locations of the vehicle during its trip, while the edges represent the transitions between states, which only connect the states with continuous timestamps. In most cases, both the states and transitions are weighted according to their possibility, and the best matching results is achieved by finding the optimal path in the graph from the first timestamp to the last.

Currently, the graph can be built in three different ways: (1) The **Hidden Markov Model (HMM)** is the most popular model in map-matching. In the HMM model, the observations are visible while their actual states are hidden. Such a model can be easily applied to the map-matching problem where the observations correspond to the trajectory samples and the vehicle's actual locations are the hidden states. Therefore, the model can find the best possible state sequence, which is the matching result, through dynamic programming [137]. (2) Like the HMM, the Conditional Random Field (CRF) is another statistical model used for finding the best possible solution. The core difference between the CRF and HMM is that the probability of a state in HMM only depends on itself, other states at the same stage and one stage ahead, whereas CRF model interacts amongst all observations. (3) The weighted graph technique (WGT) builds a weighted graph whose nodes are candidate road nodes/edges of trajectory samples, while edges represent the transition between candidates of two consecutive samples, whose weights are assigned according to the shortest distance, turn frequency [156] and other factors that affect travel preference. The final matching result is mostly achieved by returning the path with the highest cumulative edge weights [60, 63, 64, 84].

The core advantages of the state-transition model are its ability to model the user's travel behaviours by introducing various features. Therefore, most papers in this category focus on designing different weighting strategy, in particular, defining different emission/transition probability functions in HMM/CRF or assigning different vertex/edge weights in WGT. The ST-matching, which is the first work in WGT [84], weights an edge ($c_i \to c_{i+1}$) simply based on a spatial cost (distance reasonability) and a temporal cost (velocity reasonability). Yuan et al. [158] further considers mutual influences between neighbouring nodes. The scoring function in He et al. [60] stresses on road connectivity, travel time reasonability and the taxi status. Specifically, the scoring function here is a Fuzzy logic model. Rahmani et al. [105] incorporate road features (traffic lights and left turns) in scoring functions, as well as the reasonability of estimated travel time. The IF-matching algorithm [63] propose a novel feature to be considered when weighing a path segment, which is the speed rationality with respect to surrounding road segments. At the same time, it employs the spatial and temporal constraints in previous work [84]. Meanwhile, Huang et al. [64] weights a path segment in proportion to the frequency that the path was visited in historical trajectories. Besides, since the state-transition graph is much smaller than the road network, most of the algorithms under this model run much faster compared to the similarity-based methods. Therefore, many algorithms in this category also have an online version [50, 65, 129, 156]. However, since the construction of a state-transition graph requires multiple stages, the matching of current point is usually achieved after receiving several new samples, named as sliding window strategy, which leads to a matching latency. Usually, the larger the window size, the higher the map-matching accuracy can achieve but with a longer delay. Therefore, online state-transition algorithms face a trade-off between latency and accuracy.

Hmm2009 (OFF-HMM) [90] is the first work that provides a detailed explanation of how HMM model can be adapted in map-matching problems. The work gives a precise definition of emission probability, which assumes that for correct matches, the direct distance $dist(o_t, s_i)$ between the sample and its candidate edge follows a Gaussian distribution, i.e.:

$$P(o_t|s_i) = \frac{1}{\sqrt{2\pi}\sigma} e^{-0.5(\frac{dist}{\omega})^2} \tag{4.1}$$

This definition is widely adopted by later works [8, 47, 67, 107, 129], and a transition probability, which considers the shortest path between candidate road edges and its difference to the direct distance between the corresponding trajectory points. It is by far the most popular solution in the map-matching field. Later, **Vary2012 (ON-HMM-VSW)** [50] further extends this work to online scenarios. Other than simply introducing the sliding window with a fixed size **(ON-HMM-FSW)**, the paper mitigates the latency problem by varying the window size according to the uncertainty of matching. More specifically, the algorithm expands the window forwards when a new location observation is generated, while it contracts the window from behind when the candidate paths converge at a point in any Markov stage in the window. The path owning the highest probability up to the convergence point is returned and shared by all future candidate paths, which is called convergence-determined pathfinding. However, this algorithm lacks a strategy to optimise the latency and accuracy trade-off, i.e. determining at which stage to pay an accuracy cost and stop delaying. This problem is targeted by **Eddy2014 (ON-HMM-DSW)** [129], which models the latency and accuracy costs separately and tries to minimise their sum. The authors have proven that their algorithm is both error-bounded and latency-bounded. Even though, the latency problem still exists.

Crf2014 [65] proposed a CRF-based algorithm that can be applied to both online and offline situations with high accuracy. Its overall approach is similar to HMM-based algorithms as follows: Firstly, it identifies candidate points on the road network for each location measurement from the input, where the candidate points for each measurement are the most likely positions on road edges inside the radius (700m) of this measurement. Then, paths that connect discrete candidate points are computed using the standard shortest-path algorithm which considers the maximum speed limit. Lastly, the most likely admissible path is identified using a CRF of which the transition model utilises the driving patterns of drivers.

Feat2018 (OFF-WGT, ON-WGT) [156] achieves the map-matching by building an action graph among candidates. After retrieving candidate roads for each trajectory sample, the algorithm builds the action graph by connecting the candidates from neighbouring samples with their shortest path on the map. However, instead of assigning weights to the edges by the length only, the algorithm combines the road length, turning cost and other features (optional) as action costs. The final matching result is achieved by finding the path with the lowest weight connecting the candidates. To further accelerate the process, the algorithm applies a trajectory segmentation strategy, which basically compresses the trajectory using the Douglas-Peucker algorithm.

4.3.3 Candidate-Migration Model

In the candidate-migration model, the initial candidates come from the nearby vertices/edges of the first trajectory sample. Then, different from the state-transition model, which finds candidates for subsequent samples, the initial candidates keep migrating to their new locations near incoming samples according to some propagation rules, and candidates that are no longer relevant are pruned. Regarding each candidate as a vote, through the maintenance of the candidate set, the algorithms are able to find a segment with the most votes, thereby, determining the matching path. There are two types of models in this category: (1) **Particle Filter (PF)** is a state estimation technique that combines Monte Carlo sampling methods with Bayesian Inference. It recursively estimates the Probability Density Function (PDF) of the road network around trajectory samples as time advances. The function is approximated by particles, each of which maintains a value indicating how close it is to the existing samples. The values are updated each time a new sample is received, and the PDF is calculated accordingly; then, a new set of particles is resampled. During the process, particles with a higher value are more likely to propagate, while the rest are likely to be filtered out. (2) Similar to the PF, the **Multiple Hypothesis Technique (MHT)** also maintains a list of initial candidates (hypotheses). As a new sample arrives, the MHT updates the hypotheses by traversing the network from previous locations to reach the region near the new sample, and it manages to reduce computation during the process. An MHT evaluates each candidate hypothesis based on a scoring function instead of approximating the PDF for the neighbouring map area, which significantly reduces the computation cost.

Compared to the state-transition model, the candidate-migration model is more robust to the off-track matching issue since the current matching is influenced not only by a previously defined solution but also by other candidates. Therefore, it can better handle occasional outliers where the state-transition methods may break due to missing candidates. However, to ensure this, the initial particle/hypothesis list should be sufficiently large to ensure correct result coverage. Moreover, the initial idea of the candidate-migration model fits the online scenario perfectly. However, the problem of this model is its high computation cost, especially for the PF. In the PF, the approximation of true the PDF is extremely slow as the number of particles tends towards infinite, so it has to limit the number of particles to the magnitude of hundreds [23] in practice, which in turn affects the accuracy.

Pred2018 [119] propose an MHT-based algorithm that achieves high accuracy with no latency by incorporating a route prediction model. This model basically describes the probability of being at a road segment r_{k+1} at timestamp $k+1$ providing r_k, denoted as $p(r_{k+1}|r_k)$. Specially, $p(r_{k+1}|r_k)$ takes into account probabilities of all paths starting from r_k. Here, a path probability is defined as the sum of its edge transition probabilities estimated from historical trajectories. This algorithm also defines an observation model the same as HMM-based algorithms, describing the probability of observing a location g_k providing r_k ($p(g_k|r_k)$). After setting up the route transition model and the observation model, a Bayesian scoring function can be defined. It models the posterior probability of r_k, providing all observations up to timestamp k ($p(r_k|g_{1:k})$) as follows.

$$p(r_k|g_{1:k}) \propto p(g_k|r_k) \sum_{r_{k-1}} p(r_k|r_{k-1}) p(r_{k-1}|g_{1:k-1}) \tag{4.2}$$

In terms of the matching model, the initial hypothesis set is obtained by selecting those road points whose distances to the first observed location are less than a predefined threshold. After then, new hypotheses that represent reachable (candidate) road edges are added to the hypothesis set. For each candidate road edge, its prior probability is estimated from the route prediction model ($p(r_{k+1}|r_k)$), while its posterior probability is updated after receiving the latest measurement, and derived by summing the posterior probabilities of its hypotheses estimated by the Bayesian scoring function. In case of the explosion of hypotheses, old candidates are pruned if their probabilities are less than a predefined threshold. At each timestamp, the highest-probability hypothesis is returned.

4.3.4 Scoring Model

The idea of the scoring model is simple. Instead of building a graph or traversing the network, the scoring algorithms simply define a scoring function and calculate the score for each candidate vertex/edge. The candidate with the highest score is chosen as the matching result. The major difference inside the category is the definition of a scoring function. The features used in defining the matching score can be arbitrary, including those related to road conditions, vehicle status, driving behaviours and travel patterns. A group of algorithms [53,58,101,104,112,127] apply the **Naïve scoring** method. They assign a group of candidates to each trajectory segment/sample and find a road edge from each group that maximises the predefined scoring function.

Scor2015 (ON-SCO) [101] propose a scoring algorithm that calculates the total weighted score of each candidate road segment. The scoring function considers four features, i.e.the perpendicular distance between the trajectory point and its projection on the candidate segment, the shortest distance between the projection point and previous matched point, the direction difference between the vehicle and the candidate road segment, and the heading difference between the trajectory point and the previous trajectory point. The four features are modelled differently, and their scores are summed with different weights. The optimal weight of each feature is obtained by employing the Genetic Algorithm (GA).

Lane2019 [112] is the most recent work in this category, which achieves a lane-level map-matching performance. The algorithm first identifies lanes in each road by utilising the road width information in the map and partitioning them into grids accordingly. The algorithm then finds candidate lane grids around the observed location and scores these grids at each timestamp. The grid results in the maximum score are then returned. The scoring function is a linear combination of four features, i.e. the proximity between the grid and trajectory sample, the estimated location of the vehicle at the next time stage, the reachability from the grid and the intention of a turn. These features are modelled individually; their scores can be obtained from the corresponding models in every timestamp. In addition, feature scores are weighted differently in the scoring function whose coefficients are computed by a training process before map-matching starts.

4.4 Tuning Techniques

To the best of our knowledge, there barely any new map-matching models have been proposed recently. Instead, they mainly focus on proposing new tuning techniques to the existing solutions, such as reducing the latency in online map-matching [50, 119, 129], tuning the matching model by using machine learning techniques [50, 65] or utilising more data features [63, 112] and applying data pre-processing or pre-computation for better performance. Different from the matching models, these optimisations are usually independent and can be integrated into different models for enhancement.

4.4.1 Parameter Tuning

Many algorithms use parameters to derive the optimal matching result. The common parameters include the radius of a candidate search around each trajectory sample; the threshold divides on-track matching and off-track matching; the weight of a feature value in the scoring function; and the uncertainty of measurement if it is not available from datasets. These parameters were set to constant values empirically or optimised through experiments in most previous works. However, plenty of recent works adopt machine learning strategies to tune the parameters, including Inverse Reinforcement Learning [95], Support Vector Machine [50], Multivariate Adaptive Regression Splines [112], Genetic Algorithm [101] and Fuzzy Logic [20, 21]. Despite the need of training datasets, the core advantage of machine-learning-based parameter tuning is its adaptability to find optimal settings under different data features. However, there is still no matching model that is built based on learning strategy, which can be a potential research direction.

4.4.2 Data Preprocessing

The trajectory preprocessing is usually performed for two purposes, i.e. enriching data features by fusing data from different sources [12, 49, 71, 80, 123, 155] and correcting data errors [98, 128]. The idea of data fusion is to calibrate one type of data source with the help of another data type as they both provide estimations of the same object simultaneously, while the data correction estimates the state of the object according to its trajectory data and adds features (speed, heading) or corrects outliers. Existing works achieve these two goals by introducing techniques, like the Extended Kalman Filter (EKF), the Particle Filter (PF) and trajectory compression. Besides, Hashemi et al. [57] propose an artificial neural network to correct GPS points, and simple trajectory compression is also used [156] to accelerate the matching process.

4.5 Evaluation Metrics

In general, the map-matching accuracy is evaluated by measuring the similarity between the map-matching result and corresponding ground-truth. Currently, various evaluation metrics have been proposed for map-matching, including the correct road identification % [92, 99, 103, 117, 139, 153], map-matching precision/recall [8], route mismatch fraction [47, 90, 115] and matching point/route accuracy [7, 152]. There is still no consensus on which metrics can better evaluate the performance [72, 74], and none of the existing work evaluates their differences. Since a map-matching result is usually represented by either a set of matching points $\mathcal{MP}(Tr)$ or a matching route $\mathcal{MR}(Tr)$ in online and offline scenarios, respectively, we first introduce the existing metrics accordingly.

4.5.1 Point-based Metrics

Point-based metrics mainly compare the matching points $\mathcal{MP}(Tr)$ with the ground-truth point matching results $\mathcal{MP}^*(Tr)$. Normally, they compare two point sets based on their pairwise point distance. Assuming mp_i is a point in $\mathcal{MP}(Tr)$ and mp_i^* is the corresponding ground-truth, Singh et al. [114] measure the correctness by calculating their Root Mean Square Error (RMSE):

$$RMSE(\mathcal{MP}(Tr)) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(dist(mp_i, mp_i^*))^2} \tag{4.3}$$

Here, the distance function $dist(a, b)$ refers to either the Euclidean distance in the $< x, y >$ coordinate system or Great Circle distance in $< longitude, latitude >$ system. In general, a smaller *RMSE* value usually means a more accurate map-matching result. Another common measure is to compare the two matching point sets directly. Note that a point is matched correctly only when $p_i = p_i^*$; since a matching point is usually the projection point to a matching road, a correct point match usually implies a correct road match. Therefore, it also works for the cases where only a matching edge is provided as ground-truth instead of the matching point [7, 120, 137]. Overall, since each trajectory point must have one and only one matching point, both the size of $\mathcal{MP}(Tr)$ and $\mathcal{MP}^*(Tr)$ are equivalent to the count of trajectory points. Therefore, we define the accuracy of the point match by the percentage of correct matches, i.e.

$$P - ACC(\mathcal{MP}(Tr)) = \frac{\mathcal{MP}(Tr) \cap \mathcal{MP}^*(Tr)}{|Tr|} \tag{4.4}$$

In general, the point matching result is regarded as the standard output of online algorithms due to the discontinuous matching routes, so the point-based metrics are mostly used in online mode.

Unfortunately, the point matching evaluation is not very popular mainly caused by the lack of ground-truth results. Normally, the ground-truth results are obtained from the drivers manually labelling their trip. However, although the drivers clearly remember their travel route, it is very hard for them to mark their actual location at a particular moment. Therefore, most of the point matching ground-truth comes from identifying locations through in-car camera images or simply matching the trajectory point to its closest point on the ground-truth route, which is either inefficient or pointless.

4.5.2 Route-based Metrics

Route-based metrics measure the correctness of the matching routes. A simple route-based metric, which is adopted in various papers [112, 136], regards the matching route $\mathcal{MR}(Tr)$ as a set of road edges $\mathcal{ME}$ (same applies to the ground-truth $\mathcal{MR}^*(Tr)$) and evaluate the set precision/recall/F-measure (similar to the definition above). Alternatively, we can generate an overall score to indicate the accuracy [152], which is similar to the format of the Jaccard Similarity:

$$R - ACC(\mathcal{MR}(Tr)) = \frac{|\mathcal{MR}(Tr) \cap \mathcal{MR}^*(Tr)|}{|\mathcal{MR}(Tr) \cup \mathcal{MR}^*(Tr)|} \tag{4.5}$$

Another metric, named as the Route Mismatch Fraction (RMF) [90], computes the total length of the false positive and false negative matching route edges, denoted as d_+ and d_-, respectively. Assuming d_0 is the total length of the ground-truth matching route, RMF quantifies the map matching error by the fraction of $(d_+ + d_-)/d_0$. Therefore, the matching result is more similar to the ground-truth if RMF is smaller.

In fact, most of the early works used the metric "Correct Road Identification$\%$" to evaluate the percentage of roads in ground-truth that are correctly identified. It is also a route-based metric; however, this metric is never formally defined by any of them, which makes it inappropriate to simply compare their performance based on the claimed accuracies, not to mention the variety of datasets on which the experiments are conducted. Besides, it is worth noting that the current route-based metrics ignore the edge order and edge revisit as it treats a route as a set of edges, which can potentially be problematic as it fails to detect road revisits and frequent recurrent travels.

4.5.3 Metrics without Ground-Truth

As mentioned by most of the map-matching papers, obtaining a dataset with reliable ground-truth results is challenging and sometimes infeasible. Hence, as a compromise, many map-matching algorithms estimate their map-matching accuracy by solely comparing their matching result with the original trajectory.

The first intuition is that the incorrect matching results usually leads to lots of detours, so the length of a good matching route should be close to the length of the original trajectory [106, 110], i.e:

$$I_L(\mathcal{MR}(Tr)) = \frac{\sum_{i=1}^{m} e_m.l}{\sum_{i=1}^{n-1} \overline{p_i p_{i+1}}.l} \tag{4.6}$$

where $m = |\mathcal{MR}(Tr)|$ and $n = |Tr|$. Therefore, a match is more likely to be correct if its I_L score is closer to 1. Another solution considers the average pairwise distance between the matched point and the original trajectory. However, such a metric does not consider the continuity of the matching result and only focusses on the point closeness, which can be useful in the online scenario but is not in accordance with the trend of offline map-matching [85, 106].

Apart from the above quality measures, efficiency is mainly evaluated by running time. In addition, the SIGSPATIAL GIS cup 2012 [7] proposed a point-based metric that considers both the efficiency and the correctness. It prioritises the algorithm efficiency and punishes the incorrect matching results to avoid random guess; it also takes into account the map-matching confidence, which is a user-defined score for each point match representing how confident the user believes the matching is correct. Since the match confidence is not provided in our results, we will not include it in our experiments. In our evaluations, we will first test the aforementioned measures to see their consistency of evaluating map-matching results under different circumstances; then, we choose one metric for both online and offline map-matching evaluation for the rest of the experiments.

4.6 Experiments

In this section, we conduct a series of experiments on the candidate algorithms and evaluation metrics for the following topics:

- We enumerate the existing metrics used for matching result evaluation. Moreover, we compare their measurement results under different circumstances to evaluate their abilities in identifying matching errors.

- We compare various algorithms in both online and offline modes to reveal the strengths and weaknesses of each type of matching methodology.

- We conduct experiments on multiple real and synthetic datasets to test the robustness of candidate solutions to various data quality issues.

4.6.1 Dataset Settings

A complete dataset for map-matching evaluation should contain three components: the ground-truth map, the input trajectories and the corresponding ground-truth map-matching results. In fact, obtaining the ground-truth map-matching result, which is the actual route the vehicle passes when being sampled, is very challenging. Since there is no way to automatically track routes in practice, as a compromise, the ground-truth results are usually obtained by human labelling, which is done by either asking the motorists to confirm their travel histories [90] or manually improving the automatic map-matching results [7, 72, 135, 136]. Either way, it is not possible to guarantee 100% correctness, and the dataset are usually very small due to labour cost. Therefore, a synthetic dataset is also viable in some evaluations. Additionally, to evaluate the performance comprehensively, the selected datasets should be diverse on the following data features:

- **Trajectory sampling rate:** The trajectory datasets should be sampled in various frequencies, including extremely high ($< 10s$), high ($10\ 30s$), low ($> 30s$) and dynamic (high deviation) sampling rates, and the data can be used in both online and offline models. Note that the low-sampling-rate datasets can be obtained by subsampling the high-sampling-rate datasets.

- **GPS accuracy**: GPS accuracy is a crucial factor affecting the map-matching quality. Although unknown, GPS accuracy can be estimated by the average distance between the GPS locations and their corresponding ground-truth matching points.

- **Map density**: The map density was not considered as a map feature by previous map-matching experiments as most of the previous works only conduct experiments on a single dataset. However, from what we observed, the density of roads actually affects the correctness and efficiency of the map-matching results. To this end, we propose the *map density* attribute, which is defined by the average length of the roads per square kilometre of a map area (km/km^2).

- **Map size**: The map scale significantly affects the efficiency of many map-matching algorithms. As the complexity of most graph-based algorithms, like shortest-path and nearest neighbour search, scales with the map size, our datasets should also cover different map scales for scalability evaluation.

In summary, we use two datasets with several variations in our experiments. The *Global* [72] dataset is a public dataset proposed for map-matching evaluation. It contains 100 GPS trajectories sampled from 100 different areas all over the world, each of which is provided with a dedicated underlying map. Therefore, the *Global* dataset is not suitable for efficiency evaluation as it is impossible to be paralleled. However, it helps the map-matching algorithms to better identify weaknesses since each trajectory is labelled with a selection of features that may pose difficulties to map-matching algorithms, namely u-turns, hives, loops, gaps and severe congruence issues. In contrast, the *Beijing* dataset is a commercial dataset that contains abundant trajectories generated by taxis in Beijing. The trajectory sampling rate of *Beijing* dataset is relatively lower and heterogeneous than *Global*. Also, the map size is huge and the density is diverse amongst the urban and rural regions, so we extract four sub-areas with different sizes and map densities from the original map, namely *Beijing-U*, *Beijing-R* *Beijing-M* and *Beijing-L*. The *Beijing-U* and *Beijing-R* represent two maps extracted from urban and rural areas, respectively. They have roughly the same size but different map densities ($27.3 vs 13.9$) and trajectory profiles for testing the influence of map density on map-matching results. Meanwhile, the *Beijing-M* and *Beijing-L* are much larger maps than the former two, which are used for scalability evaluation.

The specifications of the datasets are listed in Table 4.2. Here, the GPS accuracy is obtained by calculating the distance between the trajectory point and its ground-truth point match result. It is interesting to see that the measurement error in the *Beijing* dataset is not very high in general, especially for the rural region. However, there are a lot of outliers that may deviate from its actual location by a huge amount, which is unable to be modelled by existing map-matching algorithms. Unfortunately, we are unable to gather enough statistics for the *Global* dataset since it does not provide ground-truth point match results and their maps are too diverse.

TABLE 4.2: Summary of experiment datasets

Name	Input Trajectory				Road Network	
	Trajectory Count	*Trajectory Point Count*	*Sampling rate(sec)*	*Avg GPS Accuracy*	*Map Size(km^2)*	*Map Density (km/km^2)*
Global	100		1		N/A	N/A
Beijing-U	7,905	247,544	11.0	9.1	9.9	27.3
Beijing-R	3,106	119,612	8.6	5.8	9.9	13.9
Beijing-M	73,072	3,285,934	10.3	7.8	57.0	24.2
Beijing-L	951,745	93,951,403	6.7	7.6	33404.6	2.6

In addition, we generate a synthetic trajectory dataset based on the *Beijing-M* map. The basic idea of the synthetic trajectory generator is to first obtain a valid route from the map, which represents a trip of a vehicle. The route is then interpolated by a set of points representing the accurate GPS samples. We then introduce GPS measurement errors (standard deviation σ) that follow the 2D Gaussian distribution and sampling errors (down-sample the trajectory to a certain sampling rate of Δt). In this dataset, we simulate different GPS accuracies by varying the measurement error level. Besides, we also introduce outliers (points that deviate from the actual location by an excessive amount) to the synthetic trajectories. Since many of the map-matching solutions assume the sampled GPS point is always near its actual position within a certain distance, an outlier whose distance exceeds this threshold may lead to a result discontinuity [90] or unreasonable matching sequence. Hence, we also set different outlier rates to test the robustness of the matching algorithms.

Our experiments are performed on a single server, which consists of two Intel(R) Xeon(R) CPU E5-2630 with 10 cores/20 threads at 2.2GHz each, 378GB memory and Ubuntu 16.04. The multicore feature and large memory enable the concurrent map-matching process, which is supported by all candidate solutions.

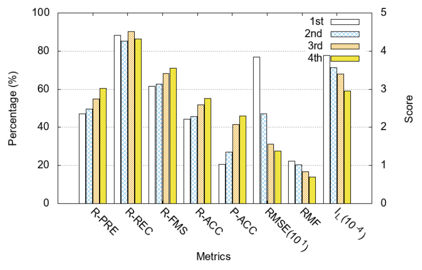
4.6.2 Experimental Design and Results

Our experiments mainly consist of three aspects: (1) the comparison of map-matching quality metrics, (2) the performance evaluation of candidate algorithms and (3) the influence of data quality issues on the map-matching performance. To fully demonstrate the performance of every algorithm, we did a lot of experiments beforehand to find the ideal value for the most influential parameters and set the rest as default.

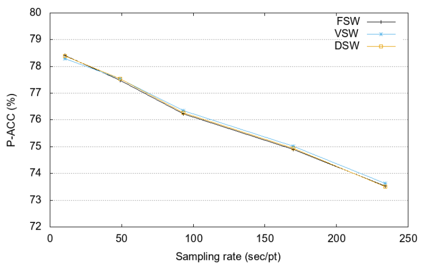
Measure comparison

We first compare the quality measures mentioned above to see their sensitivity to the change of matching result quality. The candidate metrics include *RMSE* and point matching accuracy (P-ACC) in point-based, route matching precision/recall/F-measure (*R-PRE/R-REC/R-FMS*), *R-ACC* and *RMF* in route-based and I_L in non-ground-truth metrics. We conduct a naïve weighting method (ON-SCO) method on online mode with various parameter settings to generate results with different matching quality and to see how those metrics perform under different circumstances, as depicted in Fig. 4.1a.

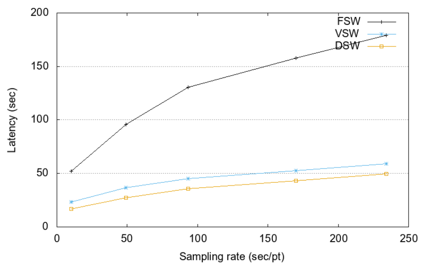
In Fig. 4.1a, the overall map-matching quality slightly increases from the 1st to the 4th attempts. As both route F-measure (R-FMS) and R-ACC are built on top of the route precision (R-PRE) and recall (R-REC), they have roughly the same trend as expected, so they are usually interchangeable when evaluating the overall accuracy. Meanwhile, precision and recall are more useful when focusing on certain aspects. Likewise, point accuracy (P-ACC) shows the same trend as route accuracy measures, however, it is more sensitive to low map-matching quality where the P-ACC drops from 41.6% to 20.6% (3rd to 1st) whereas R-ACC only reduces from 51.9% to 44.3%. On the other hand, since the root mean square error (RMSE), route mismatch fraction (RMF) and I_L are value-based measures, we normalise their value to $[0, 5]$ to better fit in the figure. Like the other point measure (P-ACC), RMSE also deteriorates quickly as the matching quality decreases. The RMF shows a good trend as it decreases steadily as the matching quality slowly increases. As a measure that requires no ground-truth, the I_L performs surprisingly good as it successfully captures the change of matching quality. However, due to the lack of reference value, both RMF and I_L cannot indicate how good a matching result is unless compared with the results from other confirmed algorithms. Nonetheless, it is still a viable measure when no ground-truth is available. Overall, the R-FMS and P-ACC are sufficient for route-based and point-based evaluation, respectively, which will be used in the following experiments.



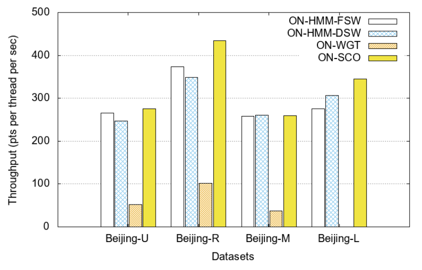
(a) Metric comparison



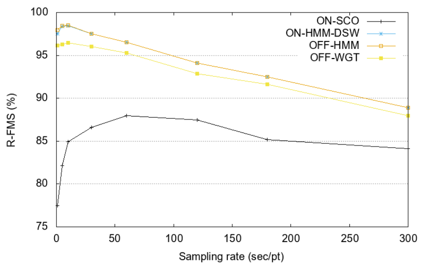
(b) Buffer strategy under different Δt (accuracy)



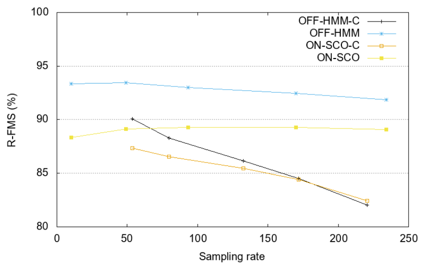
(c) Buffer strategy under different Δt (latency)



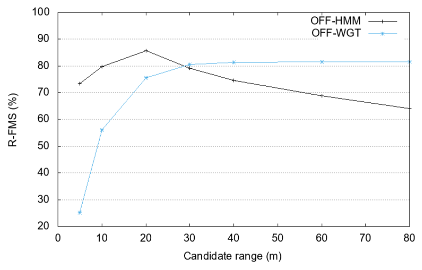
(d) Throughput of online algorithms



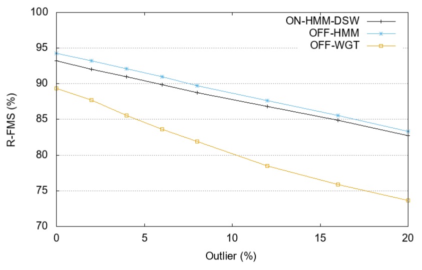
(e) Accuracy comparison under different Δt (Global)



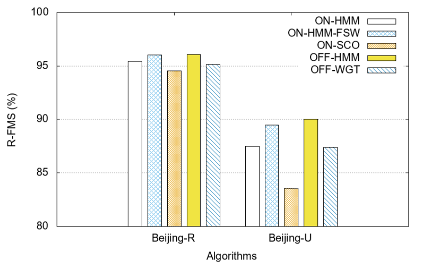
(f) Effectiveness of compression



(g) Importance of measurement error modelling (sigma=10m)



(h) Robustness towards outliers error



(i) Influence of map density

FIGURE 4.1: Summary of experimental results

Algorithm comparison

We evaluate the performance of both online and offline algorithms. In online mode, apart from the performance comparison between candidate algorithms, including *ON-HMM* [50], *ON-WGT* [156] and *ON-SCO* [101], we also evaluate how different sliding window strategies affect the accuracy and latency of the online map-matching algorithms. In offline mode, we mainly focus on the matching quality under different trajectory sampling rate.

Sliding window strategy: To ensure a fair comparison, we choose the same HMM map-matching model [90] and apply different sliding window strategies. In general, there are three buffer types: (1) the Fixed Sliding Window (*FSW*) ensures n (here, we define $n = 10$) succeeding points are received (unless the trajectory ends) before matching the current point. (2) the Varying Sliding Window [50] (*VSW*) extends the window until all the map-matching possibilities within the window converge; and (3) the Dynamic Sliding Window [129] (*DSW*) relies on a cost function, which is designed according to the ski-rental problem, to find a balance between accurate results and latency cost. Fig. 4.1b and Fig. 4.1c shows the point-match accuracy and average latency of different sliding window strategies on the Beijing-M dataset. In these figures, we down-sample our input trajectories to demonstrate the effectiveness of those strategies under different sampling rates. The figures show that the matching accuracy of all three strategies decreases at roughly the same rate as the sampling rate reduces. However, in terms of the latency, the adaptive sliding window strategies (*VSW* and *DSW*) have much shorter latency compared to *FSW* and increase slowly as the sampling rate decreases. Overall, the cost-based sliding window has the best performance.

Online performance: As an online algorithm, the running time should be a crucial factor when considering the overall performance. According to Fig. 4.1d, the ON-SCO has the most throughput amongst all other candidates. As the scoring method processes each sample independently without considering the globally optimal sequence, the matching process is simple and instant. In contrast, the ON-WGT performs poorly in online mode. To remedy the time-consuming action graph construction step, the original algorithm introduces trajectory compression and limits the size of candidates for better efficiency. However, in online mode, the compression is less effective, making the algorithm run even slower. Therefore, the ON-WGT is not considered as a competitive algorithm. From the data perspective, the map size does not affect the performance significantly; instead, the map density is more influential since algorithms run much faster on *Beijing-R* than on *Beijing-U* and *Beijing-M*; even with a larger map, it has the similar performance as *Beijing-U* due to similar map density.

Matching accuracy: We evaluate the accuracy on the *Global* dataset as it has the highest trajectory sampling rate (1 sec/point). We down-sample the trajectories to various degrees and compare both online and offline algorithms. In Fig. 4.1e, despite its high throughput, the ON-SCO has the worst accuracy amongst all other solutions. On the contrary, with the help from the dynamic sliding window, the online HMM solution (ON-HMM-DSW) achieves roughly the same performance as its offline counterpart, which indicates the importance of latency. Another interesting finding is that all of the methods receive a performance penalty when the sampling rate is extremely high. The main reason is that when neighbouring samples are too close, their relative distance can be dominated by the trajectory measurement errors, which make their actual locations unpredictable. Therefore, a down-sampling process is recommended when dealing with extremely high sampling-rate data.

Influence of data quality

Trajectory compression: Talking about downsampling, we also compare the map-matching performance on the dataset with periodical down-sampling and trajectory compression. To our surprise, as shown in Fig. 4.1f, the map-matching on compressed trajectories (Douglas-Peucker algorithm) has worse performance than regular down-sampling when they reach a similar sampling rate. The reason is that, in the Douglas-Peucker algorithm, the points that are more distant are more likely to be retained. Since the outliers usually deviate from the trajectory significantly, they are much easier to be preserved, which causes incorrect matching results. Therefore, the compression method used in data preprocessing should be chosen wisely.

Impact of data features: We generate two synthetic datasets to test the influence of data quality. In Fig. 4.1g, the generated dataset has a high sampling rate ($\Delta t = 5$), and the measurement error follows the Gaussian distribution whose $sigma = 10m$, which means most of the trajectory samples ($> 95\%$) are within 20 meters to the actual location of the object. We choose two state-transition algorithms and set their candidate search range to $[5, 80]$, which indicates how poor we believe the input data quality can be. The results show that matching quality almost reaches its best when the search range is 20m, which means most of the correct locations are selected as candidates. However, we can clearly see a downtrend when the candidate range is too large in offline HMM. It is because due to the low expectation of data quality, the emission probability of the states plays a lesser role compared to the transition probability. Moreover, some of the wrong candidates may have much better transition probability due to their closeness, which makes the correct match much easier to be dominated. In Fig. 4.1h, we generate outliers by shifting a certain percentage of samples to remote locations. It is clear that all of the candidate solutions suffers significantly from the outliers since they fail to match the outliers correctly and the existing matching sequence breaks due to a missing transition. Finally, we evaluate how the map density affects the matching quality. As shown in Fig. 4.1i, even with the same map size, the map with fewer roads is much easier for map-matching, as such a factor affects online solution more significantly since it is easier to match to a nearby wrong road if its neighbouring samples are unknown ($ON-HMM < ON-HMM-FSW < OFF-HMM$). On the other hand, the online scoring method (ON-SCO) suffers the most from the map density. Since it solely relies on scoring function, a denser map usually leads to more candidates with similar scores due to their spatial closeness. Therefore, matching on a dense map usually requires the algorithm to be much safer.

4.6.3 Experiment Findings

Overall, our experiments demonstrate several interesting findings, which are not mentioned by previous works:

- The comparison on existing evaluation metrics shows minor differences between them. Current measures can correctly reflect the map-matching quality regardless of which one you choose. More importantly, you can evaluate the matching quality using I_L in decent accuracy even if you don't have ground-truth dataset, which is quite helpful considering the lack of ground-truth datasets in this research field.

- The map-matching quality does not affected by the map size in general. However, the map density is a crucial factor to both the matching accuracy and efficiency. The current public datasets for map-matching tasks mainly run on sparse maps, which makes previous experiment results all achieve extremely high accuracy. More research is needed for map-matching on dense map.

- The increase of a trajectory's sampling rate does not always lead to better map-matching result. An extremely high sampling rate ($< 10sec$ per point, depending on other factors, like GPS error, map density, etc.) will cause a decrease on the map-matching quality due to the amplification of GPS error. A solution to it is to down-sampling the trajectory. However, down-sampling it using trajectory compression is risky as the outliers may not be smoothed. Therefore, more work is expected on trajectory preprocessing.

4.7 Remaining Challenges

According to our observations through experiments and literature review, we still find a few challenges remaining in the current map-matching algorithms. We demonstrate some of them through examples in this subsection.

4.7.1 Trajectory Systematic Error

It is well known that the trajectory measurement error affects the matching result significantly; however, another type of GPS error, namely systematic error, is rarely addressed by the existing method. In general, the systematic errors occur when the object travels through regions that weaken the GPS signal significantly and continuously, like tunnels and mountains. As shown in Fig. 4.2, the main feature of such a GPS trajectory (red) is that it deviates from its actual locations (mint) by a constant amount consecutively. Such a case is not considered by any of the existing algorithms, so the matching results (blue) show obvious breaks in multiple locations. These errors can potentially be mitigated if the sampling rate is set to be lower to skip those regions; however, the detection of such problems remains a challenging task.

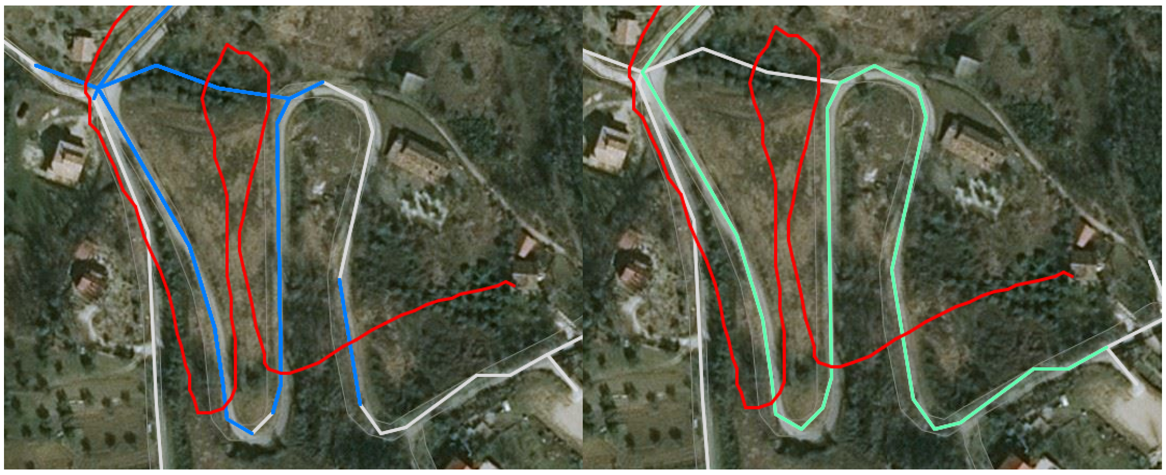


FIGURE 4.2: Map-matching error caused by systematic errors

4.7.2 Parallel Road

As shown in the experiments, the map-matching accuracy affected by the road network complexity significantly, which is the main reason why many algorithms whose authors claimed were near perfect in performance are near perfect have proven to be less effective in other experiments. Due to the ubiquity of interchanges, motorways and underpasses in modern cities, many roads are parallel to each other and situated closely, which makes it hard to decide on which road a trajectory is actually located. Therefore, matching on those roads is still very challenging. As an example in Fig. 4.3, although the trajectory is simple and clean, the matching results (ON-HMM at left and ON-SCO at right) in both figures contains errors, mainly because of the uncertainty between parallel roads. Furthermore, such cases affect the scoring-based methods more seriously as the parallel roads usually share similar map features (length, direction, speed limit, road width, etc.), so there is usually no clear winner. In fact, parallel roads also affect the acquisition of ground-truth data. As many of the existing ground-truth data are obtained by map-matching manually [72], it is also challenging for humans to identify the correct route, making the problem even more challenging.

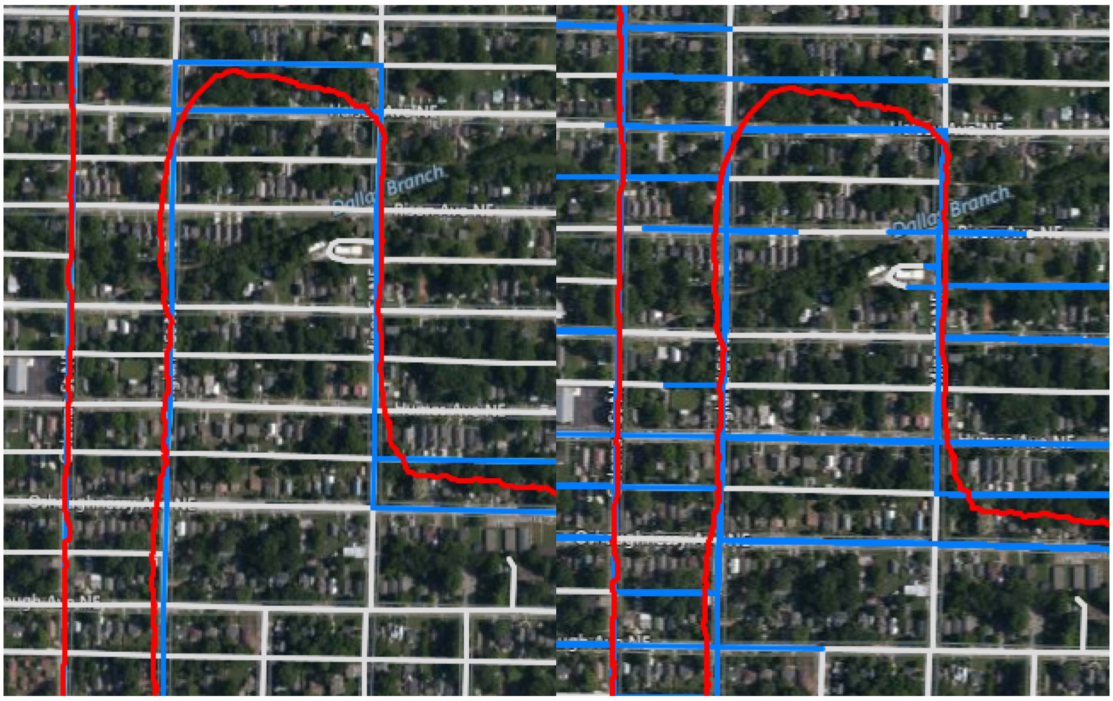

(a) (b)

FIGURE 4.3: Different matching results on parallel road (HMM and SCO)

4.8 Summary

In this chapter, we enumerate and categorise the existing map-matching algorithms according to their map-matching model, working scenarios and input data features. We discuss the main strength/weakness of each category and introduce their representative algorithms. Moreover, we analyse the current research trends in this field, including new techniques (machine learning), new data types (Bluetooth, DGPS, etc.) and new problems (lane-level map-matching), followed by the remaining challenges. Additionally, we conduct comprehensive experiments on multiple algorithms, metrics and datasets to compare the existing map-matching solutions. The experiment results show that: (1) the accuracy of online map-matching benefits a lot from delayed matching, and the latency can be wisely controlled without hurting performance; (2) it is not always true that a higher sampling rate leads to better matching performance. A downsampling trajectory is beneficial when the sampling rate is too high, but a simple trajectory compression strategy cannot serve this purpose; (3) HMM-based methods can still achieve better overall performance, while a simple scoring method can be very efficient in online scenarios without losing too much accuracy if the data quality is decent; (4) map density is a crucial factor affecting both the efficiency and matching accuracy of the algorithms, which is the main challenge in the future. Overall, this chapter summarise and compare the existing map-matching solutions and provide insightful observations and guidance for future research.

Chapter 5

A Co-optimisation Approach

5.1 Introduction

The digital map has been widely used in various kinds of location-based services, such as navigation, vehicle tracking and advertising. However, the current map quality is usually unsatisfactory. The reasons are two-fold: the frequent road changes and the low map update frequency. According to a report mentioned in [134], up to 15% of the road changes each year in some way around the world. On the contrary, the traditional way of updating maps relies on labour-intensive ground surveys, which leads to a long update period and expensive update cost. Since the inaccurate map may cause many severe consequences, like traffic accidents or even injuries [62], it has been one of the major factors affecting the quality of location-based services.

To avoid the map errors caused by the belated map update, people have started to update maps by using other resources. In recent years, the pervasive use of GPS-enabled devices produces massive users' vehicular trajectories. Since every vehicle must run on actual roads, these trajectories can be aligned to road networks via map-matching techniques [103] in order to support map-based applications. Meanwhile, they can also potentially be used to automatically construct or update digital maps [5, 134].

In fact, despite the different objectives, map-matching and map inference/update are correlated technically. In most map update methods, a map-matching process is conducted in the first place to obtain trajectories that cannot be matched to the current map, which have the potential to generate new roads [111, 133, 134, 143]. Some map inference methods also leverage the map-matching as a post-processing step to evaluate the quality of the newly-constructed roads [18]. On the other hand, the quality of map-matching results heavily relies on the quality of maps, which also benefits from good map inference/update outcomes.

However, the intrinsic inaccuracy of both maps and trajectories poses great challenges to all the above research problems: (1) digital maps contain missing roads and inaccurate layout, making trajectories matched to incorrect roads; (2) trajectories sampled by inaccurate GPS devices contain a significant amount of uncertainty, which affects the accuracy of map inference and map update; (3) existing map update methods do not examine their update results carefully, causing a large number of erroneous roads to be introduced to the map. Therefore, better approaches are required to address these challenges; (4) the time-consuming map-matching process in the existing map inference/update algorithms [18, 111, 133, 134, 143] is not wisely utilised as the generated matched trajectories are ignored which actually are valuable for estimating the quality of the newly-inferred roads.

In this chapter, we propose an iteration-based map-trajectory co-optimisation algorithm to improve the quality of both maps and trajectories. Overall, the main features and contributions of this work can be summarised as follows:

- We design a co-optimisation process, which aims to improve the quality of map update and map-matching simultaneously. To fulfil this purpose, we propose quality measures to quantify the map-matching quality and the map quality, respectively, and our goal is to maximise the quality score. To the best of our knowledge, this is the first work to propose quality measures for both map-matching and map update, and aim to improve both qualities through the update of the map.

- We propose two scores, namely confidence score and influence score, to measure the correctness of each newly road generated by map updates. For each new road, the confidence score evaluates our confidence in inferring it, while the influence score measures its contribution to the improvement of map-matching results. To this end, we propose a top-k HMM-based map-matching algorithm and the concept of matching certainty to better evaluate the road influence. Experimental results show that these scores can help better identify correct road updates over other outliers and meanwhile detect one-way roads.

- We design an iterative framework to combine map-matching, map update and our proposed co-optimisation model. Through the road filtering process and the control of quality measurements, the framework can guarantee a gradual improvement on both the map and trajectory-matching quality and overall superior performance after the iteration. Moreover, our framework is generalised to support existing map-matching/map inference/update methods and achieve better quality on top of them.

- We utilise a spatial index on trajectories to boost the process so that only trajectories that are relevant to the map update process participate in the rematch process. Our proposed method can significantly reduce the running time without losing too much of its effectiveness.

- Extensive evaluations on multiple-scale real datasets are conducted to verify the effectiveness of our proposals. Experiments also show the straight quality improvement upon the state-of-the-art map update methods with affordable overheads after being plugged into our framework. In addition to the quality improvement, our method can also run in a reasonable time compared with the existing map inference/update methods.

5.2 Framework Overview

In this section, we formally define the co-optimisation problem and briefly introduce our proposed framework.

5.2.1 Problem Definition

Since the map co-optimisation problem is related to both the trajectory map-matching and map inference/update problems, our problem also involves the trajectory data and road network data, which are defined in Section 1.2. In addition to the basic definition of trajectory, in our problem, we define the sampling interval Δt of a trajectory as the maximum time interval between any consecutive points in the trajectory. In this work, since we mainly focus on high-sampling-rate trajectories, we filter the input trajectories to satisfy the $\Delta t < 120s$ requirement.

As the two main components of our solution, we also introduce the definition of map-matching and map update:

Definition 5. *(Probability-Based Map-Matching) Given a road network G and a trajectory tr, the probability-based map-matching finds the most possible sequence of continuous road edges representing the actual route of tr. Eventually, the matching result of each trajectory point $M(p, G)$ is a road node v or an intermediate node on a road edge e, and the matching result of a trajectory segment $M(\overline{p_i p_{i+1}}, G)$ is represented as a sequence of road edges. Overall, the probability of the matching result is calculated as*

$$Pr(M(tr, G)) = \prod_{i=1}^{|tr|} Pr(M(p_i, G)) \prod_{i=1}^{|tr|-1} Pr(M(\overline{p_i p_{i+1}}, G))$$

Definition 6. *(Map Update) Given a set of trajectories $R = \{tr_1, tr_2, ..., tr_n\}$ and an initial map $G^0 = (V^0, E^0)$, we update the map by merging a new partial road network $\Delta G = (\Delta V, \Delta E)$ generated from R, i.e. $G = G^0 \cup \Delta G$.*

Note that, different from the map-matching definition introduced in Section 4.2, here we only define the probability-based map-matching, which directly refers to the state-transition map-matching model mentioned in Section 4.3. However, since the candidate-migration model also maintains the score of map-matching sequence and the result with the highest score is chosen, it can also be easily adopted in this definition. These two categories cover the majority of solutions in map-matching problem, including the most popular solutions [65, 84, 90, 137]. Meanwhile, we omit the definition of map inference since it is similar to map update without G^0.

Since the goal of our map-trajectory co-optimisation model is to improve the "quality" of both the trajectory matching result and the road network, we first give formal definitions of the quality measures for both datasets. In terms of the map-matching, for each trajectory tr, we measure its matching result on G by comparing it with the ground-truth matching result on the ground-truth map G_{gt}, i.e.: $\Psi(tr, G) = \frac{Pr(M(tr,G))}{Pr(M(tr,G_{gt}))}$. If we ignore the quality issue caused by the inaccurate map-matching algorithm, which is not the focus of our work, it is easy to deduce that $\Psi(tr, G) = 1$ when $G = G_{gt}$. However, since the road network is usually incomplete and contains erroneous roads non-existent in the real world, like the map inference/update results in our case, $\Psi(tr, G)$ sometimes can be larger than 1. For example, having an erroneous road that has the exact same shape as tr, the trajectory will be matched onto this road and its probability is definitely no less than the ground-truth. Hence, we introduce the concept of road correctness and define the map-matching quality formally.

Definition 7. *(Trajectory Map-Matching Quality) Given a trajectory tr, the ground-truth map G_{gt} and the road network G which is comprised of a correct sub-map G_+ and an erroneous sub-map G_-, the map-matching quality $Q_T(tr, G)$ is defined as follows:*

$$Q_T(tr, G) = \frac{Pr^+(M(tr,G)) - Pr^-(M(tr,G))}{Pr(M(tr,G_{gt}))}$$

where $Pr^+(M(tr,G))$ represents the summation of probabilities generated by matching to roads on G_+, same with $Pr^-(M(tr,G))$.

Here, G_+ represent all vertices and edges included in the ground-truth, i.e.: $G_+ = G_{gt} \cap G$. Likewise, $G_- = G - G_+$ and $Pr^+(M(tr, G) + Pr^-(M(tr, G) = Pr(M(tr, G)$. In addition, since both G_{gt} and $M(tr, G_{gt})$ are not obtainable in practice, we usually replace the denominator term with other factors, depending on which map-matching method applied, for normalisation purposes. The reason for such normalisation is to make sure each quality score is comparable to the scores of other trajectories so that the overall score is meaningful. More importantly, with no G_{gt} given, both G_+ and G_- become unknown. Hence, in this work, we will introduce an effective way to identify G_+ and G_- accurately.

In terms of the map quality, we also define the quality measure based on G_+ and G_-:

Definition 8. *(Map Quality) Given a weighted map G which consists of G_+ and G_-, the map quality $Q_M(G)$ is defined as:*

$$Q_M(G) = \sum_{e \in G_+} \omega_e - \sum_{e \in G_-} \omega_e$$

In general, the weight ω_e usually represent the importance of the road e. It can be defined in various ways according to the applications, like the number of trajectories passing through it, the betweenness centrality or the road width. Besides, we omit the importance of road nodes when measuring the map quality as the weight of a node is strongly correlated with the weights of its adjacent roads.

Overall, the map-trajectory quality is defined as:

$$Q(R, G) = \sum_{tr \in R} Q_T(tr, G) + Q_M(G) \tag{5.1}$$

Overall, we define our map-trajectory co-optimisation process as follows:

Definition 9. *(Map-Trajectory Co-Optimisation) Given a trajectory set R and a road network G, which contains both correct and erroneous map components, the map-trajectory co-optimisation finds a sub-map $G_{opt} \subset G$, so that $\forall \overline{G} \subset G$, $Q(R, \overline{G}) \leq Q(R, G_{opt})$.*

Optimally, $Q(R, G_{opt})$ reaches the maximum when $G_{opt} = G_+$. However, it is almost unreachable since the exact G_+ and G_- are unknown without given the ground-truth map G_{gt}. Therefore, we propose an iterative co-optimisation framework to gradually approach G_{opt} so that the final $Q(R, \overline{G})$ is the maximum obtainable. We will introduce the detail implementation of Q_T, Q_M and the way to identify G_+ and G_- in Section 5.3.

5.2.2 Co-Optimisation Framework

For better understanding, we summarise the main symbols used in this chapter in Table 5.1.

In general, our iterative co-optimisation framework consists of three components: map update, map-matching and the co-optimisation model. In addition, we conduct a data preprocessing step before the iteration to filter our irregular input and generate initial map-matching results and unmatched trajectories. Fig. 5.1 and Algorithm 1 illustrates the workflow of our iterative co-optimisation framework, which consists of the following procedures:

Data Preprocessing: In data preprocessing, the data filter removes abnormally short or sparsely sampled trajectories($\Delta t > 120s$) from raw trajectories and generates the input trajectory set R. Subsequently, an initial map-matching process is performed to generate the input of the first iteration,

TABLE 5.1: Main symbols and associated meanings

Symbol	Meaning
Overall input & output	
R, G	Trajectory set and road network
$M(R, G)$	Map-matching results of R on G
$IL(G, R)$	Inverted list index for map refinement
$I(R)$	Spatial index for running time optimisation
During the i-th iteration	
R_*^i	Unmatched trajectory set
ΔG^i	Newly inferred road network
$M_K(R, G^i)$	Top k matching results of R on map G^i
$M'_K(R, G^{i'}), R_*^{i'}, G^{i'}$	Temporary result before refinement
Co-optimisation model	
Cnf_e, Inf_e	Confidence score and influence score of e
$Cert(M_K(tr, G))$	Certainty score of the top-k matching result
$Q_T, Q_M(G)$	Quality of trajectory-matching and map
$Q(R, G), \Delta Q^i$	Map-trajectory quality and benefit of iteration i

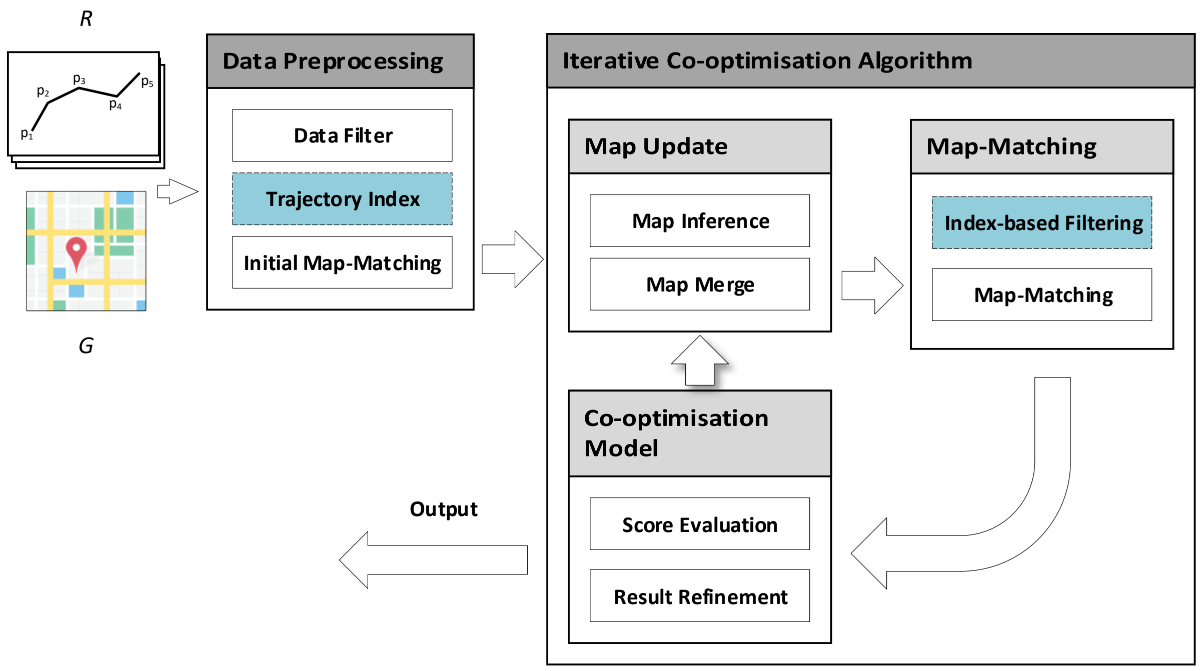


FIGURE 5.1: Framework overview

including the unmatched trajectory set R_*^0 and the matching result $M_K(R, G^0)$. Note that the symbol K in $M_K(R, G^0)$ represent the top-K map-matching results for each trajectory, which will be explained in Section 5.3.

The iterative process starts directly after the preprocessing. In each iteration i, we perform three main steps:

Map Update: The map update takes R_*^{i-1} as input and infers new road network elements $\Delta G^i =$

Algorithm 1 Map-trajectory co-optimisation algorithm

```
Input: (1) raw trajectory set $R'$, (2) initial map $G^0$
Output: (1) final graph $G$, (2) trajectory matching result on final graph $M(R, G)$
 1: procedure CoOptimisationProcess($R', G^0$)
 2:     $R \leftarrow DataFilter(R')$                                    ▷ Input data cleaning
 3:     $\Delta Q \leftarrow 0$                                          ▷ Benefit value
 4:     $M_K(R, G^0), R_*^0 \leftarrow MapMatching(R, G^0)$
 5:     $i \leftarrow 1$                                                 ▷ Iteration count
 6:     while $\Delta Q \geq 0$ do
 7:         $\Delta G^i \leftarrow MapInference(R_*^{i-1})$              ▷ Assign Confidence scores
 8:         $G'^i \leftarrow MapMerge(G^{i-1}, \Delta G^i)$
 9:         $M_K(R, G'^i), R_*'^i \leftarrow MapMatching(R, G'^i)$       ▷ Assign Influence scores
10:         $IL(\Delta G^i, R) \leftarrow InvertedListIndexGen(M_K(R, G'^i))$
11:         $G^i, G_-^i, \Delta Q \leftarrow ScoreEvaluation(G'^i)$
12:         $M_K(R, G^i), R_*^i \leftarrow ResultRefinement(R, G^i, G_-^i, IL(\Delta G^i, R), M_K(R, G'^i), R_*'^i)$
13:         $i \leftarrow i + 1$
14:     $G \leftarrow G^{i-1}$
15:     $M(R, G) \leftarrow selectthebestmatchingresultforeachtrajectoryfromM_K(R, G^{i-1})$
16: return $G, M(R, G)$
```

$(\Delta V^i, \Delta E^i)$ through the map inference algorithm. ΔG^i is further merged with G^{i-1} and forms a new temporary map $G^{i'}$. The confidence score Cnf_e for each new edge in ΔG^i is generated during the map inference.

Map-Matching: For each trajectory in R, we match it to the temporary road network $G^{i'}$. The map-matching process generates a temporary map-matching result $M_K(tr, G^{i'})$ for each trajectory and a collection of temporary unmatched trajectory set $R_*^{i'}$. In addition, we compare each map-matching result $M_K(tr, G^{i'})$ with its result from last iteration $M_K(tr, G^{i-1})$. Since the changes on map-matching results are only caused by the map update of the current iteration, we generate the *influence score*, denoted by Inf_e, for each new road e according to its contribution to changing the current map-matching results. After the map-matching process, we build an inverted list index for the new graph elements and new matching results. We create an index entry for each newly updated road edge and collect all trajectories whose current map-matching result contains this road edge. Such index will be used in the result refinement step (line 12 in Algorithm 1) in the following co-optimisation model.

Co-optimisation Model: As depicted in line 11,12, firstly, we identify the correctness of each new edge $e \in \Delta G^i$ by evaluating the scores(Inf_e and Cnf_e) generated in aforementioned steps. Then, edges falling into G_- are to be removed and an overall benefit ΔQ is calculated to evaluate the quality improvement during the current iteration. The final G^i is then determined after removing G_- from $G^{i'}$. In addition, since the G^i_- has participated in the latest map-matching process, we search through the index $IL(R, \Delta G^i)$ to find all trajectories whose matching result contains removed edges and perform another $MapMatching()$ process to update their matching results. Eventually, G^i, $M_K(R, G^i)$ and R^i_* will be sent to the next iteration or output, if the iteration ends ($\Delta Q \leq 0$), as the final result G and $M(R, G)$.

Note that the blue blocks in Fig. 5.1 are designed for running time optimisation, which will be introduced in Section 5.4.

We implement our framework using HMM-based map-matching [90, 137] and two different map inference methods, namely KDE-based [18] and trajectory clustering (TC) [82, 134], respectively, to show the flexibility of our system. It is worth noting that our framework is not tailored to those methods. Theoretically, all of the existing probability-based map-matching and map inference algorithms can fit into our framework since they can easily generate influence score and confidence score with moderate modification. We will introduce the details of our co-optimisation model in the next section with a brief explanation of such a claim.

5.3 Co-Optimisation Model Design

As mentioned above, the main objective of the co-optimisation model is to gradually improve the quality of both trajectory matching results and the map through the iteration process. Therefore, for each iteration, the model mainly focuses on three tasks: 1) evaluate the correctness of each newly updated road according to its influence and confidence scores; 2) calculate the quality improvement during the current iteration; and 3) remove the incorrect roads and refine the corresponding map-matching results. In this section, we mainly introduce these three steps.

5.3.1 Co-optimisation Quality Evaluation

Considering the logical relevance, we introduce the second step first. As formally defined in Section 5.2, the co-optimisation objective is to consistently improve the map-trajectory quality score, i.e. $Q(R, G)$, over the iterative process. Therefore, to better understand the model design, we first examine the function $Q(R, G)$.

For any iteration $i (i > 0)$, the input consists of the trajectory set R, the map from the last iteration G^{i-1}, the previous map-matching result $M(R, G^{i-1})$ and the unmatched trajectory set R_*^{i-1}. Hence, in the i-th iteration, the map update process first generates a temporary road network ΔG^i from unmatched trajectories R_*^{i-1}. Since ΔG^i contains both correct and erroneous road updates, after we merge ΔG^i and G^{i-1} into $G^{i'}$, the benefit of current iteration is defined as $\Delta Q^i = Q(R, G^{i'}) - Q(R, G^{i-1})$, which is equivalent to:

$$\Delta Q^{i'} = Q_T(R, G^{i'}) - Q_T(R, G^{i-1}) + Q_M(G^{i'}) - Q_M(G^{i-1}) \tag{5.2}$$

As we assume no erroneous road is included in G^{i-1}, both of the quality scores from the last iteration, namely $Q_T(R, G^{i-1})$ and $Q_M(G^{i-1})$, are fixed. Therefore, the goal of the current iteration is to increase the scores $Q_T(R, G^{i'})$ and $Q_M(G^{i'})$ so that $\Delta Q^i > 0$. According to Definition 8, $Q_M(G^{i'}) - Q_M(G^{i-1}) = Q_M(\Delta G^i)$. Also, the difference between $Q_T(R, G^{i'})$ and $Q_T(R, G^{i-1})$ is caused by merging ΔG^i. Hence, to further improve ΔQ^i, we need to remove more erroneous roads in ΔG^i. However, there are two issues that remain unsolved: 1) how to identify correct and wrong roads with no ground-truth given, and 2) how to quantify the change of Q_T and Q_M caused by each road removal/insertion.

The influence score Inf_e and confidence score Cnf_e are proposed to address these two issues. For each newly updated road $e \in \Delta G^i$, the influence score is generated during the map-matching step by summarising the probability changes of all trajectory matching results after e is added to the map. Hence, it measures how influential the new road e is. In contrast, the confidence score is generated during the map inference process. It is defined by the number of unmatched trajectories involved in generating e, which represents our confidence in inferring it. Intuitively, a road with a high influence and confidence score is more likely to be a correct road. However, either one of the scores alone is not sufficient to measure its correctness, for example, a short-cut road inferred from a noisy trajectory (high Inf_e but low Cnf_e) or a popular road with the wrong direction (low Inf_e and high Cnf_e). Such a feature can be utilised to distinguish the correct roads from the outliers.

On the other hand, both Inf_e and Cnf_e are strongly related to quality measurements Q_T and Q_M, respectively. Regarding the influence score, in the probability-based map-matching process, the probability difference between $Pr(M(tr, G^{i'}))$ and $Pr(M(tr, G^{i-1}))$ is completely caused by the insertion of ΔG^i. Hence, if e is the only new road appeared in $M(tr, G^{i'})$, the influence score of e on trajectory tr is defined as $\left|Pr(M(tr, G^{i'})) - Pr(M(tr, G^{i-1}))\right|$. In fact, as we only add new roads to the map without removing any existing roads, it is easy to deduce that the $Pr(M(tr, G^{i'}) \geq Pr(M(tr, G^{i-1}))$ always holds. If multiple new roads appear in $M(tr, G^{i'})$, the total probability change will be split and distributed to each new edge involved according to a weight factor $\mu_{e,tr}$ proportionate to the length of the new matching sequence introduced to $M(tr, G^{i'})$ by this road. Therefore, the overall influence score of e is defined as follows:

$$Inf_e = \sum_{tr \in R} \mu_{e,tr}(Pr(M(tr, G^{i'}) - Pr(M(tr, G^{i-1})))) \tag{5.3}$$

Hence, it is easy to deduce that:

$$Pr(M(R, G^{i'})) - Pr(M(R, G^{i-1})) = \sum_{e \in \Delta G_i} Inf_e \tag{5.4}$$

Assuming that the correctness of e is determined, the trajectory matching quality difference in ΔQ^i can then be represented by Inf_e, i.e.:

$$Q_T(R, G^{i'}) - Q_T(R, G^{i-1}) = \sum_{e \in \Delta G^i_+} Inf_e - \sum_{e \in \Delta G^i_-} Inf_e \tag{5.5}$$

Likewise, the confidence score of a new road e is defined as the number of unmatched trajectories involved in inferring e. Since each newly inferred road $e \in \Delta G^i$ derives from a set of unmatched trajectories in R^{i-1}_*. Those trajectories imply our confidence of inferring road e. It is equivalent to the ω_e defined in Definition 8.

Overall, the benefit of the current iteration is represented as:

$$\Delta Q^i = \sum_{e \in \Delta G^i_+} (Inf_e + Cnf_e) - \sum_{e \in \Delta G^i_-} (Inf_e + Cnf_e) \tag{5.6}$$

5.3.2 Road Correctness Identification

To determine the value of ΔQ^i, an algorithm is required to better distinguish the correct and erroneous roads. According to our design principle, a road is more likely to be correct if its Inf_e and Cnf_e are high. Therefore, we first propose a more optimised influence score for better identification. In terms of the implementation of Inf_e and Cnf_e, we take the HMM map-matching algorithm as an example, in which the influence score of every new road is generated during the HMM process. Since the probability in the HMM model is calculated by multiplying the emission probabilities and transition probabilities along the matching sequence, the overall probability decreases exponentially when the trajectory gets longer. Hence, we normalise the influence score by finding the $|tr|$-th root of the value, i.e.:

$$Inf_e = \sum_{tr \in R} \mu_{e,tr} (Pr(M(tr, G^{i'}) - Pr(M(tr, G^{i-1}))))^{\frac{1}{|tr|}} \tag{5.7}$$

In addition, regardless of the normalisation problem amongst trajectories, comparing $Pr(M(tr, G^{i'})$ with $Pr(M(tr, G^{(i-1)}))$ of the same trajectory is also not enough to quantify the improvement. Note that a matching result is believed to be correct only when its probability is outstanding from other possible routes. Hence, we need to identify the relative change in probability value amongst all candidates rather than an absolute value difference between its previous optimal.

Accordingly, we propose a top-k HMM model which generates top-$k(k \geq 2)$ possible routes for each trajectory, denoted by $M_K(tr, G) = \{M_i(tr, G) | 1 \leq i \leq K\}$. The modification of the HMM model [90] can be easily achieved by storing top k highest probability preceding states for each candidate along the path, and the top-k possible paths can be generated during the backward loop. Through the introduction of multiple sub-optimal matching candidates, the following can occur: (1) if $Pr(M_1(tr, G))$ exceeds the rest candidates by a huge amount, it is more likely that $M_1(tr, G)$ is the correct matching result and the others are noise. On the contrary, (2) if multiple results have a similar probability to the highest one, it means that there is still a huge uncertainty in the matching result, which is more likely caused by some erroneous GPS samples or missing roads. Hence, we define the concept *"certainty"* for the trajectory top-k matching results, denoted by $Cert(M_K(tr, G))$. The certainty is calculated as follows:

$$Cert(M_K(tr, G)) = Pr(M_1(tr, G)) * (\sum_{i=1}^{K} -Pr(M_i(tr, G)) * \ln Pr(M_i(tr, G))) \tag{5.8}$$

The definition of certainty utilises the idea of entropy, which means the trajectory certainty is lower if its top-k results are more similar, implying that the matching results are very random. Overall, the optimised influence score is defined as:

$$Inf_e = \sum_{tr \in R} \mu_{e,tr} (Cert(M_K(tr, G^{i'})) - Cert(M_K(tr, G^{i-1})))^{\frac{1}{|tr|}} \tag{5.9}$$

Note that $Cert(M_K(tr, G^{i'})) \geq Cert(M_K(tr, G^{i-1}))$, so $Inf_e \geq 0$ always holds. Therefore, the influence score for an edge e is the collective results of all trajectory certainty changes caused by it, which is obtained by comparing the current matching result of each trajectory with the previous one.

On the contrary, the confidence score is easily obtainable in most of the map inference/update methods. For example, the post-processing map-matching step in KDE-based map inference [18] aligns every unmatched trajectory to an inferred road, and in the trajectory clustering method [134], one road is inferred from a cluster of unmatched trajectories with a certain number. Therefore, the way of generating a confidence score is applicable to other map inference/update methods with minor modification.

For road identification, we combine the two scores by multiplication and set a threshold θ_s to distinguish the correct and erroneous roads. The reasons for not using linear-based scoring function are that (1) the scaling of the influence score and confidence score is different, and (2) it is worse at of filtering roads with one score extremely high and one score fairly low, which is a popular pattern of erroneous roads. Eventually, the updated map ΔG^i is divided into G^i_+ to be kept and G^i_- to be removed according to the threshold θ_s. Optimally, the iterative process should continue until $G^i_+ = \emptyset$ as we remove G^i_- each time so ΔQ^i is always non-negative after the refinement step. However, for efficiency concerns, we terminate the iteration when $\sum_{e \in \Delta G^i_+} (Inf_e + Cnf_e) \geq \sum_{e \in \Delta G^i_-} (Inf_e + Cnf_e)$ as the benefit gained from the iteration is less significant than the erroneousness introduced.

5.3.3 Result Refinement

At the end of each iteration, since we remove edges G^i_- from the map $G^{i'}$ during the co-optimisation process, some of the current matching results are also affected due to its matching to the removed edges. Therefore, the main objective for result refinement is to eliminate the influence caused by removed edges to the matching results and unmatched trajectories, which can be achieved by a rematch process on the affected trajectories.

Considering the excessive cost of the map-matching process, in the result refinement step, we only apply map-matching on to the trajectories whose matching results actually contain removed roads. To achieve this goal, we maintain an inverted list index $IL(\Delta G^i, R)$ during the map-matching process, which is shown in line 10 of Algorithm 1. For the top k matching results of each trajectory, if it contains any newly inferred edge e, we store the trajectory ID under the entry of e in the index. Eventually, each entry e holds a list of trajectory IDs. This index is used in result refinement. As shown in Algorithm 2, for each edge e removed after the score evaluation, we search through the index and find all affected trajectories. The trajectories from various entries are collected into a final set, which is taken as the input of the rematch process. Finally, the rematch results are merged with temporary matching results to form the final iteration output.

5.4 Running Time Optimisation

Although aiming for optimal quality, our iterative co-optimisation algorithm may also lead to excessive running time. Note that each iteration contains a full rotation of map inference, map merge and

Algorithm 2 Result refinement algorithm

```
    Input: (1) trajectory set $R$, (2) refined map $G^i$ and removed map elements $G^i_-$, (4) index $IL(\Delta G^i, R)$, (5) temporary matching results $M_K(R, G'^i)$ and unmatched trajectories $R'^i_*$
    Output: (1) refined matching result $M_K(R, G^i)$ and unmatched trajectories $R'^i_*$
 1: procedure ResultRefinement($R, G^i, G^i_-, IL(\Delta G^i, R), M_K(R, G'^i), R'^i_*$)
 2:     $RID \leftarrow \emptyset$                                                ▷ Trajectory ID set
 3:     for all $e \in G^i_-$ do                    ▷ Find all trajectories with wrong matching results
 4:         if $e \in \Delta G^i$ then                                 ▷ Statement should always be true
 5:             $RID \leftarrow RID \cup IL(\Delta G^i, R).GetValue(e)$
 6:     $\overline{R} \leftarrow$ find trajectories in $R$ according to $RID$            ▷ Selected trajectory set
 7:     $M_K(\overline{R}, G^i), \overline{R}^i_* \leftarrow MapMatching(\overline{R}, G^i)$
 8:     $M_K(R, G^i), R^i_* \leftarrow \emptyset$
 9:     for all $tr \in R$ do
10:         if $tr \in \overline{R}$ then                          ▷ Check if the result of $tr$ is updated
11:             $M_K(R, G^i) \leftarrow M_K(R, G^i) \cup M_K(tr, G^i)$
12:             $R^i_* \leftarrow R^i_* \cup tr^i_*$
13:         else
14:             $M_K(R, G^i) \leftarrow M_K(R, G^i) \cup M_K(tr, G'^i)$
15:             $R^i_* \leftarrow R^i_* \cup tr'^i_*$
16: return $M_K(R, G^i), R^i_*$
```

map-matching algorithms, which is equivalent to a complete run of a classic map update algorithm; thus, the overall running time can easily exceed the existing map update methods by even one order of magnitude, let alone the time spent on the co-optimisation model. More importantly, the running time does not shrink by much as the iteration goes. In fact, the time spent on map inference, map merge and the co-optimisation model gradually reduces as the number of new roads decreases over iterations. However, since the map-matching process reads all trajectories in R each time and matches them to an enlarging map, its running time may even increase slightly. Moreover, since the map-matching process usually takes up most of the running time of an iteration, which is confirmed by our experiments (averaging 67% of running time under parallel processing), it will be very beneficial if we can reduce the cost of the map-matching process. As we are not aiming for improving the map-matching algorithm itself, in this section, our main goal is to minimise the number of map-matching calls in our iterative process.

According to our framework, the map-matching algorithm is called in three different phases, as shown in Algorithm 1:

- During the data preprocessing, an initial map-matching is called (line 4) to generate the initial matching results $M_K(R, G^0)$ and unmatched trajectories R_*^0. The former is used as the previous matching result when generating influence scores in the first iteration, while the latter is used as the input of the first-round map update. Since the initial matching result is required for every trajectory, it is clear that the map-matching in data preprocessing phase is compulsory for all trajectories.

- A map-matching phase is included in each iteration (line 9), which generates a new map-matching result and unmatched trajectory(s) for each trajectory in R. Since the change of map-matching result is caused by the latest map update, it is highly possible that some of the trajectory matching results are not affected by the newly updated map components. Therefore, the map-matching on those trajectories is redundant.

- During the result refinement phase (line 12), an additional map-matching process is called for all trajectories whose temporary matching result contains roads to be removed. It is guaranteed that every matching result changes after the rematch, so the input size of the rematch phase has been optimised.

According to the above analysis, the map-matching phase has not been optimised and contains great potential. Hence, in this section, we propose an index-based trajectory filtering strategy to reduce the cost of the map-matching phase in each iteration via the elimination of redundant map-matchings.

Intuitively, a new road can only affect the map-matching results of trajectories that are closed to the update location. Therefore, the idea of the trajectory filtering is to build a spatial index for trajectory points so that all passing trajectories can be found given a certain area. In our algorithm, once a new road is updated, we perform a range query covering the updated area and find all trajectories overlapping the query region. Optimally, the query range should cover all possible candidates for the road so that trajectories that do not overlap with any query region can be safely removed from the current iteration. Therefore, the main challenge is to define the query region.

According to the Definition 5, in a probability-based map-matching algorithm, a road participates the map-matching process of a trajectory only when it becomes a candidate of a trajectory point p_i or a segment $\overline{p_i p_{i+1}}$. More specifically, in the HMM map-matching algorithm, it appears when it (1) falls into the candidate range of a trajectory point p_i or (2) is in the shortest path between a candidate of p_i and a candidate of p_{i+1}.

We illustrate the two cases in Fig. 5.2. The red line $e_{ab} = (v_a, v_b)$ represents a new road update, while the green and blue polyline represent two trajectories. In terms of the blue trajectory, it is clear to sea that v_a is close to p_3 and e_{ab} is also a candidate of p_4. On the contrary, none of the green points are shown to be close enough to e_{ab}. However, we still need to map-match the green trajectory since e_{ab} is in the shortest path between r_2 and r_3, which are the candidate of green p_2 and p_3, respectively. Hence, we explain these two cases in detail as follows:

5.4.1 Direct Matching Area (DMA)

In terms of the first case, since the candidate range of p_i is a circle whose centre is p_i and radius is set by a parameter called candidate range θ_c(explained in Section 5.5.3), a new road e can be a candidate of p_i only when $dist(p_i, e_{ab}) \leq \theta_c$, where $dist(p_i, e_{ab})$ represents the Euclidean distance between p_i and its closest point on e_{ab}. We denote such area as *Direct Matching Area (DMA)*, which is shown as a red shaded area in Fig. 5.2. Any point within the area has a distance less than θ_c to e_{ab}.

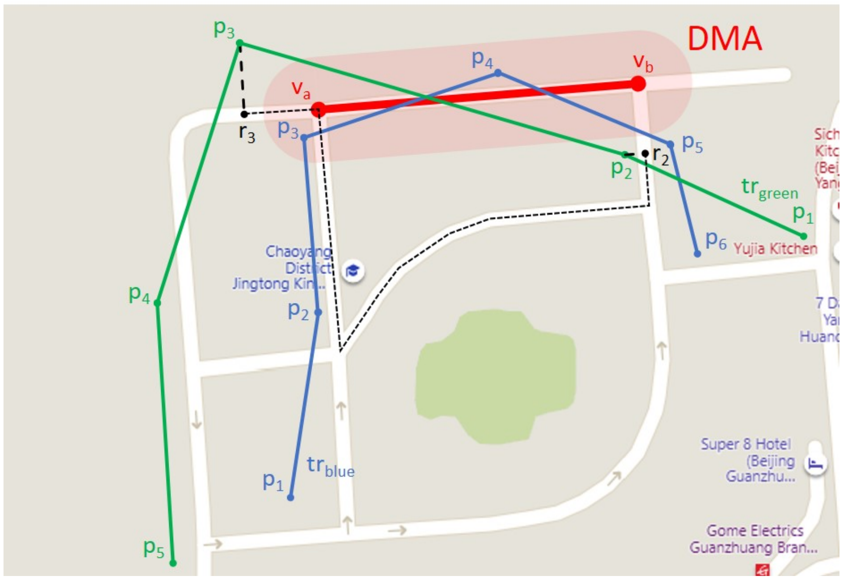


FIGURE 5.2: Example of two map-matching cases

5.4.2 Shortest Transition Area (STA)

Assume that G is the map before inserting e_{ab} and r_i represents a candidate point of p_i on G. The trajectory that falls in this category must satisfy two requirements:

- $\nexists p_i \in tr$ s.t. $p_i \in DMA$.
- $\exists r_i, r_{i+1}$ s.t. $dist_r(r_i, v_a) + e_{ab}.length + dist_r(v_b, r_{i+1}) < dist_r(r_i, r_{i+1})$ where $dist_r()$ represents the route-based distance.

In other words, the second requirement means the trajectory must have at least one pair of consecutive points p_i, p_{i+1} whose shortest distance reduces after inserting e_{ab} to the map. In Fig. 5.2, none of the green points fall into the *DMA* of e_{ab}; however, the previous route distance between r_2 and r_3, denoted as the black dash polyline, reduces after inserting e_{ab}. In fact, to achieve both requirements, the sampling rate of the trajectory should be very low to avoid being sampled when the vehicle is passing the road e_{ab}. We denote the query region of the second case as the *Shortest Transition Area (STA)*.

To define the *STA* region more precisely, we need to consider the maximum possible distance between p_i and p_{i+1}. Note that, since it is reachable from their candidate r_i to r_{i+1} within the time interval $\Delta t_i = p_{i+1}.t - p_i.t$ given the vehicle maximum speed v_{max}, we can deduce that

$$dist_r(r_i, r_{i+1}) \leq v_{max} * \Delta t_i \tag{5.10}$$

should always hold. Moreover, as the transition must passes e_{ab}, it is clear that the distance between r_i/r_{i+1} and the closer endpoint of e_{ab}, i.e. v_a/v_b is no more than $v_{max} * \Delta t_i - e_{ab}.length$. Further considering the maximum distance θ_c between p_i and its candidate r_i, we propose two ways of defining *STA*.

- **STA-Circle**: Since the route-based distance between r_i/r_{i+1} and v_a/v_b is always no less than their Euclidean distance, we can safely define two circles, centred at the endpoint v_a/v_b, whose radius is set to $rad_{STA-C} = \theta_c + v_{max} * \Delta t - e_{ab}.length$. Note that we replace Δt_i with the maximum sampling interval Δt of the trajectory set as we are not searching for a particular trajectory. The index is queried with these two circles to extract all *STA* trajectories of road e_{ab}.

- **STA-Route**: We start from both endpoints of e_{ab} and traverse the road network (e_{ab} excluded) until we reach the maximum distance $v_{max} * \Delta t - e^*.length$. Then, we create a *DMA* for all roads traversed and form two *STAs* for v_a and v_b, respectively. Note that the traverse of the start endpoint v_a should be done on the reverse road network of G while considering direction.

Eventually, a trajectory is selected only when a pair of its consecutive points p_i and p_{i+1} falls into the *STA* of v_a and v_b, respectively. In terms of the query effectiveness, we compare the query ranges of these two definitions in Fig. 5.3. Although *STA-Circle* (shown in red) only needs to perform two circle range queries, its query area is much larger than *STA-Route* (shown in grey), which leads to more unnecessary trajectory matching.

In general, for a particular edge e_{ab}, it is obvious that the size of the *STA* is usually much larger than the *DMA*. Since multiple new roads are updated within one iteration, larger query ranges can result

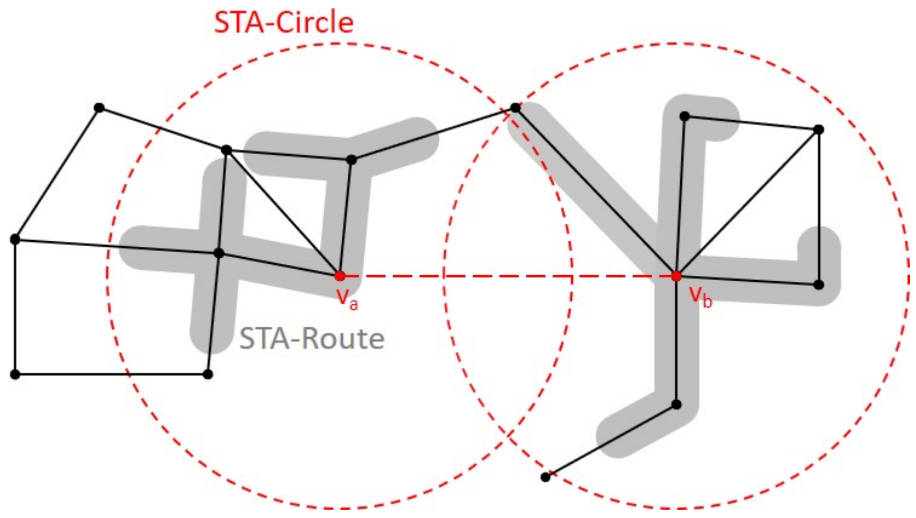


FIGURE 5.3: Example of *STA-Circle* and *STA-Route*

in less effective trajectory filtering or even overlapping queries, which leads to more processing time. However, the *STA* guarantees all affected trajectories can be found so that their matching results are corrected eventually. In fact, it is worth noting that although the *DMA* can raise correctness issues for map-matching results, it can always be remedied by an extra post-processing map-matching before the final output. The main issue is the approximate algorithms can also affect the precision of the influence and confidence scores, which further determine the correctness of new edges. However, in most cases, it is good enough to simply use *DMA* for trajectory filtering, especially when the trajectory sampling rate is high or removed roads are long, which is proved by our experiments.

Finally, the implementation of the filtering index $I(R)$ is straightforward. During the data preprocessing step, since the trajectory dataset R is fixed, we build a Sort-Tile-Recursive(STR) partitioning R-Tree [77] for all trajectory points in R; the points are indexed by their coordinates and each index entry only contains its corresponding trajectory ID and its number in the sequence. Before the map-matching phase (line 4 in Algorithm 1) in each iteration, we perform the *DMA* and *STA* queries for every new road e in ΔG and merge the resulting trajectory ID lists to form the final input of the map-matching phase.

5.5 Evaluations

Since the main goal of our co-optimisation model is to improve the quality of both map-matching and map update results, the evaluation of both results is required in our experiments. However, it is worth noting that the existing evaluation methods for map-matching and map update are not applicable in our case. Regarding the map-matching, since the generation of a ground-truth dataset requires labour-intensive work [72], only several ground-truth datasets are publicly available and all of them are extremely small. Such datasets are inappropriate in our experiments as we require large trajectory datasets for map update. Hence, we introduce a two-step evaluation process: we first use a publicly available ground-truth map-matching dataset to validate the accuracy of our map-matching algorithm and tune the parameters. In the second step, we bring out a large-scale trajectory dataset and use our map-matching method, which has been proved to be accurate, to generate map-matching results as ground-truth. This dataset will be used to evaluate the performance of our entire co-optimisation algorithm.

5.5.1 Experimental Settings

Dataset Description

We prepare two datasets for different evaluation purposes. As depicted in Table 5.2, the first dataset, termed as *Global* [72], is introduced to evaluate the accuracy of our map-matching algorithm. Then, we evaluate our co-optimisation algorithm on a large-scale trajectory dataset, denoted by *Beijing-L*, which is obtained from 5,000 taxis running in Beijing for 5 days. It is worth noting that the trajectories in *Beijing-L* only covers less than 50% of the roads(46.51%), which means more than half of the roads on the map are not travelled by any trajectory. Those roads are unable to be found through the map update if they are missing from the map. Therefore, we select two sub-areas from *Beijing-L* with heavier traffic and generate new datasets with smaller data scale but higher trajectory coverage, named as *Beijing-S* and *Beijing-M*. Most of the experiments are performed on *Beijing-S* and *Beijing-M* due to their higher coverage. Meanwhile, we test the scalability of our model on the full dataset *Beijing-L*.

TABLE 5.2: Dataset specification

Name	No. of Trajectories	Average Trajectory Points	Map Size (v/e)	% of Road Visited	% of Road Visited ≥ 5
Global	100	2473	N/A	N/A	N/A
Beijing-S	12,974	37	2,315/4,485	72.83	45.11
Beijing-M	77,676	47	12,312/22,581	74.94	54.26
Beijing-L	394,203	46	281,862/602,456	46.51	26.06

System Configuration

We ran our experiments on a server containing two Intel(R) Xeon(R) CPU E5-2630 with 10 cores/20 threads at 2.2GHz each, 378GB memory and Ubuntu 18.04. Our code mainly runs in Java-1.8 with the

KDE-based map inference running in Python-2.7, which is a modified version based on the original implementation in [18].

Baseline Methods

We compare our co-optimisation algorithm with the most outstanding solution for the map update problem, named CrowdAtlas [134]. The CrowdAtlas solution consists of an HMM map-matching algorithm, a Trajectory Clustering (TC) map inference and a map merge algorithm which utilises road closeness and road extension. By implementing the same set of algorithms as in CrowdAtlas and adding our iterative co-optimisation model on top of it, we evaluate the quality improvement of both the map update and map-matching results achieved by our model. In addition, we implement a KDE-based map inference [18] solution as an alternative inference method to show the flexibility of our framework and its ability to improve the map quality over any existing map inference/update methods.

5.5.2 Evaluation Metrics

We prepare four types of data for evaluation on *Beijing* dataset: (1) the ground-truth roadmap G_{gt} provided by the dataset, (2) an outdated roadmap G^0 as the map input, which is obtained by randomly remove a subgraph from G_{gt} (this method is widely adopted in other map update works [133, 143]), (3) raw trajectory dataset R as the trajectory input and (4) ground-truth matching result $M(R, G_{gt})$ generated by performing our map-matching algorithm on the ground-truth map G_{gt}. The goal of our algorithm is to infer as many edges as in $G_{gt} - G^0$ with less erroneous roads introduced so that the final updated map is close to G_{gt}, and the map-matching result is close to $M(R, G_{gt})$.

Like many of the existing map inference [4, 6, 82] and map-matching [89, 136] works, our algorithm is evaluated by precision/recall/F-measure, which is defined slightly differently in map-matching and map update evaluations. In the map-matching phase, we evaluate the difference between the matching results $M(R, G)$ and the ground-truth results $M(R, G_{gt})$. Since each matching result consists of a sequence of road edges in road network G, for each trajectory tr, the precision is defined as follows:

$$precision(tr) = \frac{\sum_{e \in M(tr,G) \cap M(tr,G_{gt})} len(e)}{\sum_{e \in M(tr,G)} len(e)} \tag{5.11}$$

Regarding the map evaluation, we define that an inferred road e_i is **equivalent** to a removed road e_j if and only if both the start and end points of $e_j \in G_{gt}$ are the closest points to the respective endpoints of e_i and the distances do not exceed a given threshold(set as θ_c used in map-matching).

Hence, given the updated map G, the input map G^0 and the removed subgraph $G_{gt} - G^0$, the precision is defined below:

$$precision(G) = \frac{\sum_{e \in (G-G^0) \cap (G_{gt}-G^0)} len(e)}{\sum_{e \in (G-G^0)} len(e)} \tag{5.12}$$

Note that both definitions consider the length of the road as they can better reflect the importance of each road. Meanwhile, we omit the definition of recalls and F-measures as they are standard and easy to deduce from the definition of precision.

5.5.3 Map-Matching Evaluation

We first perform map-matching on the *Global* dataset to tune the parameters of our algorithm. Same as the standard HMM-based map-matching model, there are three fields configurable in our map-matching algorithm: 1) θ_c specifies the maximum radius used for searching the candidates; In terms of the probability model; 2) the σ defined in the emission probability represents the average measurement error of the GPS device; while 3) β is used to specify the weight of transition probability. Fig. 5.4 shows the different experimental results when changing θ_c. Here, we fix the factors $\sigma = 4$ and $\beta = 0.08$ according to the empirical result presented in [90] and alter the candidate radius from 5 to 100 meters. Fig. 5.4 shows that our map-matching method reaches around 98% f-score so that we can safely use it to generate the ground-truth results of the *Beijing* dataset. Also, we set $\theta_c = 50m$ in the following experiments as it can achieve near-optimal matching quality with decent running time.

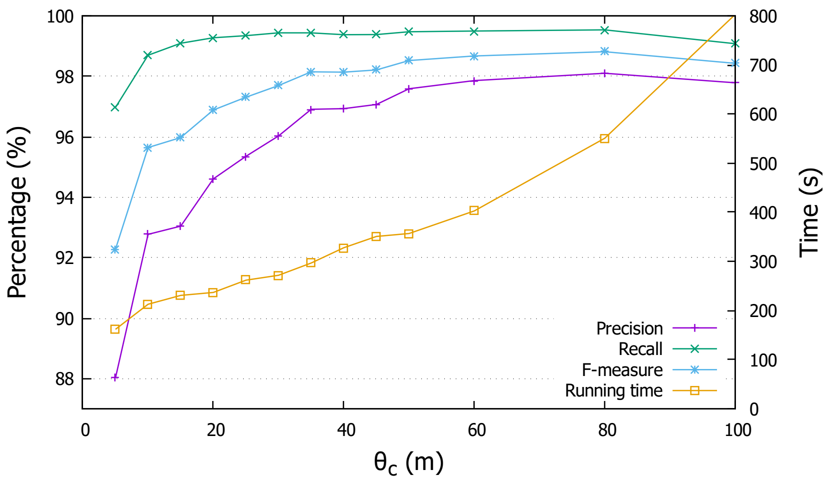


FIGURE 5.4: Map-matching performance by varying θ_c

5.5.4 Co-optimisation Algorithm Evaluation

To ensure the legit use of the baseline methods, we use the same settings as in the original papers of TC(CrowdAtlas) and KDE map inference methods. In the rest of this section, we purely focus on the discussion of the effectiveness and efficiency of our co-optimisation model and the influence of parameters, including the following:

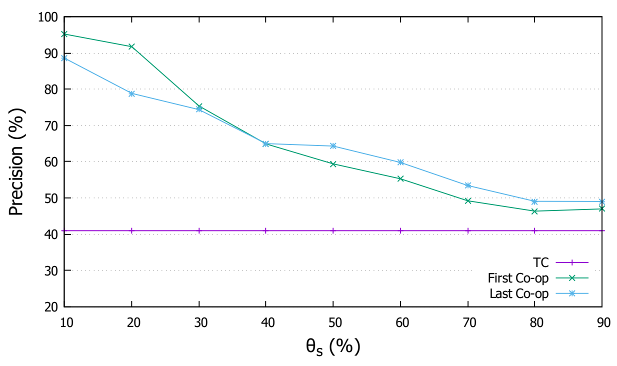
- Breakpoint extension range(λ_b): When an HMM break occurs during the map-matching process, we extend the breakpoint both ways along the trajectory to generate an unmatched sub-trajectory. The extension terminates once the next adjacent point has a road match within its λ_b range. This parameter is also used in other map update algorithms, like CrowdAtlas [134] and HyMU [133] to control the length of the unmatched trajectories. The default value is $\lambda_b = 15m$.
- Correct road ratio($\theta_s\%$): After all new roads receive their influence score and confidence score, we take the $\theta_s\%$ of roads with the highest combined scores as G_+ and the rest as G_-. The default value is $\theta_s = 20\%$.
- Map-matching result length(k): The length of the top-k map-matching sequences for each trajectory. Default value $k = 3$.

In addition, we change the road removal percentage($rm\%$, default value 6%) when deriving the input map from the ground-truth to test the performance of our algorithm when facing different map quality.

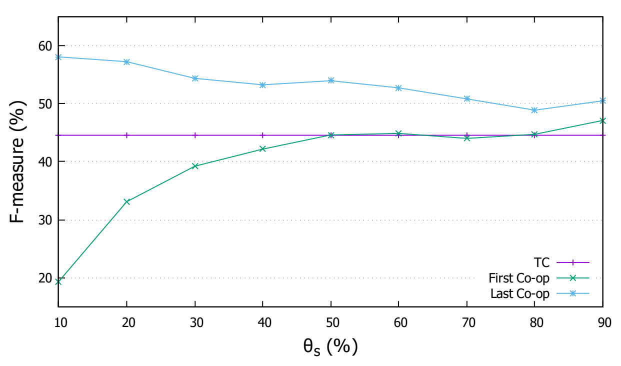
In the following experiments, the parameters are set to default values if not mentioned.

Performance Evaluation

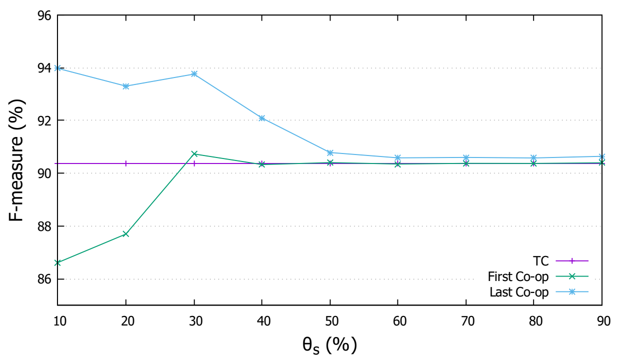
We first demonstrate the overall performance of our method. Fig. 5.5a-5.5c shows the quality of map-matching and map update when varying θ_s. Comparing with the baseline method CrowdAtlas, our method guarantees an overall better map-matching and map update accuracy after the iterative co-optimisation process. In contrast, with the help of our influence and confidence score, we can significantly improve the update accuracy by strictly filtering out low-quality candidates. Therefore, when θ_s is set to a low percentage, the precision of our map update result is significantly higher than the baseline as shown in Fig. 5.5a. However, due to its low penetration rate, lots of correct roads with relatively low scores are also removed in the first few iterations, so the overall f-measure and the corresponding map-matching result receive penalties. Even though, after enough rounds of iterations, the final result still achieves a higher F-measure while maintaining its superior precision.



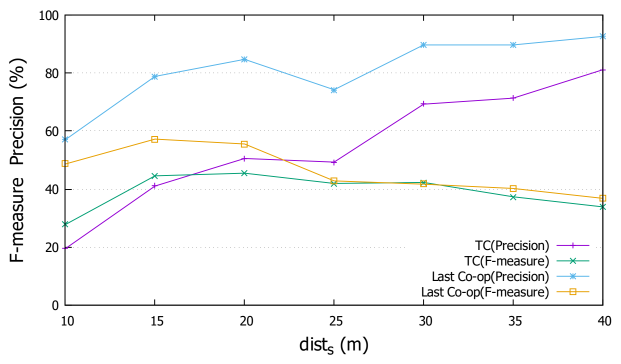
(a) Map quality by varying θ_s (Precision)



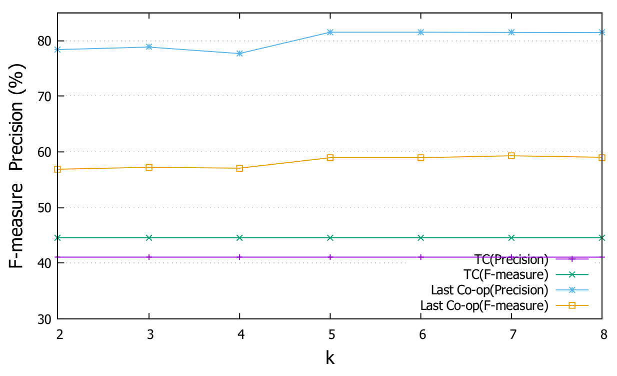
(b) Map quality by varying θ_s (F-measure)



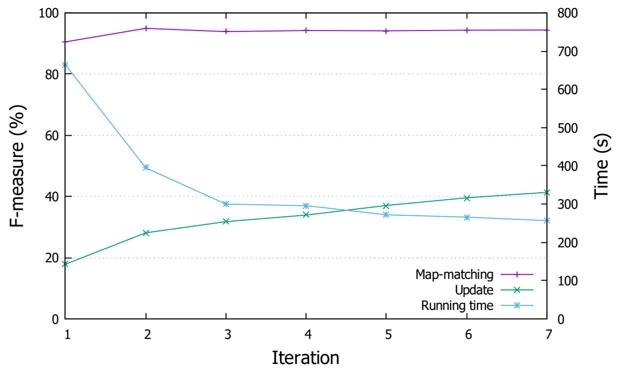
(c) Matching quality by varying θ_s (F-measure)



(d) Map quality by varying λ_b (P/F)



(e) Map quality by varying k (P/F)



(f) Map improvement with iteration (F-measure)

FIGURE 5.5: Effectiveness test for co-optimisation model

On the other hand, when $\theta_s \geq 50\%$, although most of the new roads are preserved, our algorithm still receives better performance than the baseline method, meaning that most of the erroneous roads are measured as low score items and get removed by our model.

Meanwhile, as shown in Fig. 5.5c, the map-matching results usually do not receive the same amount of improvement as the map update since the erroneous roads generated by the baseline method only affect in a small portion of the trajectory map-matching results. Hence, the removal of erroneous roads barely affects the recall and only improve the precision slightly. Moreover, we also conduct our comparison on KDE map inference, since the KDE baseline method focuses more on precision rather than recall, it delivers very high precision (91.50%) but extremely low recall (15.97%) in our framework, and our method improves its F-measure by nearly 20 percent (27.18% to 46.84%), which shows the effectiveness and flexibility of our framework when applied to different map update algorithms.

We also measure the performance by varying breakpoint extension distance λ_b. As shown in Fig. 5.5d, when $\lambda_b \leq 20m$, the map quality is much higher than the baseline, especially the precision. However, the improvement is negligible when λ_b becomes too large as the number of unmatched trajectories decreases drastically and only the distinctive trajectories are selected, which certainly generates correct roads (TC reaches > 80 precision when $\lambda_b = 40$). Even in such a case, our method can still achieve higher precision than the baseline method.

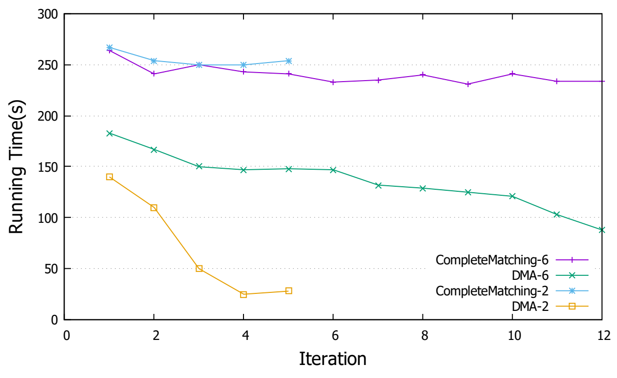
Moreover, we vary the rank length k from 2 to 8 to test the effectiveness of our top-k map-matching and certainty calculation. However, as depicted in Fig. 5.5e, the performance does not see a significant change especially when $k > 5$ since in most cases, only a few matching sequences are considered to be the matching candidate of a trajectory. Hence, introducing more irrelevant candidates does not affect the result.

Iteration Evaluation

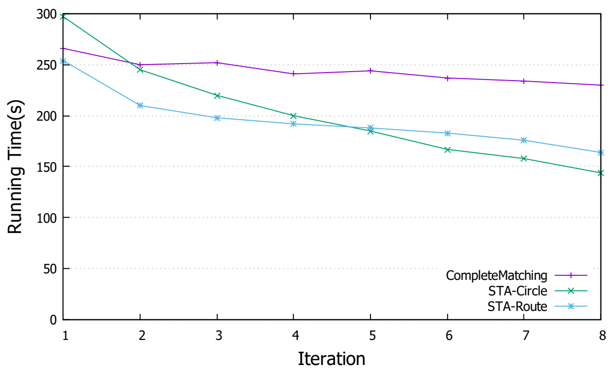
Since the effectiveness of our co-optimisation model has been proved above, we also measure the effectiveness of the iterative process. Fig. 5.5f demonstrates the change of map-matching and map quality during the iterations. It can be observed that both the map-matching and the map quality improves monotonically, which means the iterative process works as our expectation and guarantee a quality improvement. Besides, the running time of each iteration decreases as the number of newly inferred roads drops drastically. However, due to the constant cost of the map-matching process, the time cost for each iteration is still notable, which is the main reason for our running time optimisation.

Trajectory index Evaluation

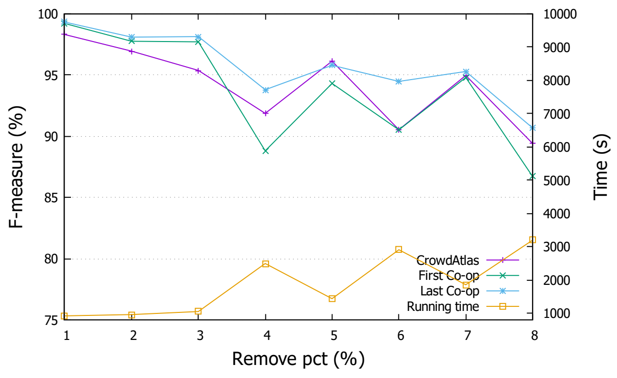
As introduced in Section 5.4, we propose three types of query ranges, namely the *DMA*, *STA-Circle* and *STA-Route* to accelerate the map-matching phase during each iteration. Fig. 5.6a shows the improvement of running time of our *DMA* index as the iteration goes. The *CompleteMatching* represents the running time of iteration with no index, and we also compare the performance by different road removal percentages (2% and 6%). Compared to the original solution whose running time does not change by much as the iteration goes, the running time of the *DMA* reduces significantly, which meets our expectation. As the number of newly inferred roads decreases as the iteration goes up, fewer trajectories should be affected by new road insertion. Also, the performance benefit shrinks



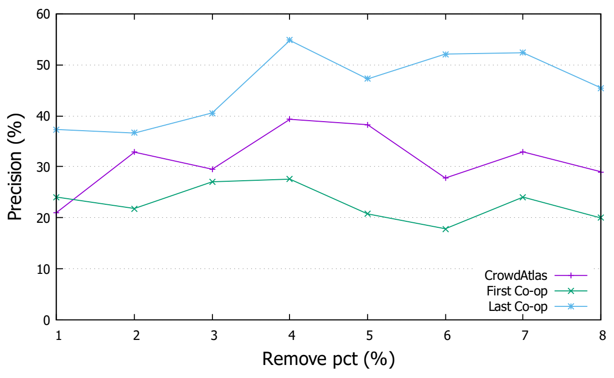
(a) Time saved by *DMA* over iterations



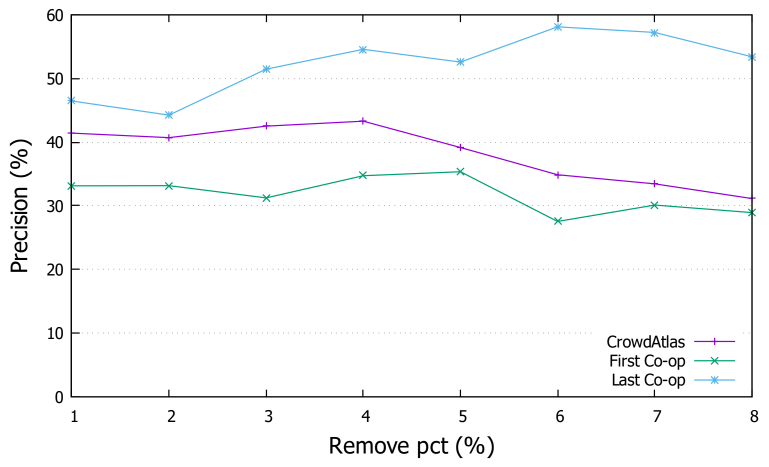
(b) Time saved by *STA-Circle* and *STA-Route* over iterations ($rm\% = 4$)



(c) Matching quality and time by varying $rm\%$ (F-measure)



(d) Map quality by varying $rm\%$ (F-measure)



(e) Map improvement on Beijing-M (F-measure)

FIGURE 5.6: Efficiency and scalability test for co-optimisation model

when the percentage of road removal increases due to more trajectories being affected by road inferences. Meanwhile, our experiment shows that the loss of accuracy in the *DMA* is not a big concern as averaging 94% of the trajectories that can be matched to the road are found by the *DMA* query, and the missing trajectories have negligible influence on the quality of the final co-optimisation results ($< 1\%$ of the F-measure change). On the contrary, although the *STA-Circle* and *STA-Route* have no accuracy issues, their running time is much slower than the *DMA* due to the larger search range. As shown in Fig. 5.6b. Comparing with the original solution, both the *STA-Circle* and *STA-Route* receives almost no boost at the first few iterations and later gains some benefit from the reduced inference road size. However, their poor performance is caused by different reasons. Due to the extremely large search range of the *STA-Circle*, the circles for different inferred roads may overlap with each other, which causes redundant searches especially when the number of roads is large. On the other hand, the *STA-Route* has few problems with the overlapping, but the time spent on finding nearby roads of each inferred road is significant and does not reduces by much as the iteration goes, which cause the poor performance even after a few iterations.

Overall, our index-based trajectory filtering helps reduce the running time as the iteration goes, which accumulates a significant amount of time saved over the iterative process. Amongst all three candidates, the *DMA* has the best time reduction with negligible damage on the accuracy aspect, which is an overall better choice in most of the cases where the accuracy is not a serious concern.

Scalability Evaluation

We test the scalability of our algorithm by varying the percentage of road removed from the map. Fig. 5.6c and 5.6d shows that the quality of map-matching and map update when varying $rm\%$. It is clear to see that our co-optimisation algorithm outperforms the CrowdAtlas baseline constantly(since we choose the $\theta_s = 20\%$, the F-measure of the first iteration is expected to be lower than the baseline). Regarding the running time shown in Fig. 5.6c, we can find there are two peaks appearings at 4% and 6%; this is because the baseline method performs badly during these runs and our co-optimisation algorithm spends more iterations to make the final results much better than the initial one, which shows good robustness of our solution.

In addition, we run our algorithm on the *Beijing-M* and *Beijing-L* dataset to test the scalability. Fig. 5.6e shows the map quality improvement on Beijing-M dataset. We can clearly see a similar improvement as in *Beijing-S*. In fact, the same improvement can be found in the*Beijing-L* dataset (35% to 44% when $rm\% = 4$). However, since one round of map-matching on *Beijing-L* dataset takes around 3 hours, even with the *DMA* index, the whole iterative process still takes half a day to complete. Therefore, we only run a few times on *Beijing-L* to ensure the improvement is consistently visible under different settings.

5.6 Summary

In this chapter, we propose an iterative-based map-trajectory co-optimisation algorithm which improves both the trajectory map-matching result and map quality. Following such an idea, we propose a co-optimisation framework which is comprised of the map-matching, map update and our co-optimisation model. We implement the most representative methods for map-matching (HMM) and map update (KDE, TC) for best performance, and we propose and layer a co-optimisation algorithm into the framework to further refine the map-matching and map update results. Through the introduction of map-matching certainty, road influence and confidence, we build an evaluation process for each newly updated road so that incorrectly updated road can be removed and the wrong map-matching results can be remedied. In addition to the original solution, we further improve its efficiency by proposing an index-based trajectory filtering strategy. We conduct our experiments on both public map-matching benchmark dataset and trajectory datasets with different scales. The results show superior performance compared with the state-of-the-art map inference/update solutions and improved efficiency gained from the index-based filtering. Overall, our proposed framework works well in improving the map and map-matching quality. It also shows its flexibility to work with most of the existing map-matching and inference algorithms. However, the experiments also depict that the performance is affected by the number of iterations significantly, and it is quite unstable due to the unstable quality value and the benefit result in each iteration. Therefore, there is still room for improvement in terms of defining the confidence score, influence score, as well as the way of identifying a correct road. With the help of recent machine learning methods which are known for outstanding data classification performance, it is possible that correct road identification can be improved through the learning process.

Chapter 6

Map Service Platform

6.1 Introduction

The ubiquity of GPS trajectory data and digital maps enables various types of location-based applications, including navigation, vehicle tracking, traffic analysis and location-based recommendation. However, since both GPS trajectories and maps are intrinsically inaccurate, various techniques have been proposed to deal with the data quality issues. In particular, the map-matching method aims to find the actual travel path of an object given its inaccurate GPS trajectory, while the map inference/update tries to construct/update a map automatically using the trajectories sampled in the map region. Nowadays, due to the importance of the maps and trajectories, those techniques are widely adopted as the crucial data cleaning/preprocessing step for most of the location-based applications in both academia and industry.

However, to the best of our knowledge, limited sources are available online for those processes. For example, as the essential preprocessing step of most trajectory-based applications, the map-matching tools are rarely seen in most of the open-source graph/spatial/geometric toolboxes. Meanwhile, the existing projects with map-matching embedded, like the Graphhopper [52] and Barefoot [66], only provide a single map-matching algorithm, mostly the classical HMM-based solution proposed in 2009 [90], as the candidate solution. Abundant solutions proposed afterwards are not included in any toolbox, not to mention the lack of support on different working scenarios (online/offline map-matching on low/high-sampling rate data). In contrast, while many map-matching, map inference and map update algorithms have been proposed in recent years, some of them share their implementation publicly for peer review and validation. However, these projects are usually implemented in different languages, provide limited comments and have designated input/output formats. Therefore, they are hard to use for reproduction and comparison.

On the other hand, the recent upsurge of research on map-matching [112, 119], map inference [116, 161] and map update [111, 133] shows a strong need of comparing their own proposals to the previous baseline algorithms. Meanwhile, the frequent use of those data preprocessing procedures in practice requires a toolbox that supports those functionalities on different input formats, scales and working scenarios. Considering the close relationship between map-matching and map inference/update and their broad applications in both academia and industry, we develop a map service platform that supports aforementioned processes, including map-matching, map inference and our proposed map-trajectory co-optimisation, on various working scenarios and different datasets with various algorithms available[1]. Overall, the main features and contributions of our platform are as follows:

- To the best of our knowledge, our proposed platform is the first open-source system that supports both map-matching and map inference/update processes with multiple candidate solutions. The platform enables the data quality improvement solely on GPS trajectories (map-matching), maps (map inference/update) or on both datasets simultaneously (map-trajectory co-optimisation). Besides, the variety of options of different map-matching/map inference solutions ensures the best performance under different data inputs and provides a general platform for easy comparison for future research.

- In addition to the basic functionality, our proposed platform provides abundant toolkits for data conversion, data preprocessing, data visualisation and data quality evaluation for both GPS trajectories and maps. Moreover, we implement various spatial tools, like different spatial indices (R-tree, grid, kd-tree, etc.), different coordinate systems (WGS84 [142], GCJ-02 [141] and Mercator projection [140]) and different shortest-path algorithms (Dijkstra, A*, etc.), for ease-of-use and future research convenience.

- The platform is designed and implemented under clear modularisation, which supports easy plug-ins for multiple programming languages (Java, Python, Go, etc.) and easy extensions. Therefore, future research on map-matching and map inference/update can be introduced and compared with existing solutions easily.

Note that the main objective of our platform is to provide a practical platform for public use rather than achieving research breakthroughs. Therefore, in this chapter, we mainly introduce the core components and the main functionalities of the whole system.

[1]The platform is open-source, link: `https://github.com/Hellisk/map-service`

6.2 Framework Overview

In general, the core functionality of our map service platform is the support of map-matching, map inference and map-trajectory co-optimisation processes on various types of data input and different working scenarios with a wide range of algorithm options. In addition, to ensure that users can easily implement their own algorithms and compare with other in-built candidates on different datasets, the platform also provides functionalities for data cleaning, result evaluation, visualisation and configuration. Overall, as shown in Fig. 6.1, the framework of the platform consists of the following components:

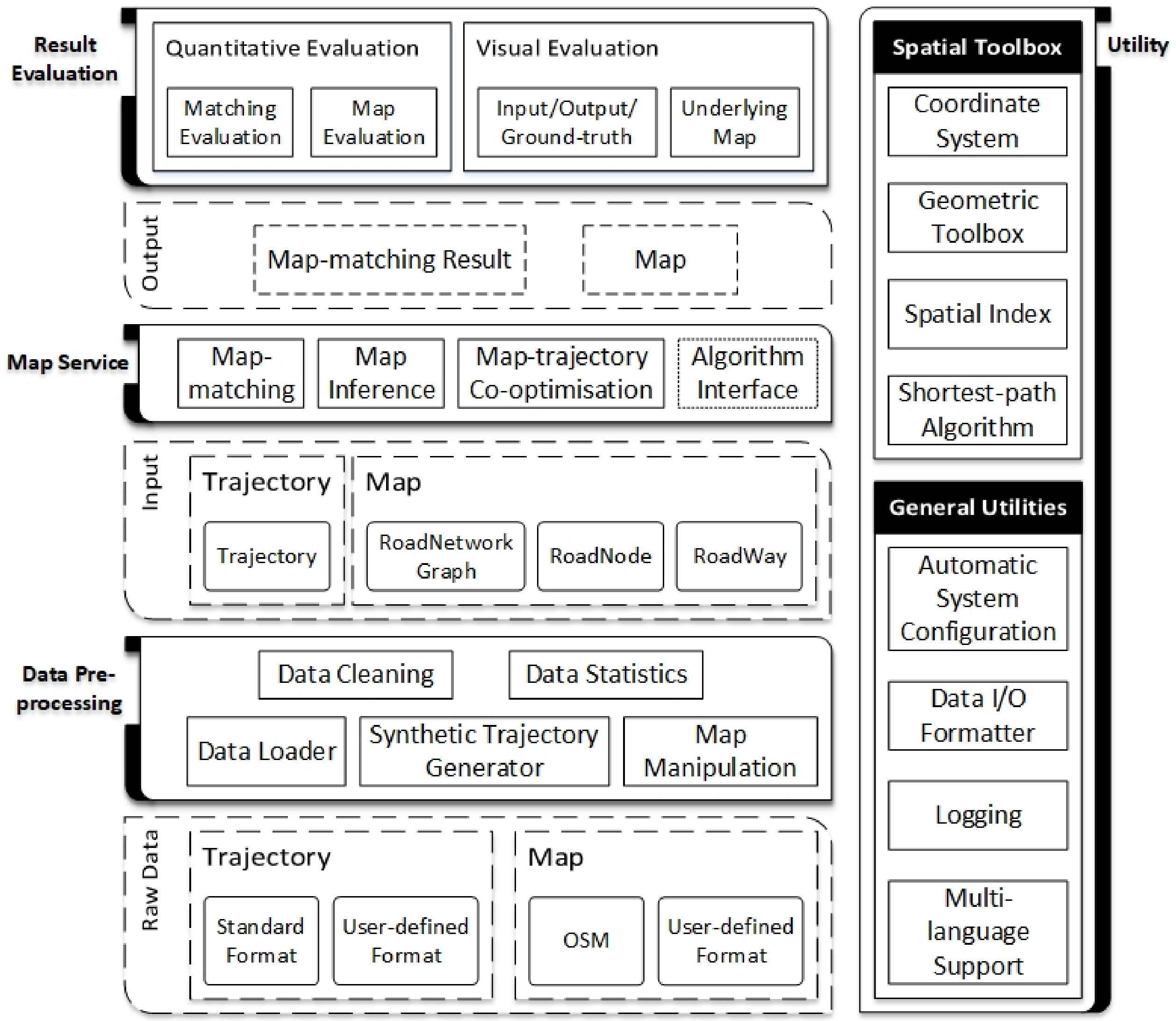


FIGURE 6.1: Framework overview

- **Data Preprocessing**: In data preprocessing, various types of input GPS trajectory/map formats can be parsed into our unified formats. Besides, we provide various data cleaning options, like deduplication and trajectory compression, to the users, which (1) improves the quality of input data or (2) manipulates the data to satisfy certain requirements. Meanwhile, we also provide synthetic dataset generators, including trajectory generator and map manipulator, which can be used for scalability and accuracy tests of the algorithms.

- **Map Service**: The map service is the main component of the platform. There are three main functionalities provided: map-matching, map inference and map-trajectory co-optimisation. Both map-matching and map inference have multiple available algorithms. Moreover, with the formally defined interfaces, they both support future algorithm extensions. While for the map-trajectory co-optimisation, in addition to the default solution proposed in [27], other alternative map-matching and map inference algorithms can also be adopted in the framework.

- **Result Evaluation**: In result evaluation, both the map-matching, map inference and co-optimisation results can be evaluated by various metrics. Besides, we provide a visualisation tool that can display spatial objects, including trajectories, travel paths and road network skeleton, on public map layouts, like OpenStreetMap [94] and Google Maps [51], for visual evaluation.

- **Utilities**: Other than basic definitions of the data formats, most of the supporting functions are defined in this package. In particular, as the crucial components in map-matching and map inference, several spatial indices and shortest path algorithms are provided for different query performance requirements. We also provide the conversion between different coordinate systems and the corresponding distance functions to ensure data from different coordinate systems can all be processed in our system. Lastly, we provide customisable configuration files for map-matching, map inference and co-optimisation, respectively, so that users can easily test various algorithms using different settings without the need for changing the code.

From the user's perspective, input datasets and a configuration file are required for running a preprocessing procedure (data preprocessing, map-matching, map inference or map-trajectory co-optimisation). The running of the procedure includes the following steps:

- First, the system parses the configuration file for arguments, including: (1) which procedure is invoked; (2) the format of the input data and the name of input folder(s) and log file folder; (3) which algorithm is chosen (when map-matching, map inference or map-trajectory co-optimisation procedure is chosen) and (4) the values of the parameters in the chosen algorithm.

- Second, the system reads the input data from a specified folder(s) and starts the corresponding processing procedure.

- Third, the output results are evaluated through the result evaluation process. Data visualisation is provided as an alternative evaluation method.
- Forth, the output results can be found in specified folders with unified formats. Meanwhile, the evaluation results are written as logs in log file.

The platform is mainly implemented in Java-11, with some of the algorithms written in Python-2.7 and Go-1.12.6 (future algorithms written on these languages are compatible). The entire project is built and maintained by Apache Maven [10] for easy deployment.

6.3 Implementation Details

As aforementioned, four main components are included in our framework. In this section, we mainly introduce the implementation details of each component, respectively, which provides a brief understanding and instruction for future users.

6.3.1 Data Preprocessing

The main goal of the data preprocessing is to prepare the input data for the following data processing algorithms. In general, three types of data are required in data processing and evaluation: the input, output and ground-truth data required for evaluation. Considering the map-matching, map inference and map-trajectory co-optimisation mentioned in Chapter 3, 4 and 5, respectively, the required data types include the following:

- **Map-matching**: trajectory (input), road network (input), map-matching result (output/ground-truth).
- **Map inference**: trajectory (input), road network (output/ground-truth).
- **Map-trajectory co-optimisation**: trajectory (input), road network (output/ground-truth), map-matching result (output/ground-truth).

Note that in the co-optimisation process, the unmatched trajectory and top-k map-matching results are generated as intermediate results. However, they can also be represented as a trajectory and a variation of a map-matching result, respectively. Hence, only three types of data are utilised in our system, defined as follows:

- **Trajectory**: A *Trajectory* Tr is stored as an array of *TrajectoryPoint*, each of which contains 2-D coordinates, a timestamp, a speed (optional) and a heading (optional).

- **Road network**: A *RoadNetworkGraph* G consists of a list of *RoadNodes* (vertices) and a list of *RoadWays* (edges). Each *RoadNode* contains 2-D coordinates while each *RoadWay* stores the pointers to starting and ending *RoadNodes* and a list of intermediate vertices. Note that both *RoadNode* and *RoadWay* maintain a hash map that stores additional attributes, like road width (edge), speed limit (edge), intersection type (vertex), etc. Such definition ensures its compatibility to maps from various sources with/without semantic features.

- **Matching result**: A *TrajectoryMatchResult* $\mathcal{M}_G(Tr)$ contains both a list of point match result $\mathcal{MP}_G(Tr)$(*PointMatch* defined in Section 4.3) and a route match result $\mathcal{MR}_G(Tr)$, which can be used for storing both online and/or offline map-matching results and supporting both point-based and route-based evaluation.

The principle of defining data formats is to support more types of input sources and algorithms that require different semantic features. On top of that, the data preprocessing module provides the following functionalities:

- **Data loader**: The input data from various sources can be converted to our defined formats using our data loader. The data loader supports the parse of customisable data formats for both trajectory and road network, as well as some typical formats like OpenStreetMap and shapefile (.shp).

- **Data cleaning and validation**: Since the data quality issues are common in spatial-temporal datasets, we implement some typical data cleaning and validation processes for better input data quality. In particular, the cleaning process mainly focusses on (1) removing duplicated, unreachable or noisy trajectory points, (2) unreachable vertices and duplicated edges in a road network. Besides, we provide functions for validating the correctness of map components in terms of their connectivities.

- **Data generator and modifier**: To support experiments on synthetic datasets, we propose the generator of both trajectories and maps based on real datasets. In addition, we support the manipulation of existing maps and trajectories to simulate different levels of noise in real datasets.

- **Data statistics**: For analytic purposes, we implement a tool that generates the statistics of trajectory and map datasets in terms of the features that may affect their performance in map-matching, map inference and co-optimisation.

6.3.2 Map Services

As the main service in our platform, the map service contains three components: map-matching, map inference and map-trajectory co-optimisation. Since these processes are formally defined in previous chapters, here, we mainly focus on their implementation details. In general, when designing this module, the core principle is to ensure the extendibility of the services so that new solutions can be easily plugged in and compare with existing candidates. Hence, one of the main objective in this module is to define the interfaces of those services in a clear and concise way.

Map inference: According to Section 3.2, despite the map inference algorithms constructing a map in complete different ways, they all take the same type of input ($R =$ a list of *Trajectory*) as well as generate the same output format ($G =$ a *RoadNetworkGraph*). Therefore, the map inference interface is defined as follows:

$$G \leftarrow MapInferenceProcess(R, Pram) \tag{6.1}$$

where the $Pram$ represents a list of parameters that are used in the map inference algorithm. Although the parameters needed in various algorithms are different, they can all be generalised as a list of key-value pairs.

Map-matching: Different from the map inference problem, the solutions for the map-matching problem are categorised according to their working scenarios. In offline map-matching, the input includes a trajectory Tr and the underlying map G, while the output is the map-matching result $\mathcal{M}_G(Tr)$, i.e.:

$$\mathcal{M}_G(Tr) \leftarrow OfflineMapMatchingProcess(Tr, G, Pram) \tag{6.2}$$

However, in online map-matching, the trajectory points arrive in a streaming fashion. Therefore, at each moment, the online map-matching is performed with only the current trajectory point p and the output only contains the point match result, i.e.:

$$\mathcal{MP}_G(Tr) \leftarrow OnlineMapMatchingProcess(p, G, Pram) \tag{6.3}$$

Map-trajectory co-optimisation As mentioned in Section 5.2, the input of the co-optimisation process consists of a trajectory set R and a road network G while the output is a refined map $G*$ and the map-matching results of the trajectory set, i.e. $\mathcal{M}_G(R)$. Therefore, the interface is defined as:

$$(G*, \mathcal{MP}_{G*}(R)) \leftarrow CooptimisationProcess(R, G, Pram) \tag{6.4}$$

Meanwhile, since the co-optimisation framework support the plug-in of various map-matching and map inference algorithms, we further define their interfaces used in the framework:

$$(\mathcal{M}_G(R), UnR) \leftarrow CooptimisationMatchingProcess(R, G, Pram) \tag{6.5}$$

$$G' \leftarrow CooptimisationInferenceProcess(UnR, Pram) \tag{6.6}$$

Here, the UnR is the set of unmatched trajectories generated by map-matching algorithm, and G' is the road network that constructed from UnR. Comparing Eq. 6.6 with Eq. 6.1, we can clearly see the definitions are identical (R and UnR are both defined as a set of trajectories), which means the existing map inference algorithms can easily be adapted in the co-optimisation framework. However, despite that R is a collection of Tr, the Eq. 6.5 is still slightly different from Eq. 6.2 due to the generation of UnR. In fact, it can be achieved by a post-processing step after the map-matching, i.e.:

$$UnR \leftarrow UnmatchedTrajGenProcess(R, \mathcal{M}_G(R), G, Pram) \tag{6.7}$$

Therefore, with the help of the process in Eq. 6.7, the existing offline map-matching algorithms can be applied to the framework as well. Overall, our map service module provides a set of interfaces that support the users to easily adapt their solutions to the current module. Besides, we implemented more than five candidate algorithms for both map-matching and map inference problems for future performance comparison.

6.3.3 Result Evaluation

In the result evaluation, we provide both quantitative evaluation and visual evaluation options. In terms of the quantitative evaluation, we offer four metrics for map quality evaluation and six map-matching quality metrics, which are introduced in Section 3.3 and 4.6, respectively. Regarding the visual comparison, we utilise a java-based visualisation package, termed as UnfoldingMaps [125], and implement our supports to multiple spatial data types. Currently, our visualisation tool offers the following features:

- The underlying map in our visualisation can be configured to Google Maps, OpenStreetMap, satellite images or single colour background;

- The supported data types include trajectories, road networks, map-matching results and the overlay of multiple data types.
- The colour of different data source is customisable. It can also be adaptive to the line weight (e.g.: the roads with more traffic have warmer colour)

6.3.4 Utilities

Since one of the main purposes of our platform is to shorten the time spent on implementation and experiments for future research in these fields, we provide auxiliary tools for easy development, including a spatial toolbox and other general utilities, as shown in Fig. 6.1. The purpose for the spatial toolbox is to help accelerate the implementation of new algorithm on our platform, while the main functionalities of general utilities are to better support more data formats and more languages, and to use the platform more conveniently. In general, our utilities can benefit the implementation from the following aspects:

- In *geometric toolbox*, we define different geometric objects (point, rectangle, circle, etc.) and implement a bunch of spatial operations (intersect, contain, disjoint, etc.). Since all the spatial data formats used in our platform (trajectory, road network, etc.) are built based on these objects, it is quite convenient for designing new algorithms and indices and they can be easily adapted to the existing data formats.
- In *coordinate system*, We implement the distance calculation methods on different coordinate systems, including Euclidean distance and Great Circle distance, and the conversion between different geodetic systems, namely GCJ-02, WGS84 and Universal Transverse Mercator system(UTM). Different from the previous algorithms that only support one type of coordinate system, the algorithm implemented on our platform does not need to bother on calculating distances manually; instead, it can automatically support both coordinate systems and run on different geodetic systems with a simple coordination conversion.
- As the main components in most of the existing map-matching and map inference algorithms, the *shortest-path algorithms* and *spatial indices* can significantly affect the performance of the algorithms. Therefore, we implement multiple shortest path algorithms (e.g.: Dijkstra, A*) and spatial indexing structures (e.g.: Grid, R-tree, KD-tree) that ensure the comparison between different algorithms is not affected by the inconsistent implementation of shortest path algorithms and indices. Besides, it can significantly reduce the cost for algorithm implementation and help the users to find a better index or shortest path algorithm more quickly.

- We develop an ease-of-use configuration module that helps users to easily parse the parameters they define without worrying about its conflict to the settings of other algorithms. In addition, the configuration module helps reproduce experiments and achieve batch experiments easily.

6.4 Case Study

To help users better understand our platform, we provide an example to demonstrate how to use our map services. We take our *Beijing* dataset as an example and perform a map-matching process (map inference and co-optimisation are similar) on Windows (Linux also supported). To complete a map-matching process from scratch, we are required to perform four steps: data preprocessing, map-matching, result evaluation and visualisation.

6.4.1 Data Preprocessing

The input of a map-matching process includes a trajectory dataset and a map. Assuming the input data come from different sources, our first step is to convert them into a uniform trajectory/map format in order to be processed in the subsequent steps. First, we specify a root folder for all map service tasks, for example ”F:/data/”. Therefore, for the Beijing raw dataset, we save them in the folder of ”F:/data/Beijing/raw/”. The trajectory is stored in ”.../raw/trajectory/” and map is stored in ”.../raw/map/”, respectively. The reason for doing so is that our platform maintains a tree-based directory structure automatically, which means all data generated by the platform will go to the corresponding folder according to their data types, making the data navigation much easier.

After raw data are placed in the right folders, we open the preprocessing configuration file at ”src/resource/preprocessing.properties” and modify the settings as follows:

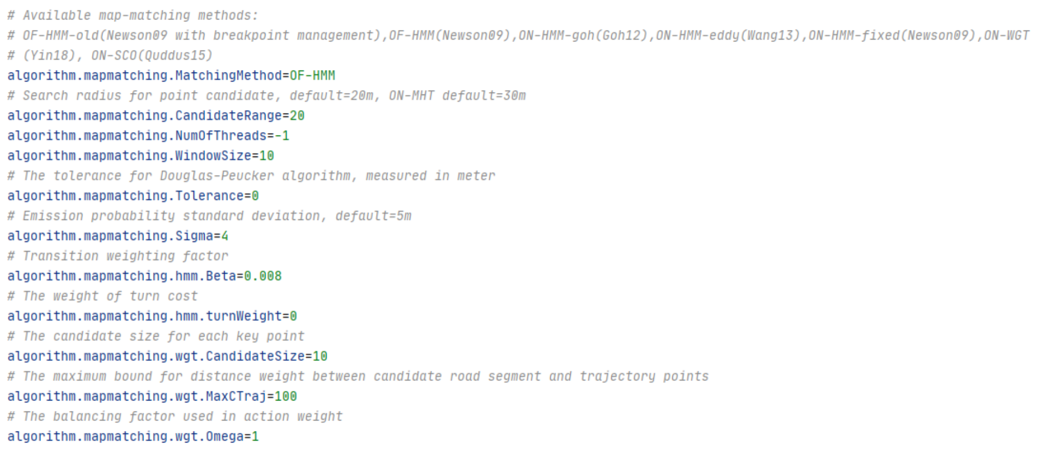
- OS=Win
- data.RootPath=F:/data/
- data.Dataset=Beijing

Meanwhile, we also set other parameters that will be used in data preprocessing, like the map boundary, trajectory sampling rate, trajectory size, trajectory minimum length, etc. Note that the parser of Beijing dataset has been implemented as ”BeijingMapLoader” and ”BeijingTrajectory-Loader” in ”src/main/java/util/io/”, and we also provide the parser of OpenStreetMap maps in ”OS-MMapLoader”. However, for new data types, the data loader should be implemented separately.

Eventually, run the ”ProcessingMain” class in ”src/main/java/preprocessing/” to start the preprocessing, the trajectories and maps after preprocessing will be stored in folder ”F:/data/Beijing/input/”. Both of them follow the uniform formats defined in Section 6.3 regardless of the raw data format.

6.4.2 Map-matching and Evaluation

After the data preparation, we are ready to start the map-matching process by first configuring the ”mapmatching.properties”. Same as in data preprocessing, the first section of the property file (parameters starting with ”data.*”) consists of all data related settings. In the second part, it specifies all parameters related to map-matching algorithms. As shown in Figure 6.2, we can choose the map-matching algorithm by changing the value of ”algorithm.mapmatching.MatchingMethod”. Besides, all parameters are categorised based on which algorithm they belong to, which allows us to better adjust the parameters during multiple runs. Finally, run the ”MapMatchingMain” class in ”src/main/java/algorithm/mapmatching/” to start the matching.



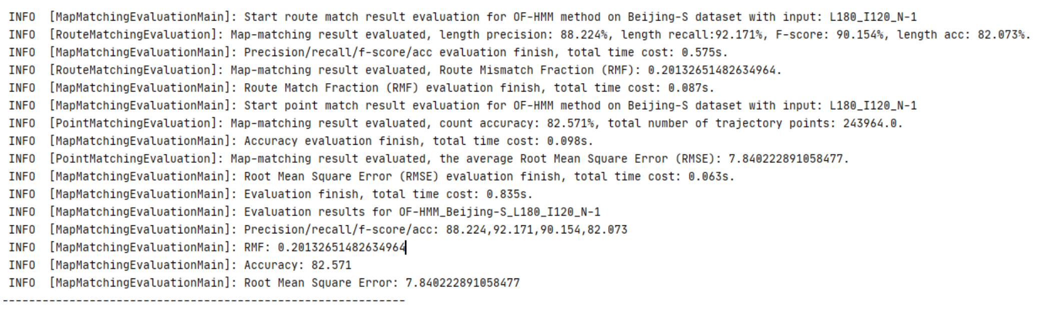
```
# Available map-matching methods:
# OF-HMM-old(Newson09 with breakpoint management),OF-HMM(Newson09),ON-HMM-goh(Goh12),ON-HMM-eddy(Wang13),ON-HMM-fixed(Newson09),ON-WGT
# (Yin18), ON-SCO(Quddus15)
algorithm.mapmatching.MatchingMethod=OF-HMM
# Search radius for point candidate, default=20m, ON-MHT default=30m
algorithm.mapmatching.CandidateRange=20
algorithm.mapmatching.NumOfThreads=-1
algorithm.mapmatching.WindowSize=10
# The tolerance for Douglas-Peucker algorithm, measured in meter
algorithm.mapmatching.Tolerance=0
# Emission probability standard deviation, default=5m
algorithm.mapmatching.Sigma=4
# Transition weighting factor
algorithm.mapmatching.hmm.Beta=0.008
# The weight of turn cost
algorithm.mapmatching.hmm.turnWeight=0
# The candidate size for each key point
algorithm.mapmatching.wgt.CandidateSize=10
# The maximum bound for distance weight between candidate road segment and trajectory points
algorithm.mapmatching.wgt.MaxCTraj=100
# The balancing factor used in action weight
algorithm.mapmatching.wgt.Omega=1
```

FIGURE 6.2: Example of map-matching parameters

After the map-matching is complete, we are able to find the matching results in ”F:/Data/Beijing/output/matchResult/”. In addition, run ”MapMatchingEvaluationMain” in ”src/main/java/evaluation/matchingevaluation/” to obtain the quantitative evaluation results, shown as Figure 6.3. We can also find the logs of map-matching and evaluation from ”F:/Data/Beijing/matching/log/”.

6.4.3 Visual Comparison

We provide the display of major data types in our application. In map-matching application, the most common scenario is to compare the map-matching result with original trajectory or with the ground-truth map-matching result. In our platform, we provide the functions for displaying various data types, but the user has to specify which data to display in ”UnfoldingMapDisplay” in



```
INFO  [MapMatchingEvaluationMain]: Start route match result evaluation for OF-HMM method on Beijing-S dataset with input: L180_I120_N-1
INFO  [RouteMatchingEvaluation]: Map-matching result evaluated, length precision: 88.224%, length recall:92.171%, F-score: 90.154%, length acc: 82.073%.
INFO  [MapMatchingEvaluationMain]: Precision/recall/f-score/acc evaluation finish, total time cost: 0.575s.
INFO  [RouteMatchingEvaluation]: Map-matching result evaluated, Route Mismatch Fraction (RMF): 0.20132651482634964.
INFO  [MapMatchingEvaluationMain]: Route Match Fraction (RMF) evaluation finish, total time cost: 0.087s.
INFO  [MapMatchingEvaluationMain]: Start point match result evaluation for OF-HMM method on Beijing-S dataset with input: L180_I120_N-1
INFO  [PointMatchingEvaluation]: Map-matching result evaluated, count accuracy: 82.571%, total number of trajectory points: 243964.0.
INFO  [MapMatchingEvaluationMain]: Accuracy evaluation finish, total time cost: 0.098s.
INFO  [PointMatchingEvaluation]: Map-matching result evaluated, the average Root Mean Square Error (RMSE): 7.840222891058477.
INFO  [MapMatchingEvaluationMain]: Root Mean Square Error (RMSE) evaluation finish, total time cost: 0.063s.
INFO  [MapMatchingEvaluationMain]: Evaluation finish, total time cost: 0.835s.
INFO  [MapMatchingEvaluationMain]: Evaluation results for OF-HMM_Beijing-S_L180_I120_N-1
INFO  [MapMatchingEvaluationMain]: Precision/recall/f-score/acc: 88.224,92.171,90.154,82.073
INFO  [MapMatchingEvaluationMain]: RMF: 0.20132651482634964
INFO  [MapMatchingEvaluationMain]: Accuracy: 82.571
INFO  [MapMatchingEvaluationMain]: Root Mean Square Error: 7.840222891058477
------------------------------------------------------------
```

FIGURE 6.3: Quantitative evaluation of map-matching

"src/main/java/util/visualization/". To display the map-matching components, we specify the file paths, the line colours and stroke weights in the code, we can also choose the underlying map tiles from various providers(Google, OpenStreetMap, Microsoft, etc.). After running the "Visualization-Main", we can see the display of a map-matching result, shown in Figure 6.4. Here, the map tiles are obtained from Google. but the grey lines covering the map comes from our map data. The red line with dots represents the original trajectory, while the line with light green represent the map-matching output (ground-truth can be displayed in the same way). The map can be zoomed at different scales, or using different map tiles, like Figure 6.5 which shows the same trajectory on satellite image with another scale.

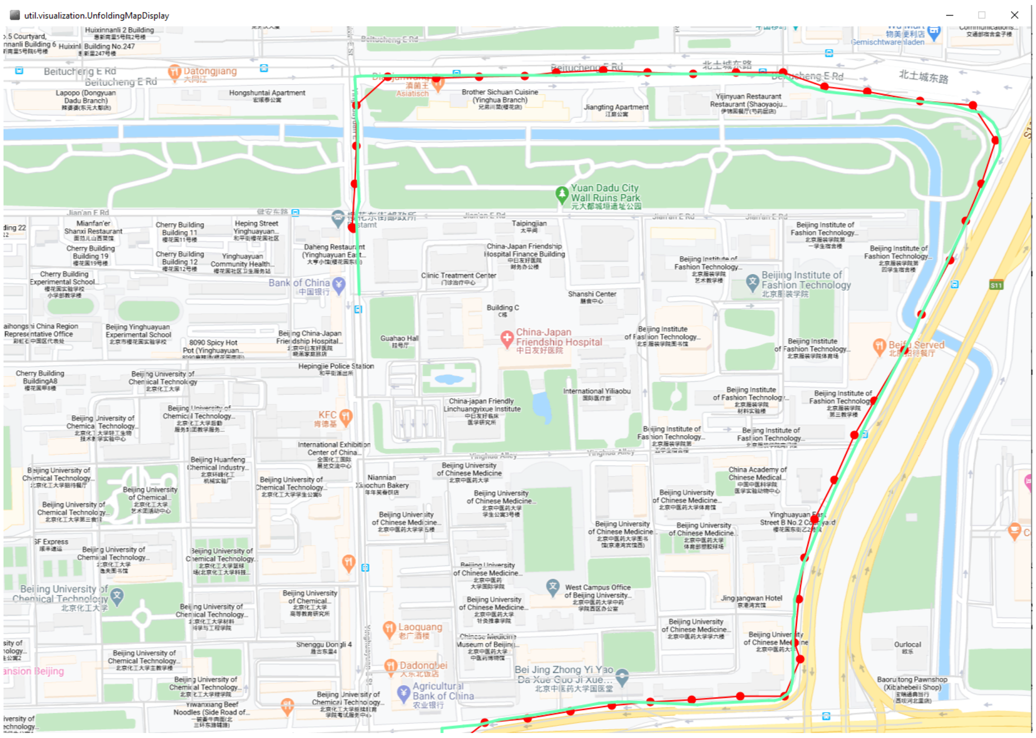

util.visualization.UnfoldingMapDisplay
Beitucheng E Rd
北土城东路
Yuan Dadu City Wall Ruins Park
元大都城垣遗址公园
Jian'an E Rd
China-Japan Friendship Hospital
中日友好医院
Beijing Institute of Fashion Technology
北京服装学院
Bank of China
Beijing University of Chemical Technology
北京化工大学
Bei Jing Zhong Yi Yao Da Xue Guo Ji Xue
Agricultural Bank of China
Dadongbei
Laoguang
KFC

FIGURE 6.4: Map-matching result visualisation on Google Maps

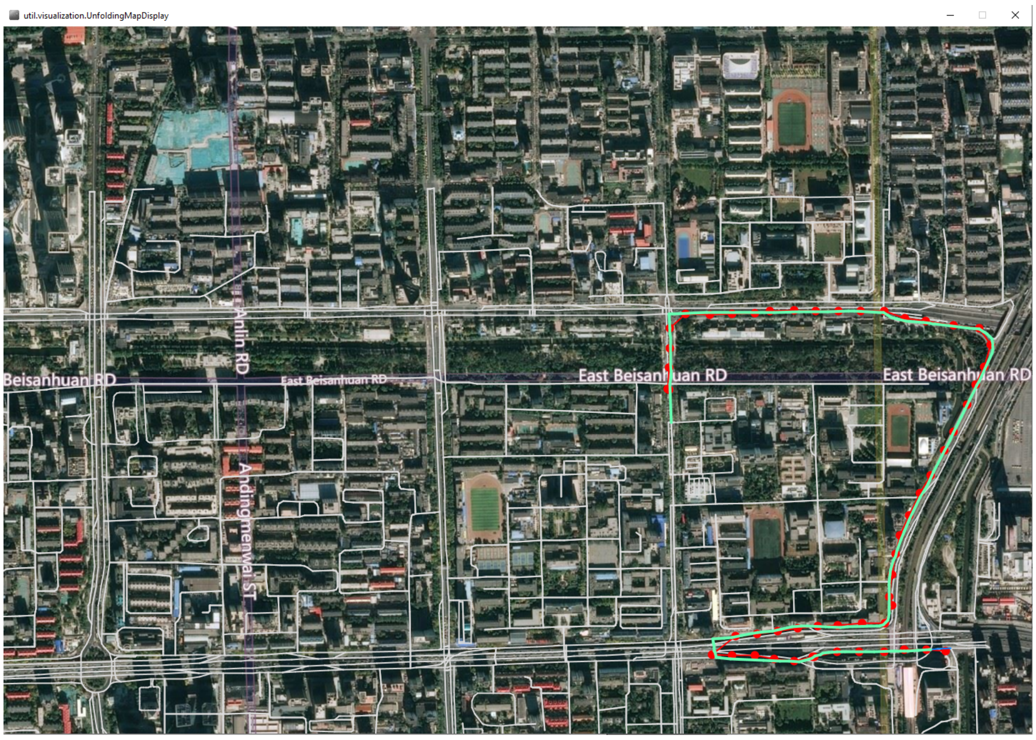


FIGURE 6.5: Map-matching result on Microsoft satellite image

6.5 Conclusion

In this chapter, we introduce the main features of our map service platform. In general, our platform is designed for two main purposes: (1) help the future research in map-matching, map inference and map update in terms of faster implementation and easier comparison, (2) provide an easy-to-use platform for data cleaning/data preprocessing processes on trajectories and maps. By introducing the framework and main components of our platform, we demonstrate our contributions in helping the users to implement their algorithms more efficiently, providing a wide range of candidate algorithms and evaluation methods for easy comparison and designing ease-of-use configuration strategies for faster deployment and batch experiments.

Chapter 7

Final Remarks

7.1 Conclusions

In this thesis, we present our research on data quality issues in GPS trajectories and road maps. In particular, we conduct extensive studies on the map-matching and map inference problem and propose our map-trajectory framework that utilises the map-matching and map inference solutions and solves the data quality on both datasets simultaneously. In addition, we develop and release a map service platform which provides abundant tools for data quality improvement on GPS trajectories and maps and includes all aforementioned functionalities.

In Chapter 3, we present a comprehensive survey and experimental study of existing map inference algorithms. Specifically, we propose a new categorisation method, compare the representative algorithms experimentally and evaluate the existing quantitative measures. Besides, to test the robustness of algorithms and quantitative measures, we introduce a synthetic trajectory generator and an artificial map generator to simulate different trajectory errors and map quality issues, respectively. According to our experiments, we observe that besides their respective weakness, the existing quantitative measures are unable to identify several map issues and do not consider road importance. Regarding the candidate algorithms, the existing algorithms struggle to guarantee performance when the GPS errors exceed their expected threshold, and they still find a hard time identifying roads that are rarely travelled. Moreover, more input trajectories do not always lead to better inference results. Overall, we identify the method that has the best scalability (RA-K-MEANS), the best accuracy (RA-TOPIC), and the best suitability for map update (IB-ME), respectively, and also point out potential future research directions.

In Chapter 4, we enumerate and categorise the existing map-matching algorithms according to their map-matching model, working scenarios and input data features. We discuss the main strengths and weaknesses of each category and introduce their representative algorithms. Moreover, we introduce several typical tuning techniques used in recent map-matching solutions, which involve new research fields (machine learning) and new data types (Bluetooth, DGPS, etc.). On the other hand, we conduct comprehensive experiments on multiple algorithms, metrics and datasets to compare the existing map-matching solutions. The experiment results show the following: (1) the accuracy of online map-matching benefits a lot from delayed matching, and the latency can be wisely controlled without hurting performance; (2) it is not always true that higher sampling rates lead to better matching performance. Down-sampling trajectories is beneficial when the sampling rate is too high, but a simple trajectory compression strategy cannot serve this purpose. (3) the HMM-based methods can still achieve overall better performance, while a simple scoring method can be very efficient in online scenarios without losing too much accuracy if the data quality is decent; (4) map density is a crucial factor affecting both the efficiency and matching accuracy of the algorithms, which is the main challenge in the future. Overall, this chapter summarises and compares the existing map-matching solutions and provides insightful observations and guidance for future research.

In Chapter 5, we propose an iterative-based map-trajectory co-optimisation algorithm which improves both the trajectory map-matching result and map quality. Following such an idea, we propose a co-optimisation framework, which is comprised of the map-matching, map update and our co-optimisation model. We implement the most representative methods for map-matching (HMM) and map update (KDE, TC) for best performance, and we propose a co-optimisation algorithm upon them to further refine the map-matching and map update results. Through the introduction of map-matching certainty, road influence and confidence, we build an evaluation process for each newly updated roads so that incorrectly updated road can be removed and the wrong map-matching results can be remedied. In addition to the original solution, we further improve its efficiency by proposing an index-based trajectory filtering strategy. We conduct our experiments on both public map-matching benchmark datasets and trajectory datasets with different scales. The results show the superior performance compared with the state-of-the-art map inference/update solutions and the improved efficiency gained from the index-based filtering. Overall, our proposed framework works well in improving the map and map-matching quality. It also shows its flexibility to work with most of the existing map-matching and inference algorithms.

In Chapter 6, we introduce the core functionality and main features of our map-service platform. As an open-source project, our platform supports various data preprocessing, including map-matching, map inference and map-trajectory co-optimisation, for GPS trajectory data and map data. In addition, we provide abundant data cleaning and evaluation tools that can simplify future research implementation and comparison using our platform. Overall, our platform is beneficial to two types of users: (1) it helps future researchers working in map-matching, map inference and map update to implement and compare their solutions more easily and conveniently; and (2) it provide an easy-to-use platform for developers to preprocess and evaluate their trajectory and map data.

7.2 Directions for Future Work

In the future, we plan to continue our work on data quality issues in maps and trajectories. Specifically, there are two potential directions in which we are planning to work.

7.2.1 Trajectory Reconstruction on Low-Quality AVI Data

An Automated Vehicle Identification (AVI) system is comprised of a set of sensors installed at certain locations in the road network that reports the occurrence of a vehicle/pedestrian once observed. Currently, various types of sensors are used for vehicle detection, like Bluetooth scanners [15], Wi-Fi and cameras. Different from the GPS positioning, the observations of a vehicle only occur around the detectors' location. Since most sensors are installed sparsely around a map region, the observation frequency of the same vehicle is extremely low, making it challenging to speculate the actual travel trajectory given its observations, which is called trajectory reconstruction problem [88]. In fact, this problem is closely related to the map-matching problem we studied in our thesis. The major differences and the main challenges are as follows:

- The sampling rate of AVI data is much lower than a GPS trajectory, and the sampling is done aperiodically due to the geographical feature of the sensors.
- There are some unique features in the AVI system that do not appear in GPS trajectory, like the missing detection and overlapping detection zones [16]. Those features should be modelled properly when adapting the map-matching solutions on trajectory reconstruction problem.

According to our study, several map-matching algorithms are deemed to fit the trajectory reconstruction problem, which consists of simple shortest path solutions, HMM map-matching solution and weighted graph-based matching solutions. Recently, we are working on adapting those solutions with the consideration of the aforementioned features in trajectory reconstruction.

7.2.2 Pattern Learning for Map Intersection Inference

As shown in our study of map inference problem, the current map inference quality is still far from being satisfactory even with the state-of-the-art inference method. One of the main reason is the failure of inferring intersections correctly. Due to the low sampling rate and inaccurate positioning, the trajectory usually cannot represent the vehicle's actual path precisely especially when the vehicle makes abrupt turns around intersections. Therefore, the existing map inference algorithms strictly eliminate noisy data around intersections and simplify the problem by representing the intersection as simple structures, like crossroads or T-junctions. However, the current road network is much more complicated in terms of the design of intersections. Not only does the number of entrances for an intersection vary from three to five, six, but also the connectivity between entrances are completely different. Considering that the real-world intersections have some fixed design patterns, given the existing intersection patterns and the trajectory data passing through each pattern, we are able to train a classifier using machine learning techniques to identify an intersection pattern given a set of trajectories. The main challenges of this work are as follows:

- To better differentiate the intersection patterns, the trajectory used for both training and testing should be of high quality. Therefore, the data cleaning process is crucial for the performance of the classifier. However, given that the input data quality is fixed, finding the noisy trajectories that may affect the performance is a challenging problem.

- The choice of the classifier model is another challenging task as there is no existing experience in classifying trajectory data for such purpose. Besides, the data structure of the input trajectories should be chosen wisely. Since the major differences between the intersection patterns are their connectivities and the number/location of the entrances, simply regarding the input trajectories as a set of points will result in lost connectivity features, which is not good for classification.

References

[1] G. Agamennoni, J. I. Nieto, and E. M. Nebot. Robust inference of principal road paths for intelligent transportation systems. *IEEE Transactions on Intelligent Transportation Systems*, 12(1):298–308, 2011.

[2] M. Ahmed, B. T. Fasy, M. Gibson, and C. Wenk. Choosing thresholds for density-based map construction algorithms. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 24. ACM, 2015.

[3] M. Ahmed, B. T. Fasy, K. S. Hickmann, and C. Wenk. A path-based distance for street map comparison. *ACM Transactions on Spatial Algorithms and Systems*, 1(1):3, 2015.

[4] M. Ahmed, S. Karagiorgou, D. Pfoser, and C. Wenk. A comparison and evaluation of map construction algorithms using vehicle tracking data. *GeoInformatica*, 19(3):601–632, 2015.

[5] M. Ahmed, S. Karagiorgou, D. Pfoser, and C. Wenk. Map construction algorithms. In *Map Construction Algorithms*, pages 1–14. Springer, 2015.

[6] M. Ahmed and C. Wenk. Constructing street networks from gps trajectories. In *European Symposium on Algorithms*, pages 60–71. Springer, 2012.

[7] M. Ali, J. Krumm, T. Rautman, and A. Teredesai. Acm sigspatial gis cup 2012. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, pages 597–600. ACM, 2012.

[8] H. Aly and M. Youssef. semmatch: Road semantics-based accurate map matching for challenging positioning data. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 5. ACM, 2015.

[9] R. Andersen, C. Borgs, J. Chayes, J. Hopcraft, V. S. Mirrokni, and S.-H. Teng. Local computation of pagerank contributions. In *International Workshop on Algorithms and Models for the Web-Graph*, pages 150–165. Springer, 2007.

[10] Apache. Apache maven project. `https://maven.apache.org/`. Accessed: 2019-11-01.

[11] C. Ascher, C. Kessler, M. Wankerl, and G. Trommer. Dual imu indoor navigation with particle filter based map-matching on a smartphone. In *2010 International Conference on Indoor Positioning and Indoor Navigation*, pages 1–5. IEEE, 2010.

[12] M. M. Atia, A. R. Hilal, C. Stellings, E. Hartwell, J. Toonstra, W. B. Miners, and O. A. Basir. A low-cost lane-determination system using gnss/imu fusion and hmm-based multistage map matching. *IEEE Transactions on Intelligent Transportation Systems*, 18(11):3027–3037, 2017.

[13] Y. Bang, J. Kim, and K. Yu. An improved map-matching technique based on the fréchet distance approach for pedestrian navigation services. *Sensors*, 16(10):1768, 2016.

[14] D. Bernstein, A. Kornhauser, et al. An introduction to map matching for personal navigation assistants. 1996.

[15] A. Bhaskar and E. Chung. Fundamental understanding on the use of bluetooth scanner as a complementary transport data. *Transportation Research Part C: Emerging Technologies*, 37:42–72, 2013.

[16] A. Bhaskar, M. Qu, and E. Chung. Bluetooth vehicle trajectory by fusing bluetooth and loops: Motorway travel time statistics. *IEEE Transactions on Intelligent Transportation Systems*, 16(1):113–122, 2014.

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

[18] J. Biagioni and J. Eriksson. Map inference in the face of noise and disparity. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, pages 79–88. ACM, 2012.

[19] U. Blanke, R. Guldener, S. Feese, and G. Tröster. Crowdsourced pedestrian map construction for short-term city-scale events. In *Proceedings of the First International Conference on IoT in Urban Space*, pages 25–31. ICST (Institute for Computer Sciences, Social-Informatics and . . . , 2014.

[20] C. A. Blazquez, J. Ries, and P. A. Miranda. Towards a parameter tuning approach for a map-matching algorithm. In *2017 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, pages 85–90. IEEE, 2017.

[21] C. A. Blazquez and A. P. Vonderohe. Simple map-matching algorithm applied to intelligent winter maintenance vehicle data. *Transportation Research Record*, 1935(1):68–76, 2005.

[22] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[23] P. Bonnifait, J. Laneurit, C. Fouque, and G. Dherbomez. Multi-hypothesis map-matching using particle filtering. In *16th World Congress for ITS Systems and Services*, pages 1–8, 2009.

[24] K. Buchin, M. Buchin, D. Duran, B. T. Fasy, R. Jacobs, V. Sacristan, R. I. Silveira, F. Staals, and C. Wenk. Clustering trajectories for map construction. In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 14. ACM, 2017.

[25] 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*, pages 3–12. ACM, 2009.

[26] P. Chao, W. Hua, and X. Zhou. An iterative map-trajectory co-optimisation framework based on map-matching and map update. In *International Conference on Database Systems for Advanced Applications*. Springer, 2019.

[27] P. Chao, W. Hua, and X. Zhou. Trajectories know where map is wrong: an iterative framework for map-trajectory co-optimisation. *World Wide Web*, pages 1–27, 2019.

[28] P. Chao, Y. Xu, W. Hua, and X. Zhou. A survey on map-matching algorithms, 2019.

[29] C. Chen and Y. Cheng. Roads digital map generation with multi-track gps data. In *2008 International Workshop on Education Technology and Training & 2008 International Workshop on Geoscience and Remote Sensing*, volume 1, pages 508–511. IEEE, 2008.

[30] C. Chen, C. Lu, Q. Huang, Q. Yang, D. Gunopulos, and L. Guibas. City-scale map creation and updating using gps collections. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1465–1474. ACM, 2016.

[31] G. Cheng, Y. Wang, S. Xu, H. Wang, S. Xiang, and C. Pan. Automatic road detection and centerline extraction via cascaded end-to-end convolutional neural network. *IEEE Transactions on Geoscience and Remote Sensing*, 55(6):3322–3337, 2017.

[32] O. Cheong, J. Gudmundsson, H.-S. Kim, D. Schymura, and F. Stehn. Measuring the similarity of geometric graphs. In *International Symposium on Experimental Algorithms*, pages 101–112. Springer, 2009.

[33] D. Cui, J. Xue, and N. Zheng. Real-time global localization of robotic cars in lane level via lane marking detection and shape registration. *IEEE Transactions on Intelligent Transportation Systems*, 17(4):1039–1050, 2015.

[34] J. Dai, B. Yang, C. Guo, C. S. Jensen, and J. Hu. Path cost distribution estimation using trajectory data. *Proceedings of the VLDB Endowment*, 10(3):85–96, 2016.

[35] P. Dankelmann. The diameter of directed graphs. *Journal of Combinatorial Theory, Series B*, 94(1):183–186, 2005.

[36] T.-S. Dao, K. K. Leung, C. M. Clark, and J. P. Huissoon. Co-operative lane-level positioning using markov localization. In *2006 IEEE Intelligent Transportation Systems Conference*, pages 1006–1011. IEEE, 2006.

[37] J. J. Davies, A. R. Beresford, and A. Hopper. Scalable, distributed, real-time map generation. *IEEE Pervasive Computing*, 5(4):47–54, 2006.

[38] T. K. Dey, J. Wang, and Y. Wang. Graph reconstruction by discrete morse theory. *arXiv preprint arXiv:1803.05093*, 2018.

[39] O. H. Dørum. Deriving double-digitized road network geometry from probe data. In *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 15. ACM, 2017.

[40] N. M. Drawil, H. M. Amar, and O. A. Basir. Gps localization accuracy classification: A context-based approach. *IEEE Transactions on Intelligent Transportation Systems*, 14(1):262–273, 2012.

[41] J. Du and M. J. Barth. Next-generation automated vehicle location systems: Positioning at the lane level. *IEEE Transactions on Intelligent Transportation Systems*, 9(1):48–57, 2008.

[42] D. Duran, V. Sacristán, and R. I. Silveira. Map construction algorithms: an evaluation through hiking data. In *Proceedings of the 5th ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems*, pages 74–83. ACM, 2016.

[43] S. Edelkamp and S. Schrödl. Route planning and map inference with global positioning traces. In *Computer science in perspective*, pages 128–151. Springer, 2003.

[44] W. Elleuch, A. Wali, and A. M. Alimi. An investigation of parallel road map inference from big gps traces data. *Procedia Computer Science*, 53:131–140, 2015.

[45] F. Emmert-Streib, M. Dehmer, and Y. Shi. Fifty years of graph matching, network alignment and network comparison. *Information Sciences*, 346:180–197, 2016.

[46] D. Eppstein and M. T. Goodrich. Studying (non-planar) road networks through an algorithmic lens. In *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*, page 16. ACM, 2008.

[47] S. Fang and R. Zimmermann. Enacq: energy-efficient gps trajectory data acquisition based on improved map matching. In *Proceedings of the 19th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 221–230. ACM, 2011.

[48] A. Fathi and J. Krumm. Detecting road intersections from gps traces. In *International Conference on Geographic Information Science*, pages 56–69. Springer, 2010.

[49] T. FENG and H. J. TIMMERMANS. Map matching of gps data with bayesian belief networks. *Journal of the Eastern Asia Society for Transportation Studies*, 10:100–112, 2013.

[50] C. Y. Goh, J. Dauwels, N. Mitrovic, M. T. Asif, A. Oran, and P. Jaillet. Online map-matching based on hidden markov model for real-time traffic sensing applications. In *2012 15th International IEEE Conference on Intelligent Transportation Systems*, pages 776–781. IEEE, 2012.

[51] Google. Google maps. `https://www.google.com.au/maps`. Accessed: 2019-11-01.

[52] GraphHopper. Graphhopper. `https://www.graphhopper.com/`. Accessed: 2019-11-01.

[53] J. S. Greenfeld. Matching gps observations to locations on a digital map. In *81th annual meeting of the transportation research board*, volume 1, pages 164–173. Washington, DC, 2002.

[54] T. Guo, K. Iwamura, and M. Koga. Towards high accuracy road maps generation from massive gps traces data. In *Geoscience and Remote Sensing Symposium, 2007. IGARSS 2007. IEEE International*, pages 667–670. IEEE, 2007.

[55] M. Hashemi. A testbed for evaluating network construction algorithms from gps traces. *Computers, Environment and Urban Systems*, 66:96–109, 2017.

[56] M. Hashemi and H. A. Karimi. A critical review of real-time map-matching algorithms: Current issues and future directions. *Computers, Environment and Urban Systems*, 48:153–165, 2014.

[57] M. Hashemi and H. A. Karimi. A machine learning approach to improve the accuracy of gps-based map-matching algorithms. In *2016 IEEE 17th International Conference on Information Reuse and Integration (IRI)*, pages 77–86. IEEE, 2016.

[58] M. Hashemi and H. A. Karimi. A weight-based map-matching algorithm for vehicle navigation in complex urban networks. *Journal of Intelligent Transportation Systems*, 20(6):573–590, 2016.

[59] S. He, F. Bastani, S. Abbar, M. Alizadeh, H. Balakrishnan, S. Chawla, and S. Madden. Roadrunner: improving the precision of road network inference from gps trajectories. In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 3–12. ACM, 2018.

[60] Z.-c. He, S. Xi-Wei, L.-j. Zhuang, and P.-l. Nie. On-line map-matching framework for floating car data with low sampling rate in urban road networks. *IET Intelligent Transport Systems*, 7(4):404–414, 2013.

[61] T. Hofmann. Probabilistic latent semantic indexing. In *ACM SIGIR Forum*, volume 51, pages 211–218. ACM, 2017.

[62] D. Hopper. 7 times google maps straight up ruined people's lives. http://www.cracked.com/article_25510_7-times-google-maps-straight-up-ruined-peoples-lives.html, 2018.

[63] G. Hu, J. Shao, F. Liu, Y. Wang, and H. T. Shen. If-matching: Towards accurate map-matching with information fusion. *IEEE Transactions on Knowledge and Data Engineering*, 29(1):114–127, 2017.

[64] Y. Huang, W. Rao, Z. Zhang, P. Zhao, M. Yuan, and J. Zeng. Frequent pattern-based map-matching on low sampling rate trajectories. In *2018 19th IEEE International Conference on Mobile Data Management (MDM)*, pages 266–273. IEEE, 2018.

[65] T. Hunter, P. Abbeel, and A. Bayen. The path inference filter: model-based low-latency map matching of probe vehicle data. *IEEE Transactions on Intelligent Transportation Systems*, 15(2):507–529, 2014.

[66] B. C. IT. Barefoot. `https://github.com/bmwcarit/barefoot`. Accessed: 2019-11-01.

[67] G. R. Jagadeesh and T. Srikanthan. Robust real-time route inference from sparse vehicle position data. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 296–301. IEEE, 2014.

[68] E. Kaplan and C. Hegarty. *Understanding GPS: principles and applications*. Artech house, 2005.

[69] 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*, pages 89–98. ACM, 2012.

[70] S. Karagiorgou, D. Pfoser, and D. Skoutas. A layered approach for more robust generation of road network maps from vehicle tracking data. *ACM Transactions on Spatial Algorithms and Systems (TSAS)*, 3(1):3, 2017.

[71] D. Kim, B. Kim, T. Chung, and K. Yi. Lane-level localization using an avm camera for an automated driving vehicle in urban environments. *IEEE/ASME Transactions on Mechatronics*, 22(1):280–290, 2016.

[72] M. Kubička, A. Cela, P. Moulin, H. Mounier, and S.-I. Niculescu. Dataset for testing and training of map-matching algorithms. In *Intelligent Vehicles Symposium (IV), 2015 IEEE*, pages 1088–1093. IEEE, 2015.

[73] M. Kubička, A. Cela, H. Mounier, and S.-I. Niculescu. On designing robust real-time map-matching algorithms. In *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pages 464–470. IEEE, 2014.

[74] M. Kubicka, A. Cela, H. Mounier, and S.-I. Niculescu. Comparative study and application-oriented classification of vehicular map-matching methods. *IEEE Intelligent Transportation Systems Magazine*, 10(2):150–166, 2018.

[75] P. Lamb and S. Thiébaux. Avoiding explicit map-matching in vehicle location. In *6th World Conference on Intelligent Transportation Systems (ITS-99)*, 1999.

[76] A. Leick, L. Rapoport, and D. Tatarnikov. *GPS satellite surveying*. John Wiley & Sons, 2015.

[77] S. T. Leutenegger, M. A. Lopez, and J. Edgington. Str: A simple and efficient algorithm for r-tree packing. In *Data Engineering, 1997. Proceedings. 13th international conference on*, pages 497–506. IEEE, 1997.

[78] H. Li, L. Kulik, and K. Ramamohanarao. Automatic generation and validation of road maps from gps trajectory data sets. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, pages 1523–1532. ACM, 2016.

[79] J. Li, Q. Qin, C. Xie, and Y. Zhao. Integrated use of spatial and semantic relationships for extracting road networks from floating car data. *International Journal of Applied Earth Observation and Geoinformation*, 19:238–247, 2012.

[80] L. Li, M. Quddus, and L. Zhao. High accuracy tightly-coupled integrity monitoring algorithm for map-matching. *Transportation Research Part C: Emerging Technologies*, 36:13–26, 2013.

[81] Q. Li, L. Chen, M. Li, S.-L. Shaw, and A. Nüchter. A sensor-fusion drivable-region and lane-detection system for autonomous vehicle navigation in challenging road scenarios. *IEEE Transactions on Vehicular Technology*, 63(2):540–555, 2013.

[82] X. Liu, J. Biagioni, J. Eriksson, Y. Wang, G. Forman, and Y. Zhu. Mining large-scale, sparse gps traces for map inference: comparison of approaches. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 669–677. ACM, 2012.

[83] X. Liu, Y. Zhu, Y. Wang, G. Forman, L. M. Ni, Y. Fang, and M. Li. Road recognition using coarse-grained vehicular traces. *Hp Labs*, 2012.

[84] Y. Lou, C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang. Map-matching for low-sampling-rate gps trajectories. In *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*, pages 352–361. ACM, 2009.

[85] F. Marchal, J. Hackney, and K. W. Axhausen. Efficient map matching of large global positioning system data sets: Tests on speed-monitoring experiment in zürich. *Transportation Research Record*, 1935(1):93–100, 2005.

[86] R. Matthaei, G. Bagschik, and M. Maurer. Map-relative localization in lane-level maps for adas and autonomous driving. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pages 49–55. IEEE, 2014.

[87] J. G. McNeff. The global positioning system. *IEEE Transactions on Microwave theory and techniques*, 50(3):645–652, 2002.

[88] G. Michau, A. Nantes, A. Bhaskar, E. Chung, P. Abry, and P. Borgnat. Bluetooth data in an urban context: Retrieving vehicle trajectories. *IEEE Transactions on Intelligent Transportation Systems*, 18(9):2377–2386, 2017.

[89] R. Mohamed, H. Aly, and M. Youssef. Accurate and efficient map matching for challenging environments. In *Proceedings of the 22nd ACM SIGSPATIAL international conference on advances in geographic information systems*, pages 401–404. ACM, 2014.

[90] P. Newson and J. Krumm. Hidden markov map matching through noise and sparseness. In *Proceedings of the 17th ACM SIGSPATIAL international conference on advances in geographic information systems*, pages 336–343. ACM, 2009.

[91] B. Niehoefer, 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 *2009 First International Conference on Advances in Satellite and Space Communications*, pages 156–161. IEEE, 2009.

[92] W. Y. Ochieng, M. A. Quddus, and R. B. Noland. Map-matching in complex urban road networks. 2003.

[93] OpenCycleMap. Opencyclemap. `https://wiki.openstreetmap.org/wiki/OpenCycleMap`. Accessed: 2019-11-01.

[94] OpenStreetMap. Openstreetmap. `https://www.openstreetmap.org/`. Accessed: 2019-11-01.

[95] T. Osogami and R. Raymond. Map matching with inverse reinforcement learning. In *Twenty-Third International Joint Conference on Artificial Intelligence*, 2013.

[96] F. Peyret, D. Bétaille, J. Laneurit, and R. Toledo-Moreo. Lane-level positioning for cooperative systems using egnos and enhanced digital maps. In *Proc. ENC GNSS*, pages 1–9, 2008.

[97] D. Pfoser and C. S. Jensen. Capturing the uncertainty of moving-object representations. In *International Symposium on Spatial Databases*, pages 111–131. Springer, 1999.

[98] O. Pink and B. Hummel. A statistical approach to map matching using road network geometry, topology and vehicular motion constraints. In *2008 11th International IEEE Conference on Intelligent Transportation Systems*, pages 862–867. IEEE, 2008.

[99] J.-S. Pyo, D.-H. Shin, and T.-K. Sung. Development of a map matching method using the multiple hypothesis technique. In *ITSC 2001. 2001 IEEE Intelligent Transportation Systems. Proceedings (Cat. No. 01TH8585)*, pages 23–27. IEEE, 2001.

[100] J. Qiu and R. Wang. Inferring road maps from sparsely sampled gps traces. *Journal of Location Based Services*, 10(2):111–124, 2016.

[101] M. Quddus and S. Washington. Shortest path and vehicle trajectory aided map-matching for low frequency gps data. *Transportation Research Part C: Emerging Technologies*, 55:328–339, 2015.

[102] M. A. Quddus, R. B. Noland, and W. Y. Ochieng. A high accuracy fuzzy logic based map matching algorithm for road transport. *Journal of Intelligent Transportation Systems*, 10(3):103–115, 2006.

[103] M. A. Quddus, W. Y. Ochieng, and R. B. Noland. Current map-matching algorithms for transport applications: State-of-the art and future research directions. *Transportation research part c: Emerging technologies*, 15(5):312–328, 2007.

[104] M. A. Quddus, W. Y. Ochieng, L. Zhao, and R. B. Noland. A general map matching algorithm for transport telematics applications. *GPS solutions*, 7(3):157–167, 2003.

[105] M. Rahmani and H. N. Koutsopoulos. Path inference from sparse floating car data for urban networks. *Transportation Research Part C: Emerging Technologies*, 30:41–54, 2013.

[106] E. Rappos, S. Robert, and P. Cudré-Mauroux. A force-directed approach for offline gps trajectory map matching. In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 319–328. ACM, 2018.

[107] R. Raymond, T. Morimura, T. Osogami, and N. Hirosue. Map matching with hidden markov model on sampled road network. In *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, pages 2242–2245. IEEE, 2012.

[108] M. Ren. *Advanced map matching technologies and techniques for pedestrian/wheelchair navigation*. PhD thesis, University of Pittsburgh, 2012.

[109] N. Schuessler and K. W. Axhausen. Map-matching of gps traces on high-resolution navigation networks using the multiple hypothesis technique (mht). *Arbeitsberichte Verkehrsund Raumplanung*, 568:1–22, 2009.

[110] J. Schweizer, S. Bernardi, and F. Rupi. Map-matching algorithm applied to bicycle global positioning system traces in bologna. *IET Intelligent Transport Systems*, 10(4):244–250, 2016.

[111] Z. Shan, H. Wu, W. Sun, and B. Zheng. Cobweb: a robust map update system using gps trajectories. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing*, pages 927–937. ACM, 2015.

[112] M. Sharath, N. R. Velaga, and M. A. Quddus. A dynamic two-dimensional (d2d) weight-based map-matching algorithm. *Transportation Research Part C: Emerging Technologies*, 98:409–432, 2019.

[113] W. Shi, S. Shen, and Y. Liu. Automatic generation of road network map from massive gps, vehicle trajectories. In *2009 12th International IEEE Conference on Intelligent Transportation Systems*, pages 1–6. IEEE, 2009.

[114] J. Singh, S. Singh, S. Singh, and H. Singh. Evaluating the performance of map matching algorithms for navigation systems: an empirical study. *Spatial Information Research*, pages 1–12, 2018.

[115] R. Song, W. Lu, W. Sun, Y. Huang, and C. Chen. Quick map matching using multi-core cpus. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, pages 605–608. ACM, 2012.

[116] R. Stanojevic, S. Abbar, S. Thirumuruganathan, S. Chawla, F. Filali, and A. Aleimat. Robust road map inference through network alignment of trajectories. In *Proceedings of the 2018 SIAM International Conference on Data Mining*, pages 135–143. SIAM, 2018.

[117] S. Syed and M. E. Cannon. Fuzzy logic-based map matching algorithm for vehicle navigation system in urban canyons. In *ION National Technical Meeting*, number 1, pages 26–28, 2004.

[118] P. Szwed and K. Pekala. An incremental map-matching algorithm based on hidden markov model. In *International Conference on Artificial Intelligence and Soft Computing*, pages 579–590. Springer, 2014.

[119] S. Taguchi, S. Koide, and T. Yoshimura. Online map matching with route prediction. *IEEE Transactions on Intelligent Transportation Systems*, 20(1):338–347, 2018.

[120] Y. Tang, A. D. Zhu, and X. Xiao. An efficient algorithm for mapping vehicle trajectories onto road networks. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, pages 601–604. ACM, 2012.

[121] A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Madden, H. Balakrishnan, S. Toledo, and J. Eriksson. Vtrack: accurate, energy-aware road traffic delay estimation using mobile phones. In *Proceedings of the 7th ACM conference on embedded networked sensor systems*, pages 85–98. ACM, 2009.

[122] R. Toledo-Moreo, J. M. Armingol, M. Clavijo, A. de la Escalera, J. del Ser, F. Jiménez, B. Musleh, J. E. Naranjo, I. I. Olabarrieta, and J. Sánchez-Cubillo. Positioning and digital maps. In *Intelligent Vehicles*, pages 141–174. Elsevier, 2018.

[123] R. Toledo-Moreo, D. Bétaille, F. Peyret, and J. Laneurit. Fusing gnss, dead-reckoning, and enhanced maps for road vehicle lane-level navigation. *IEEE Journal of Selected Topics in Signal Processing*, 3(5):798–809, 2009.

[124] TomTom. Tomtom. `https://www.tomtom.com/`. Accessed: 2019-11-01.

[125] Unfolding. Unfolding map. `http://unfoldingmaps.org/`. Accessed: 2019-11-01.

[126] F. Van Diggelen. Innovation: Gps accuracy-lies, damn lies, and statistics. *GPS WORLD*, 9:41–45, 1998.

[127] N. R. Velaga, M. A. Quddus, and A. L. Bristow. Developing an enhanced weight-based topological map-matching algorithm for intelligent transport systems. *Transportation Research Part C: Emerging Technologies*, 17(6):672–683, 2009.

[128] A. Vu, A. Ramanandan, A. Chen, J. A. Farrell, and M. Barth. Real-time computer vision/dgps-aided inertial navigation system for lane-level vehicle navigation. *IEEE Transactions on Intelligent Transportation Systems*, 13(2):899–913, 2012.

[129] G. Wang and R. Zimmermann. Eddy: an error-bounded delay-bounded real-time map matching algorithm using hmm and online viterbi decoder. In *Proceedings of the 22nd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 33–42. ACM, 2014.

[130] H. Wang, Z. Wang, G. Shen, F. Li, S. Han, and F. Zhao. Wheelloc: Enabling continuous location service on mobile phone for outdoor scenarios. In *2013 Proceedings IEEE INFOCOM*, pages 2733–2741. IEEE, 2013.

[131] J. Wang, X. Rui, X. Song, X. Tan, C. Wang, and V. Raghavan. A novel approach for generating routable road maps from vehicle gps traces. *International Journal of Geographical Information Science*, 29(1):69–91, 2015.

[132] S. Wang, Y. Wang, and Y. Li. Efficient map reconstruction and augmentation via topological methods. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems*, page 25. ACM, 2015.

[133] T. Wang, J. Mao, and C. Jin. Hymu: A hybrid map updating framework. In *International Conference on Database Systems for Advanced Applications*, pages 19–33. Springer, 2017.

[134] Y. Wang, X. Liu, H. Wei, G. Forman, C. Chen, and Y. Zhu. Crowdatlas: Self-updating maps for cloud and personal use. In *Proceeding of the 11th annual international conference on Mobile systems, applications, and services*, pages 27–40. ACM, 2013.

[135] H. Wei, Y. Wang, G. Forman, and Y. Zhu. Map matching by fréchet distance and global weight optimization. *Technical Paper, Departement of Computer Science and Engineering*, page 19, 2013.

[136] H. Wei, Y. Wang, G. Forman, and Y. Zhu. Map matching: comparison of approaches using sparse and noisy data. In *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 444–447. ACM, 2013.

[137] H. Wei, Y. Wang, G. Forman, Y. Zhu, and H. Guan. Fast viterbi map matching with tunable weight functions. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, pages 613–616. ACM, 2012.

[138] V. J. Wei, R. C.-W. Wong, C. Long, and D. M. Mount. Distance oracle on terrain surface. In *Proceedings of the 2017 ACM International Conference on Management of Data*, pages 1211–1226. ACM, 2017.

[139] C. E. White, D. Bernstein, and A. L. Kornhauser. Some map matching algorithms for personal navigation assistants. *Transportation research part c: emerging technologies*, 8(1-6):91–108, 2000.

[140] Wikipedia. Mercator projection. `https://en.wikipedia.org/wiki/Mercator_projection`. Accessed: 2019-11-01.

[141] Wikipedia. Restrictions on geographic data in china. `https://en.wikipedia.org/wiki/Restrictions_on_geographic_data_in_China`. Accessed: 2019-11-01.

[142] Wikipedia. World geodetic system. `https://en.wikipedia.org/wiki/World_Geodetic_System`. Accessed: 2019-11-01.

[143] H. Wu, C. Tu, W. Sun, B. Zheng, H. Su, and W. Wang. Glue: a parameter-tuning-free map updating system. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 683–692. ACM, 2015.

[144] J. Wu, Y. Zhu, T. Ku, and L. Wang. Detecting road intersections from coarse-gained gps traces based on clustering. *JCP*, 8(11):2959–2965, 2013.

[145] Z. Xiao, H. Wen, A. Markham, and N. Trigoni. Lightweight map matching for indoor localisation using conditional random fields. In *Proceedings of the 13th international symposium on Information processing in sensor networks*, pages 131–142. IEEE Press, 2014.

[146] X. Xie, W. Philips, P. Veelaert, and H. Aghajan. Road network inference from gps traces using dtw algorithm. In *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*, pages 906–911. IEEE, 2014.

[147] X. Xie, K. B.-Y. Wong, H. Aghajan, P. Veelaert, and W. Philips. Road network inference through multiple track alignment. *Transportation Research Part C: Emerging Technologies*, 72:93–108, 2016.

[148] A. Y. Xue, J. Qi, X. Xie, R. Zhang, J. Huang, and Y. Li. Solving the data sparsity problem in destination prediction. *The VLDB Journal*, 24(2):219–243, 2015.

[149] A. Y. Xue, R. Zhang, Y. Zheng, X. Xie, J. Huang, and Z. Xu. Destination prediction by sub-trajectory synthesis and privacy protection against such prediction. In *2013 IEEE 29th international conference on data engineering (ICDE)*, pages 254–265. IEEE, 2013.

[150] A. Y. Xue, R. Zhang, Y. Zheng, X. Xie, J. Yu, and Y. Tang. Desteller: A system for destination prediction based on trajectories with privacy protection. *Proceedings of the VLDB Endowment*, 6(12):1198–1201, 2013.

[151] B. Yang, J. Dai, C. Guo, C. S. Jensen, and J. Hu. Pace: a path-centric paradigm for stochastic path finding. *The VLDB Journal*, 27(2):153–178, 2018.

[152] C. Yang and G. Gidofalvi. Fast map matching, an algorithm integrating hidden markov model with precomputation. *International Journal of Geographical Information Science*, 32(3):547–570, 2018.

[153] D. Yang, B. Cai, and Y. Yuan. An improved map-matching algorithm used in vehicle navigation system. In *Proceedings of the 2003 IEEE International Conference on Intelligent Transportation Systems*, volume 2, pages 1246–1250. IEEE, 2003.

[154] L. Yang, A. Y. Xue, Y. Li, and R. Zhang. Destination prediction by identifying and clustering prominent features from public trajectory datasets. *EAI Endorsed Transactions on Scalable Information Systems*, 2(5), 2015.

[155] A. G.-O. Yeh, T. Zhong, and Y. Yue. Angle difference method for vehicle navigation in multilevel road networks with a three-dimensional transport gis database. *IEEE Transactions on Intelligent Transportation Systems*, 18(1):140–152, 2016.

[156] Y. Yin, R. R. Shah, G. Wang, and R. Zimmermann. Feature-based map matching for low-sampling-rate gps trajectories. *ACM Transactions on Spatial Algorithms and Systems (TSAS)*, 4(2):4, 2018.

[157] S. Yokoi, J.-I. Toriwaki, and T. Fukumura. An analysis of topological properties of digitized binary pictures using local features. *Computer Graphics and Image Processing*, 4(1):63–73, 1975.

[158] J. Yuan, Y. Zheng, C. Zhang, X. Xie, and G.-Z. Sun. An interactive-voting based map matching algorithm. In *Proceedings of the 2010 Eleventh International Conference on Mobile Data Management*, pages 43–52. IEEE Computer Society, 2010.

[159] K. Zheng, Y. Zheng, X. Xie, and X. Zhou. Reducing uncertainty of low-sampling-rate trajectories. In *2012 IEEE 28th International Conference on Data Engineering*, pages 1144–1155. IEEE, 2012.

[160] K. Zheng and D. Zhu. A novel clustering algorithm of extracting road network from low-frequency floating car data. *Cluster Computing*, pages 1–10, 2018.

[161] R. Zheng, Q. Liu, W. Rao, M. Yuan, J. Zeng, and Z. Jin. Topic model-based road network inference from massive trajectories. In *Mobile Data Management (MDM), 2017 18th IEEE International Conference on*, pages 246–255. IEEE, 2017.

[162] L. Zhu, J. R. Holden, and J. D. Gonder. Trajectory segmentation map-matching approach for large-scale, high-resolution gps data. *Transportation Research Record*, 2645(1):67–75, 2017.