Map matching imprecise trajectories for smart mobility applications

Jagadeesh, George Rosario

2019

Jagadeesh, G. R. (2019). Map matching imprecise trajectories for smart mobility applications. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/89847>

<https://doi.org/10.32657/10220/47724>

MAP MATCHING IMPRECISE TRAJECTORIES FOR SMART MOBILITY APPLICATIONS

GEORGE ROSARIO JAGADEESH

School of Computer Science and Engineering

A thesis submitted to the Nanyang Technological University
in partial fulfilment of the requirement for the degree of
Doctor of Philosophy

2019

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

11/02/2019

.....
Date

G. R. Jagadeesh

.....
George Rosario Jagadeesh

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiarism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

.....11/2/2019.....
Date

....
Prof. Thambipillai Srikanthan

Authorship Attribution Statement

This thesis contains material from four papers published in the following peer-reviewed journal and conferences where I was the first and corresponding author.

The work presented in Chapters 3 and 4 and an early version of this work has been published as follows:

- G. R. Jagadeesh and T. Srikanthan, "Online Map-Matching of Noisy and Sparse Location Data with Hidden Markov and Route Choice Models", *IEEE Transactions on Intelligent Transportation Systems*, vol. 18(9), pp. 2423-2434, 2017.
- G. R. Jagadeesh and T. Srikanthan, "Probabilistic Map Matching of Sparse and Noisy Smartphone Location Data", *IEEE 18th International Conference on Intelligent Transportation Systems (ITSC)*, Las Palmas, pp. 812-817, September 2015.
- G. R. Jagadeesh and T. Srikanthan, "Robust Real-Time Route Inference from Sparse Vehicle Position Data", *IEEE 17th International Conference on Intelligent Transportation Systems (ITSC)*, Qingdao, pp. 296-301, October 2014.

A preliminary version of the work presented in Chapter 5 has been published as follows:

- G. R. Jagadeesh and T. Srikanthan, "Heuristic optimizations for high-speed low-latency online map matching with probabilistic sequence models", *IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, Rio de Janeiro, pp. 2565-2570, November 2016.

The contributions of the co-authors in the above papers are as follows:

- I collected trajectory data, designed and implemented the algorithms, carried out the experimental evaluations and drafted the manuscripts.
- Prof. Srikanthan guided the study, reviewed the experimental results and the manuscripts, and suggested improvements.

11/02/2019
.....

Date

G. R. Jagadeesh
.....

George Rosario Jagadeesh

Acknowledgements

I owe a great debt of gratitude to my supervisor and mentor Prof. Thambipillai Srikanthan for his guidance, support and kindness. I particularly appreciate the freedom and flexibility he gave me in performing my research amidst other commitments. His feedback on my work has been valuable and has enabled me to examine the relevance of the research from a broader perspective. Apart from his role in this PhD project, I wish to thank Prof. Srikanthan for all that he has done for me during my long association with him.

I gratefully acknowledge the support from Mohit Sindhwani and Chan Siew Kiat of Quantum Inventions who provided some of the data used in this work. I wish to thank Riya Bubna for her help in performing some early experiments on route choice modelling. I am grateful to the anonymous reviewers of my journal papers, whose feedback has helped me to improve my work.

I would like to thank my colleagues and friends at the Hardware and Embedded Systems Lab for the pleasant working environment. A special word of thanks is due to Jeremiah Chua, who has always promptly responded to my requests for technical assistance.

Finally, I wish to express my gratitude to my family. This thesis would not have been possible without the support and understanding of my wife Sheeba who has shouldered most of the domestic and parenting responsibilities so that I can focus on my research. My sons, Iniyan and Nilavan, have been a great source of joy and motivation. I am fortunate to have the support and affection of my parents, siblings, parents-in-law and other family members.

Table of Contents

List of Figures	viii
List of Tables	xii
Abstract	xiii
1 Introduction	1
1.1 Motivation	1
1.2 Objectives and Scope	4
1.3 Summary of Contributions	5
1.3.1 Publications	7
1.4 Organization of the Thesis	8
2 Background and Literature Review	9
2.1 Trajectory Data and Applications	9
2.1.1 Sources of Trajectory Data	10
2.1.2 Positioning Technologies in Smartphones	11
2.1.3 Trajectory-Based Smart Mobility Applications	14
2.2 Review of Map Matching Algorithms	16
2.2.1 Geometrical Map Matching Algorithms	19
2.2.2 Route-Based Map Matching Algorithms	22
2.2.3 Probabilistic Map Matching Algorithms	25
2.3 Discussion and Summary	38

3	A Probabilistic Map Matching Algorithm	42
3.1	Problem Statement	42
3.2	The HMM-Based Map Matching Algorithm	43
3.2.1	Emission Probability Model.....	45
3.2.2	Transition Probability Model.....	47
3.2.3	Determination of the Most Likely State Sequence	50
3.2.4	Methods for Estimation of Model Parameters.....	52
3.3	The Experimental Setup	53
3.3.1	Trajectory Datasets.....	54
3.3.2	Estimation of Parameters from the Trajectory Datasets	56
3.3.3	Evaluation Method and Metrics	59
3.4	Evaluation Results.....	60
3.5	Summary	70
4	A Route Choice Model for Robust Map Matching	71
4.1	A Limitation of HMM-based Map Matching.....	71
4.2	Short-Term Route Choice Modelling.....	73
4.2.1	Choice Set Generation.....	73
4.2.2	Route Choice Model.....	76
4.2.3	Estimation of Model Parameters	78
4.3	The Improved Map Matching Algorithm	79
4.4	Evaluation.....	82
4.5	Summary	87
5	Optimization Techniques for Online Map Matching	88
5.1	Optimization for Improving Output Latency	88
5.1.1	Background and Related Work	88
5.1.2	The Proposed Method.....	90
5.1.3	Evaluation	93

5.2	Optimization for Improving Computation Time	98
5.2.1	Related Work.....	99
5.2.2	Algorithm for Clustered MSP Computation	102
5.2.3	Evaluation	109
5.3	Summary	113
6	Traffic Condition Estimation From Imprecise Trajectories	114
6.1	Overview of Traffic Condition Estimation	114
6.2	A Confidence Measure for Map Matching.....	120
6.3	Link Speed Estimation Method.....	122
6.3.1	Stopping Delay Estimation	124
6.3.2	Travel Time Allocation and Speed Estimation	128
6.4	Evaluation.....	129
6.4.1	Evaluation of the Confidence Measure	130
6.4.2	Estimation of Mean Stopping Delays and Stopping Probability Parameters	133
6.4.3	Evaluation of the Traffic Condition Estimation System.....	134
6.5	Summary	138
7	Conclusions and Future Work	140
7.1	Conclusions	140
7.2	Future Work.....	144
	References	147
	Appendix A: Dijkstra's Shortest Path Algorithm	161

List of Figures

2.1:	Wi-Fi Positioning.....	13
2.2:	Illustration of two forms of map matching	17
2.3:	Illustration of the concept of candidate paths for an input trajectory on a simple grid-like road network.....	21
2.4:	Illustration of the HMM framework for map matching.	27
2.5:	A linear-chain conditional random field where unshaded and shaded nodes represent state and observed variables, respectively.	34
3.1:	HMM representation for map matching.	44
3.2:	An example of the convergence condition in the Viterbi algorithm.	52
3.3:	Roads travelled during the collection of cellular trajectories.....	55
3.4:	Histogram of the measure of temporal implausibility.....	58
3.5:	Histogram of the measure of circuitousness	58
3.6:	Comparison of map matching error on the cellular dataset	61
3.7:	Comparison of map matching error on the Wi-Fi-and-cellular dataset...	61
3.8:	Comparison of map matching error on the synthetic dataset (100 m noise)	62
3.9:	Comparison of map matching error on the synthetic dataset (200 m noise).....	62
3.10:	Comparison of map matching error on the synthetic dataset (300 m noise).....	62
3.11:	Comparison of map matching error on the synthetic dataset (400 m noise).....	63

3.12:	Comparison of map matching error on the synthetic dataset (500 m noise).....	63
3.13:	Comparison of mean output latency on the cellular dataset	65
3.14:	Comparison of mean output latency on the Wi-Fi-and-cellular dataset.	65
3.15:	Comparison of mean output latency on the synthetic dataset (100 m noise)	65
3.16:	Comparison of mean output latency on the synthetic dataset (200 m noise).....	66
3.17:	Comparison of mean output latency on the synthetic dataset (300 m noise).....	66
3.18:	Comparison of mean output latency on the synthetic dataset (400 m noise).....	66
3.19:	Comparison of mean output latency on the synthetic dataset (500 m noise).....	67
3.20:	Comparison of running time on the cellular dataset.....	68
3.21:	Comparison of running time on the Wi-Fi-and-cellular dataset	68
3.22:	Comparison of running time on the synthetic dataset (100 m noise)	68
3.23:	Comparison of running time on the synthetic dataset (200 m noise)	68
3.24:	Comparison of running time on the synthetic dataset (300 m noise)	69
3.25:	Comparison of running time on the synthetic dataset (400 m noise)	69
3.26:	Comparison of running time on the synthetic dataset (500 m noise)	69
4.1:	A map-matched path that is improbable from a route choice perspective	72
4.2:	An example of multiple alternative paths produced by the choice set generation method	75
4.3:	Evaluation of the improved algorithm on the cellular dataset.....	84
4.4:	Evaluation of the improved algorithm on the Wi-Fi-and-cellular dataset	84

4.5:	Evaluation of the improved algorithm on the synthetic dataset (100 m noise)	85
4.6:	Evaluation of the improved algorithm on the synthetic dataset (200 m noise).....	85
4.7:	Evaluation of the improved algorithm on the synthetic dataset (300 m noise).....	85
4.8:	Evaluation of the improved algorithm on the synthetic dataset (400 m noise).....	86
4.9:	Evaluation of the improved algorithm on the synthetic dataset (500 m noise).....	86
5.1:	An illustration of back pointers in the Viterbi algorithm.....	91
5.2:	Comparison of mean output latency on the cellular dataset	96
5.3:	Comparison of mean output latency on the Wi-Fi-and-cellular dataset.	96
5.4:	Comparison of mean output latency on the synthetic dataset (100 m noise).	97
5.5:	Comparison of mean output latency on the synthetic dataset (200 m noise).....	97
5.6:	Comparison of mean output latency on the synthetic dataset (300 m noise).....	97
5.7:	Comparison of mean output latency on the synthetic dataset (400 m noise).....	98
5.8:	Comparison of mean output latency on the synthetic dataset (500 m noise).....	98
5.9:	A simple grid network showing potential exit nodes for a source region.....	104
5.10:	Simplified representation of the abstract graph G'	106
5.11:	Error regions of consecutive location measurements may overlap.	109
5.12:	Comparison of running time on the cellular dataset.....	110
5.13:	Comparison of running time on the Wi-Fi-and-cellular dataset.	110

5.14:	Comparison of running time on the synthetic dataset (100 m noise)	110
5.15:	Comparison of running time on the synthetic dataset (200 m noise)	111
5.16:	Comparison of running time on the synthetic dataset (300 m noise)	111
5.17:	Comparison of running time on the synthetic dataset (400 m noise)	111
5.18:	Comparison of running time on the synthetic dataset (500 m noise)	111
6.1:	Screen capture from Google Maps showing live traffic conditions.....	115
6.2:	An example showing two consecutive map-matched points and the part of the overall path that lies between them	117
6.3:	Overview of the traffic condition estimation system used in this work.	118
6.4:	An example of the input to the link speed estimation method	123
6.5:	Stopping probability as a function of distance to the upcoming intersection	126
6.6:	Two consecutive map-matched points on the same road segment	128
6.7:	Evaluation of the confidence measure on the synthetic dataset (100 m noise)	131
6.8:	Evaluation of the confidence measure on the synthetic dataset (200 m noise).....	132
6.9:	Evaluation of the confidence measure on the synthetic dataset (300 m noise).....	132
6.10:	Evaluation of the confidence measure on the synthetic dataset (400 m noise).....	132
6.11:	Evaluation of the confidence measure on the synthetic dataset (500 m noise).....	133
6.12:	Histogram of the running-speed estimation error.....	137

List of Tables

2.1:	A summary of selected HMM-based map matching algorithms.....	28
3.1:	Estimated parameters of the Gaussian mixture distribution	57
3.2:	Average map matching errors across all sampling intervals	61
4.1:	Estimated parameters of the route choice model	82
5.1:	Overall mean output latency and map matching error for different values of threshold τ across all datasets and sampling intervals.....	94
5.2:	Mean output latencies for each dataset, averaged across all sampling intervals	95
6.1:	Precision and recall of map matching for different values of confidence measure threshold, averaged across all noise levels and sampling intervals	131
6.2:	Mean stopping delays in seconds for different combinations of intersection types and turning manoeuvres, estimated from a real trajectory dataset	133
6.3:	Mean absolute errors (km/h) of running speed estimates for expressways.....	135
6.4:	Mean absolute errors (km/h) of running speed estimates for non-expressway roads.	135
6.5:	Mean absolute errors (km/h) of running speed estimates for all roads	135

Abstract

Trajectory data generated by smart mobile devices are useful for inferring traffic conditions and travel patterns in order to enable smart mobility. The effective utilisation of such data depends on a crucial step known as map matching that matches raw trajectories to paths in a transport network. Existing trajectory-based applications mostly rely on relatively precise trajectories generated by the Global Positioning System (GPS). However, it is preferable to use noisy trajectories generated by alternative technologies such as Wi-Fi positioning and cellular network positioning due to their high energy efficiency, availability in GPS-denied environments and better privacy protection. Furthermore, bandwidth and storage-space considerations dictate that trajectories be sampled sparsely. This thesis examines the challenging problem of map matching such noisy and sparse trajectories in an accurate and efficient manner.

Following a critical review of the literature, a Hidden Markov Model (HMM) based algorithm is proposed for map matching temporally sparse trajectories with high levels of location measurement noise. It includes new probabilistic models for modelling the measurement noise and representing the probability of transitioning between candidate points on a road network. Evaluations with several imprecise trajectory datasets show that the proposed algorithm is consistently more accurate compared to a widely-used baseline algorithm and produces a 55% reduction in the overall map matching error.

An innovative approach is adopted to further improve the robustness of the HMM-based map matching algorithm against noise and sparsity in the input data. The HMM-based algorithm is complemented with a model of drivers' route choice in order to assess partial paths produced by the former from the

perspective of route preference. A multinomial logit route choice model, estimated from a real trajectory dataset, is used to calculate the probability of a path being chosen among several alternative paths, which are generated by an effective choice set generation procedure. Experimental results show that the use of the route choice model leads to a 12% reduction in the error with respect to the HMM-based algorithm.

Online map matching of imprecise trajectories incurs substantial output latencies and computation times, thereby rendering them unsuitable for real-time applications. This limitation is addressed by devising optimization techniques for improving the timeliness and efficiency of map matching. A heuristic technique based on a probabilistic measure is presented to enable early generation of partial state sequences in the HMM by an online variant of the Viterbi algorithm. This results in an overall reduction of 33% in the output latency with negligible loss of accuracy. A novel method is proposed for accelerating a computationally expensive step involved in probabilistic map matching, namely, the computation of many-to-many shortest paths between candidate points. This produces a reduction of up to 78% in the overall computation time. The above research contributions have led to the realization of an efficient map matching solution that is robust to high levels of imprecision in the input data.

The applicability of the proposed map matching solution for an important smart mobility application, namely, traffic condition estimation, is evaluated. For this, a system that estimates road-segment-wise running speeds from imprecise trajectory data is proposed. The system relies on a confidence-based filtering approach to cope with the uncertainties in the map-matched paths. It also includes a novel method of estimating the stopping delays at intersections along the paths. Experimental results show that despite relying on noisy and sparse trajectories, the accuracy of speed estimates produced by the system are found to be within the range generally considered acceptable for applications that use traffic data. As the collection of sparse non-GPS trajectories from smartphone users imposes only marginal energy, bandwidth and privacy costs on them, the contributions of this thesis can potentially pave the way for large scale deployment of trajectory-based smart mobility applications.

Chapter 1

Introduction

1.1 Motivation

Urban mobility is widely regarded as one of the biggest challenges currently faced by cities worldwide. Traffic congestion has emerged as a serious problem with huge economic costs. The total annual cost of traffic congestion in the USA, the UK, France and Germany has been estimated to be \$200 billion in 2013 [1]. With the urban population worldwide predicted to grow at a rapid rate [2], the traffic congestion problem is expected to worsen in the coming years. As there are constraints limiting the expansion of the transport infrastructure to meet the growing travel demand, authorities in most countries have turned to Intelligent Transport Systems (ITS), which are applications that rely on a variety of advanced technologies to achieve smarter use of transport networks. In recent years, mainly due to the pervasiveness of smart mobile devices, there has been a shift in the nature of ITS with greater importance being placed on data collection, analytics and the availability of relevant, timely information [3]. The term *smart mobility* has come to be widely used to describe ITS solutions that rely on data analytics and sharing of real-time information.

Data from location-aware mobile devices form an important part of the data used by smart mobility solutions. Most smartphone users agree to share their location information with various application providers, who periodically track the phone's location for a variety of purposes. Irrespective of the original reason

for collecting location data, they are generally usable for smart mobility solutions. Raw location data corresponding to a moving object is known as a *trajectory* [4], which is a time-ordered sequence of points corresponding to the estimated locations of the object. While trajectories may represent many different types of movements, this thesis is concerned with trajectories generated by devices on board vehicles moving along road networks. A number of smart mobility applications that process and extract valuable mobility-related information from trajectory data have emerged in recent times. These include traffic condition estimation [5, 6], travel behaviour analysis [7, 8], practical route discovery [9, 10] and future location prediction [11, 12].

Trajectory-based applications such as the above-mentioned ones involve a fundamental step known as map matching. Map matching, also referred to as ‘path inference’ in some contexts, is the process of matching a sequence of location measurements (i.e. a raw trajectory) to a sequence of road segments (i.e. a path) in a road network. While it may appear to be straightforward, map matching is actually a difficult task mainly due to two sources of imprecision present in trajectory data, which are commonly known as measurement error and sampling error [13]. Measurement error (or ‘noise’) depends on the accuracy of the positioning technology used for location determination. Sampling error depends on the rate at which the location of the mobile device is sampled. Existing map matching algorithms perform very well when the measurement and sampling errors are low. However, map matching becomes highly challenging when the degree of imprecision in the trajectory data is high.

A number of positioning technologies such as Global Positioning System (GPS), Wi-Fi positioning and cellular network positioning are used for determining the location of a smart mobile device. The accuracies of the above positioning technologies vary significantly. One study found the median errors of GPS, Wi-Fi and cellular network positioning to be 8 m, 74 m and 600 m, respectively [14]. Smartphone operating systems such as Android allow applications to either access fine-grained location based on GPS or coarse-grained

location based on Wi-Fi access points and cellular networks [15]. Despite the superior accuracy of GPS, it is advantageous to use coarse-grained location estimates produced by alternative positioning technologies, especially for projects that rely on crowdsourcing [16], due to the following reasons.

- GPS-based location determination depletes the smartphone's battery several times faster than the other positioning technologies [17-19]. For instance, cellular network positioning has been found to be at least 10 times more energy-efficient compared to GPS [18]. Users attach considerable importance to the phone's battery life and are not receptive to battery-draining applications. Many users keep GPS disabled in order to conserve battery power. As a result, many smartphone applications collecting trajectory data prefer to rely on energy-efficient non-GPS positioning technologies (e.g., [12] and [20]).
- There are many situations in which GPS is either not available or performs poorly. For instance, GPS is generally unavailable in tunnels and most 'urban canyons' (i.e. streets flanked by tall buildings), where the GPS signals transmitted by satellites are obstructed. The proportion of roads without adequate reception of GPS signals is increasing due to increasing urban densities. GPS signal reception in smartphones is also significantly impeded when they are placed in pockets, handbags etc. In such situations without GPS availability, only coarse-grained trajectories of the mobile user, generated by alternative positioning technologies, are obtainable.
- There are significant privacy risks associated with the collection of precise, fine-grained trajectory data. Such data can be used to identify places visited by a user, establish his or her identity and extract various sensitive information [21, 22]. This vulnerability is typically addressed by intentionally degrading the accuracy of the locations in order to obfuscate them [23, 24]. Against this background, some trajectory-based solutions ensure privacy protection by opting for imprecise locations provided by Wi-Fi and cellular network positioning [25].

Sampling error is the other major factor affecting the quality of trajectory data. Due to data bandwidth and storage-space considerations, most smartphone applications that run continuously in the background sample the location of the mobile users at a low sampling rate, with the sampling interval typically in the order of a few minutes. This creates gaps in the data that give rise to uncertainty regarding the movement of the user between the sampled data points. Such uncertainty due to the temporal sparsity of data is also a common feature of trajectories extracted from geotagged social media posts [26], an emerging source of trajectory data.

Despite the prevalence and advantages of sparse non-GPS trajectory data, their use for smart mobility applications is hindered by the lack of effective techniques for coping with the high degree of imprecision present in such data. In particular, most of the existing map matching algorithms deal with relatively accurate GPS data and perform adequately only when the sampling interval is below certain limits. Recently, there has been an increased recognition in the literature about the need to map match sparse non-GPS trajectories [27-30]. However, this remains a challenging problem in need of robust and efficient solutions.

1.2 Objectives and Scope

The aim of this research is to develop effective algorithmic techniques for map matching imprecise trajectory data in order to enable the use of such data for smart mobility applications. Its scope is limited to smartphone-generated trajectories corresponding to the movements of vehicles along road networks. (Techniques such as [31] and [25] can be used to filter out trajectories that do not correspond to vehicular travel, but that is outside the scope of this thesis.) Furthermore, this research focuses on *online* map matching, which involves timely processing of streaming trajectory data. The objectives of this research project are outlined below:

- To perform a critical study of the literature on map matching and to identify avenues for advancing the state of the art.
- To develop a robust map matching algorithm capable of coping with the uncertainties present in imprecise trajectory data.
- To optimize the computational efficiency and timeliness of map matching for real-time processing of large scale data.
- To evaluate the feasibility of using the developed solution, along with imprecise input data, for smart mobility applications.

1.3 Summary of Contributions

The major research contributions of this thesis are summarised as follows:

A comprehensive study of the various approaches adopted towards solving the challenging problem of map matching imprecise trajectories is performed (Chapter 2). More than 40 solutions proposed in the literature are reviewed and classified according to a coherent taxonomy. The merits and demerits of each approach are critically examined.

A robust probabilistic framework, developed in two stages, is presented for map matching imprecise trajectory data (Chapters 3 and 4, published in [32] and [33]). In the first stage, a Hidden Markov Model (HMM) based approach is adopted. Effective probabilistic models are proposed for modelling the noise present in the location measurements and for representing the probability of transitioning between candidate points on a road network. Evaluations using real as well as synthetic-noise trajectory datasets show a substantial improvement in the map matching accuracy compared to a baseline algorithm. In the second stage, a refinement is proposed to overcome the inability of the HMM-based map matching approach to take route context information into consideration. For this, a model of drivers' route choice behaviour is developed and estimated from a trajectory dataset. A method of combining the route choice model with the

HMM-based solution is proposed. This refinement significantly improves the map matching accuracy, especially with high levels of sparsity and noise in the input data.

Optimization techniques are proposed for improving the efficiency of online map matching in terms of output latency and computation time (Chapter 5, published and awaiting publication, respectively, in [34] and [35]). A sizeable reduction in the output latency is achieved by applying a heuristic method based on a probabilistic measure to generate partial map-matched outputs earlier than the baseline algorithm. An efficient algorithm is proposed to speed-up a computationally expensive step involved in probabilistic map matching, namely, the computation of many-to-many shortest paths between clusters of candidate points. The proposed algorithm for clustered many-to-many shortest path computation, when used as a subroutine in the map matching algorithm, produces a major reduction in the overall computation time. The abovementioned optimization techniques are applicable not only to the proposed map matching algorithm, but also to most other probabilistic map matching algorithms in the literature.

The map matching solution developed in this work is applied to the task of estimating traffic conditions from imprecise trajectory data in order to study the feasibility of such a proposition (Chapter 6). As misidentified road segments in the map-matched path have an adverse impact on the quality of traffic condition estimation, an effective confidence measure that improves the precision of map matching at the cost of its recall is devised. A method is presented for estimating the running speed along each of the road segments in the map-matched path by proportioning the elapsed time among them while factoring in the potential delays at the intersections. Through experimental evaluation, the accuracy of the estimated running speeds is found to be reasonably robust to the imprecision in the trajectory data.

1.3.1 Publications

Some parts of the research described in this thesis have been published or awaiting publication in international journals and conferences as given below.

Journals

1. G. R. Jagadeesh and T. Srikanthan, "Online Map-Matching of Noisy and Sparse Location Data with Hidden Markov and Route Choice Models", *IEEE Transactions on Intelligent Transportation Systems*, vol. 18(9), pp. 2423-2434, 2017 [33].
2. G. R. Jagadeesh and T. Srikanthan, "Fast Computation of Clustered Many-to-Many Shortest Paths and its Application to Map Matching", *ACM Transactions on Spatial Algorithms and Systems (TSAS)*, 2019. (Under revision) [35].

Conferences

1. G. R. Jagadeesh and T. Srikanthan, "Robust Real-Time Route Inference from Sparse Vehicle Position Data", *IEEE 17th International Conference on Intelligent Transportation Systems (ITSC)*, Qingdao, pp. 296-301, October 2014 [36].
2. G. R. Jagadeesh and T. Srikanthan, "Probabilistic Map Matching of Sparse and Noisy Smartphone Location Data", *IEEE 18th International Conference on Intelligent Transportation Systems (ITSC)*, Las Palmas, pp. 812-817, September 2015 [32].
3. G. R. Jagadeesh and T. Srikanthan, "Heuristic optimizations for high-speed low-latency online map matching with probabilistic sequence models", *IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, Rio de Janeiro, pp. 2565-2570, November 2016 [34].

1.4 Organization of the Thesis

The rest of this thesis is organised as follows:

- Chapter 2 provides background information on topics relevant to this research and presents a comprehensive review of the literature on map matching in order to identify potential ways in which the current state of the art could be advanced.
- Chapter 3 presents a probabilistic algorithm based on HMM for online map matching of imprecise trajectories. The experimental setup used for the evaluation of the proposed algorithm is described in detail and the evaluation results are presented.
- Chapter 4 first describes a procedure for modelling short-term route choice behaviour from real data. Subsequently, the improved map matching algorithm, in which a route choice model is used to complement the HMM-based algorithm, is presented and evaluated.
- Chapter 5 covers the description and evaluation of the optimization techniques proposed for improving the output latency and computation time, respectively, of the map matching algorithm.
- Chapter 6 describes the application of the proposed map matching solution for traffic condition estimation. A confidence measure to filter out likely erroneous portions of the map-matched paths is first proposed. Then, a method of estimating the road-segment-wise running speeds is presented and evaluated.
- Chapter 7 concludes the thesis by summarising the salient research results and outlines ways in which this work can be extended in future.

Chapter 2

Background and Literature Review

This chapter presents the context in which the proposed research is situated. First, an overview of the topic of trajectory data is provided including the sources of such data, technologies used for generating them and applications that rely on them. Subsequently, a comprehensive review of the existing body of work on map matching, the fundamental step that makes trajectory data usable, is presented.

2.1 Trajectory Data and Applications

The abundance of data is one of the defining characteristics of the current age. Most individuals in modern societies generate digital data related to almost every aspect of their lives through their use of smart mobile devices and their interaction with a range of businesses and services. Large companies are known to capture thousands of terabytes of data from users. While unregulated collection and utilization of personal data has negative implications, such data can also be used for the common good. In particular, aggregated trajectory data that captures the movements of individuals, when correctly processed, has huge potential for improving transport planning, traffic management and the quality of information provided to travellers.

In the context of this research, a trajectory is a time-ordered discrete sequence of raw location measurements corresponding to the movement of a person along a road network in a motor vehicle. A location measurement consists

of an estimated latitude, longitude and timestamp. In some cases, trajectory data may also include additional information such as instantaneous speed and heading. However, such additional information is generally not available and therefore is not considered in this work.

2.1.1 Sources of Trajectory Data

Prior to the mass adoption of smart mobile devices, fleet vehicles such as taxis fitted with a GPS-based tracking device served as a major source of trajectory data. Large scale trajectory data sourced from taxi operators have been utilized for ITS applications in many cities worldwide including Singapore [37], Beijing [38] and Stockholm [39]. However, this source of trajectory data has now been eclipsed in scale by trajectory data generated by smart mobile devices. During the early period of smartphone adoption in 2010, a study found that trajectory data from smartphones were growing at double the average rate of growth of all other sources of trajectory data [40]. It is highly likely that this trend has become much stronger in recent years due to the exponential growth in the number of smartphones and a vast proportion of the currently generated trajectory data comes from smartphones. This research work focusses on trajectory data produced by smartphones.

A large majority of smartphone users currently share trajectory data with various service providers. In most cases, the sharing of trajectory data is carried out without any intentional effort from the users. Smartphone-generated trajectory data generally belong to one of the following categories.

- **Location tracking by applications:** Many smartphone applications request and obtain the user's consent to access information regarding the phone's location. A recent experimental study indicates that various applications typically access a phone's location thousands of times in a week [41]. This is apparently done to improve the service provided in some manner such as providing the most relevant query results or location-based weather updates.

- **Indirect location sharing by users:** In some situations, users may unintentionally generate and share a trajectory when they perform a sequence of actions on the smartphone. For instance, a trajectory can be extracted from a series of geotagged social media content, such as photos or textual messages, posted by a user [26]. Similarly, a trajectory can be constructed from a sequence of *check-ins* made by a user in a location-based social network such as Foursquare.
- **Cell-IDs tracked by cellular operators:** Cellular service providers continually track the base station to which a phone is connected to. They also estimate a parameter called timing advance, which correlates with the distance between a base station and a phone. The above information, along with data on the locations and range of the base stations can be used to estimate the location of the phone. Some cellular operators are capable of further improving the location estimates through techniques such as time difference of arrival (TDOA) and angle of arrival (AOA) [42].

2.1.2 Positioning Technologies in Smartphones

Most smartphones have the capability to receive radio signals from different sources such as GPS satellites, Wi-Fi access points and cellular base stations. By relying upon this capability, a smartphone can determine its location in a number of ways as outlined below.

Global Positioning System

The GPS [43] is a satellite-based radio navigation system that provides location and time information around the globe. The GPS constellation originally consisted of 24 satellites arranged in six orbital planes with 4 satellites per orbital plane. The satellites are equipped with high-precision atomic clocks. A GPS receiver notes the time interval taken for a signal transmitted from a satellite at a known location to reach it and derives the satellite-to-receiver distance. The observation of at least four satellites will permit determination of the co-

ordinates of the receiver using trilateration. When a high-quality GPS receiver has an unobstructed view of four or more GPS satellites, it can determine its location with a horizontal position accuracy of less than 2 m in 95% of the cases [44]. GPS receivers embedded in smartphones are generally less accurate compared to dedicated GPS receivers. A large worldwide study in 2015 found that the mean GPS location accuracy of smartphones is 4.9 m under open-sky conditions [45]. However, GPS accuracy degrades in densely built-up urban areas due to satellite occlusions and multipath reflections. When the GPS functionality is used in a smartphone, it depletes the phone's battery at a much faster rate compared to other alternative positioning methods [17-19]. This is because GPS receivers need to use complex signal processing techniques that consume substantial amount of energy in order to process the weak signals received from GPS satellites located more than 20,000 km above the earth's surface.

Wi-Fi Positioning

Wi-Fi positioning systems estimate the location of a smart mobile device based on the strengths of the signals received from a number of uniquely-identified Wi-Fi access points. The set of access points and the strength of the respective signals received from them are collectively considered as a Wi-Fi *fingerprint* that is used to determine the location. A prerequisite for Wi-Fi positioning is the existence of a large database of Wi-Fi fingerprints tagged to known locations. Such databases are generally created through a process known as wardriving, where a device equipped with both Wi-Fi and GPS is taken through an area to record the Wi-Fi signals and the GPS locations at which they were received [14]. Wi-Fi fingerprint databases are commonly built through crowdsourcing, where a large number of users contribute data for populating the database. There are several public Wi-Fi fingerprint databases with data from several hundred million Wi-Fi access points (e.g. WiGLE [46]). Companies such as Apple and Google maintain their own crowdsourced Wi-Fi fingerprint databases and cater to the users of iOS and Android mobile operating systems, respectively. When a smartphone needs to determine its location, it performs a Wi-Fi scan and supplies the received

Figure 2.1: Wi-Fi Positioning

fingerprint to a location server that compares it with the fingerprints in the database and supplies the location corresponding to the closest match. The Wi-Fi positioning method is illustrated in Figure 2.1.

While most estimates of typical Wi-Fi positioning accuracy published by Wi-Fi location service providers are in the range of 20-40 m, an independent study found the median error to be 74 m [14]. When a mobile device receives Wi-Fi signals from several reference access points, the location error rarely exceeds 150 m. However, there is a possibility of higher errors occurring due to the relocation of previously mapped reference access points. Wi-Fi positioning consumes much less battery power compared to GPS [19]. However, unlike GPS, Wi-Fi positioning requires data connectivity for communicating with the location server. A major limitation of Wi-Fi positioning is that it generally does not work in non-built-up areas where Wi-Fi signals are absent.

Cellular Network Positioning

A smartphone is generally capable of observing signals of varying strengths from multiple cellular base stations. If the true locations of the base stations are known, it is possible to estimate the smartphone's location using standard geometric techniques such as trilateration [47]. However, the base station locations are usually not made public by the cellular operators. Therefore, similar to Wi-Fi positioning, techniques that rely on cellular base station fingerprints are used in practice [18]. In the case of Android smartphones, when the phone's coarse

location is needed, a fingerprint comprising of both Wi-Fi and cellular signal information is sent to the Google location server, which returns a location. Details of the positioning algorithm used by Google have not been published. Cellular network positioning is significantly more energy efficient compared to the other alternatives. Also, it has much better coverage than Wi-Fi positioning and is usually the only available option in places without GPS and Wi-Fi signals. However, the accuracy of the locations determined solely using cellular network positioning is generally quite low. The location accuracy depends on the density of the cellular base stations. In urban areas, where the cell density is higher, the average error of cellular network positioning is of the order of a few hundred metres [14, 48].

2.1.3 Trajectory-Based Smart Mobility Applications

Trajectory data aggregated from a large number of individuals in a city contains valuable latent information regarding several aspects of urban mobility such as the spatiotemporal distribution of traffic congestion, travel demand in terms of trip origins and destinations, and traveller behaviour in the choice of travel modes and routes. A number of applications that extract useful mobility information from trajectory data have emerged, mostly over the last decade. They belong to a key focal area of smart mobility initiatives, namely, the creation and delivery of high quality transport information to meet diverse needs [3]. In the following, a few relevant trajectory-based smart mobility applications are briefly described.

- **Traffic condition estimation** [5, 6, 49, 50]: Trajectory data obtained from mobile users can be used to estimate the travel time, speed and the degree of congestion along the traversed road segments. Such estimation of traffic conditions provides a higher degree of coverage, accuracy and cost effectiveness compared to conventional traffic data collection methods. It has been estimated that providing drivers with smart, congestion-avoiding routes based on real-time traffic data will result in annual savings of about \$500 billion in terms of time and fuel by 2020 [40].

- **Travel behaviour analysis** [7, 8]: Understanding aspects of individual travel behaviour such as route choice is important for accurately estimating the travel demand in a transportation system. Conventional travel survey methods used for collecting travel behaviour data are considered to be not respondent friendly [51]. In recent years, trajectory data obtained from the smartphones of participants, without any explicit effort from them, has emerged as an attractive alternative to conventional travel surveys.
- **Future location prediction** [11, 12]: Given the trajectory obtained from a mobile user on a road network up to a certain time, it is possible to predict the subsequent path and the intended destination based on knowledge about movement patterns. Such an ability to predict future paths and destinations is useful for several location based services such as providing targeted advertising, relevant traffic updates and point-of-interest recommendations.
- **Practical route discovery** [9, 10]: Currently, millions of users rely on online services such as Google Maps to provide them with optimal routes to their destinations. However, there exist routes that may not be the theoretically optimal route based on standard criteria (e.g. fastest path), but are preferred by experienced drivers based on a variety of factors. It is useful to discover such practical routes in a time-dependent manner by mining large-scale trajectory data.

The above applications cannot directly make use of raw trajectories, which are nothing but sequences of timestamped latitude-longitude coordinates. They require the trajectories to be expressed in terms of a path in a road network made up of connected road segments. Therefore, a map-matching algorithm is used to infer the travelled path from raw trajectories. The quality of the outputs produced by the above applications depends on the accuracy of the map matching step.

2.2 Review of Map Matching Algorithms

It is reasonable to assume that a vehicle is confined to points on a finite network of roads during a trip. However, due to the inaccuracies of the positioning technologies used, the measured locations of a vehicle typically lie away from its true on-road locations. Map matching, in its basic form, is the process of associating each raw location measurement obtained from a vehicle to a point on the road network map. This process is illustrated using an example in Figure 2.2(a), where the black dots denote the raw locations and the stars indicate the map-matched on-road points. In this example, the map-matched points do not lie along a connected path that could have been travelled by the vehicle. (The expressway segment on which the second map-matched point from the top lies is not connected to the slip-road segment containing the next map-matched point.) This type of map-matching, referred to as *basic map matching* in this thesis, does not seek to infer the path travelled by the vehicle.

Basic map matching is most commonly used for the real-time tracking of vehicles in vehicle navigation and fleet tracking systems. In such systems, GPS locations are typically obtained every second and immediately map matched to a point on a road segment. It is possible for some GPS locations to be matched to a wrong road segment and therefore, as shown in Figure 2.2(a), the sequence of map-matched road segments may not form a feasible path. (Such a path is generally not required by the systems that use basic map matching.) Starting with the advent of GPS-based vehicle navigation systems in the early 1990s, a large number of algorithms have been proposed for the problem of basic map matching. These algorithms range from methods that rely on geometrical and topological information of the road network (e.g. [52] and [53]) to algorithms that use more advanced techniques such as fuzzy logic (e.g. [54] and [55]). Quddus et al. [56] have presented a comprehensive review of more than 30 basic map matching algorithms published until 2006. A more recent survey of the literature on basic map matching has been presented by Hashemi and Karimi [57].

(a) Basic map matching: map-matched points may not lie along a connected path.

(b) Map matching with path inference.

Figure 2.2: Illustration of two forms of map matching (Base map: OpenStreetMap).

This thesis does *not* focus on the problem of basic map matching described above. Instead, it focusses on another form of map matching that aims to infer the actual path of the vehicle and ensures that all the map-matched road-segments lie along the inferred path. This form of map matching, which may be referred to as *map matching with path inference*, started receiving the attention of researchers a little more than a decade ago when large scale trajectory data started to become available. It is required in many trajectory-based applications such as the ones described in the previous section. Map matching with path inference is illustrated in Figure 2.2(b), where the thick dashed line indicate the inferred path. It needs to be clarified that successive map-matched points may not necessarily fall on directly connected road segments as shown in the example in figure. This is because map matching with path inference typically operates on temporally sparse location measurements. In cases where the successive map-matched road segments are apart from each other, the algorithm interpolates the path between them. In the remainder of this thesis, unless otherwise stated, the phrase ‘map matching’ refers to map matching with path inference.

Map matching may be performed in two modes, namely, offline and online. In offline mode, map matching is applied to an entire trajectory corresponding to a trip after the trip is completed. Offline map matching is used in some trajectory-based applications such as travel behaviour analysis, where immediate processing of data is not important. In online mode (also known as incremental mode), streaming trajectory data is map matched in a ‘live’ manner. However, this does not necessarily mean that each raw location in a trajectory is map matched instantaneously as soon as it arrives. Rather, online map matching typically involves a small latency with the algorithms generally requiring a few (but not all) subsequent locations to be examined before map matching a location. Time-critical applications such as trajectory-based traffic condition estimation make use of online map matching.

Map matching is one of the most widely researched problems in applied computing in recent times. A search in Google Scholar [58] for research articles

published in the last 10 years (2008-2017) with the phrase ‘map matching’ in the title returns more than 550 unique articles. A number of algorithms conforming to a variety of approaches have been proposed. The map matching algorithms proposed in the literature may be grouped into the following broad categories:

- 1) Geometrical map matching algorithms
- 2) Route-based map matching algorithms
- 3) Probabilistic map matching algorithms

Algorithms from the literature belonging to each of the above categories are critically examined in the following subsections. Only those algorithms that are applicable to the problem of inferring the travelled path from sparse and noisy trajectories are considered. Although most of the reviewed algorithms were originally proposed for map matching low-noise GPS trajectories, they are generally applicable, albeit with reduced effectiveness, to non-GPS trajectories with a higher degree of noise. Hence, the following descriptions of the algorithms are mostly kept general and not specific to any positioning technology. The terms ‘location measurements’ and ‘raw locations’ are used to refer to the timestamped coordinates present in the trajectories, irrespective of the positioning technology used to generate them.

2.2.1 Geometrical Map Matching Algorithms

Geometric map matching methods associate a vehicle’s trajectory to a path in the road network based on criteria such as spatial proximity or curve similarity. A simple and naïve way of inferring the travelled path from a sparse trajectory is to match each raw location to its nearest road segment and connect successive road segments using the shortest paths between them. However, multiple experimental evaluations have shown that adopting such an approach produces highly inaccurate results [59, 60]. Yang et al. [61] proposed a refinement to this naïve ‘nearest-road’ map matching method in order to improve its accuracy. In their method, a raw location is matched to the nearest road segment only if the latter is at least twice as close to the former compared to any other road segment.

Ambiguous raw locations typically situated near intersections and between parallel roads are handled differently based on a set of rules.

Zheng and Quddus [62] proposed an algorithm for map matching sparse GPS trajectories in an online manner based on a weighted combination of the following measures: (i) the distance between the location measurement and a candidate road segment, (ii) the difference between the road segment's direction and the GPS heading, (iii) the difference between the Euclidean distance between the previous and current location measurements and the distance along the shortest path between the previously map-matched point and the current on-road candidate point and (iv) the difference between the heading of the shortest path and the heading of the relevant portion of the trajectory.

It can be seen that the algorithms described above adopt a greedy approach in which a locally optimal choice is made at each step leading to a solution that may not be globally optimal. An alternative, non-greedy approach to map matching involves considering all possible candidate paths in the road network for a given trajectory and choosing the one that best matches the trajectory. Figure 2.3 illustrates the idea of candidate paths using a simple grid representing a road network. The dashed lines represent two of the several possible candidate paths corresponding to a trajectory consisting of location measurements o_1 , o_2 and o_3 . It could be seen that portions of each candidate path lie within the error range of each location measurement indicated by the dotted circles. The stars in the figure denote on-road candidate points corresponding to the location measurements in the trajectory.

One of the earliest non-greedy (i.e. globally optimal) map matching algorithms was proposed by Yin and Wolfson [63]. Their algorithm finds the best matching candidate path without explicitly keeping track of all possible candidate paths. The road network is represented as a weighted graph and a weight is assigned to each edge (i.e. road segment). The weight of a road segment depends on its proximity to the trajectory and the similarity between its heading and the heading of the corresponding portion of the trajectory. Subsequently,

Figure 2.3: Illustration of the concept of candidate paths for an input trajectory on a simple grid-like road network.

Dijkstra’s shortest path algorithm [64] is applied to the weighted graph to find the minimum-weight path, which is the best matching candidate path according to the geometric criteria used by the authors.

Brakatsoulas et al. [13] used the Fréchet distance [65], a measure of similarity between curves, to find the best candidate path corresponding to an input trajectory. Their solution includes a procedure for determining if there exists a path in the road network that is within a certain Fréchet distance of the trajectory. The binary search technique is applied to find the path with the minimum Fréchet distance to the trajectory.

Geometric information about the trajectory and the road segments is essential for map matching. However, treating map matching purely as a problem of matching shapes has its limitations. The accuracies of simple deterministic algorithms based entirely on geometric criteria degrade drastically when the sampling interval in the trajectory data is increased [63, 66]. Also, such algorithms are generally incapable of coping with non-GPS trajectories that have a higher degree of ambiguity. While the geometrical approach was favoured in the early years of map matching, most algorithms published in recent years adopt other approaches that are more robust to the imprecision in the trajectory data.

2.2.2 Route-Based Map Matching Algorithms

Generally, vehicles do not move along a road network in an aimless, random manner. Instead, they mostly travel along routes that are considered optimal according to some criteria preferred by the drivers. For example, a vehicle may travel along the shortest route, the fastest route (i.e. the one with the minimum travel time) or another route chosen based on a combination of multiple factors. Some map matching algorithms rely on this idea to infer a path corresponding to a vehicle's trajectory. Among all the feasible candidate paths corresponding to a trajectory, such algorithms choose a candidate path that qualifies as an optimal route according to some criteria. Algorithms that follow this approach are referred to as route-based map matching algorithms.

Griffin et al. [67] proposed a solution in which a subset of the raw locations in a trajectory, selected using the Douglas-Peucker polyline simplification algorithm [68], are treated as waypoints and sent to a web-based routing service that associates each waypoint with a nearby road segment and returns the best route passing through those road segments. However, the routing service may associate some of the selected raw locations to wrong road segments, thereby causing the returned route to be incorrect. Therefore, a refinement step based on a set of rules is used to identify and remove problematic waypoints. Subsequently, the routing service is invoked again with the reduced set of waypoints.

Rahmani and Koutsopoulos [69] used a multistage *candidate graph* such that the nodes in each stage represent the on-road candidate points of a raw location in the trajectory. An edge that connects two candidate points in successive stages represents the shortest paths between them in the road network. A set of candidate paths in the candidate graph is generated by computing the shortest paths (according to a generalized cost) from each of the nodes in the first stage to each of the nodes in the last stage. The algorithm chooses the best among the candidate paths in a deterministic manner as follows: First, a set of paths with travel times below a threshold is formed. Subsequently,

the path from this set that has the minimum travel distance is chosen. This map matching algorithm was subsequently improved by Sarlas [70], who used a probabilistic route choice model to choose the most likely path among the candidate paths.

A method for finding the shortest path among all the candidate paths without explicitly constructing the candidate graph was proposed by Eisner et al. [11]. At each stage, a virtual node is added with outgoing edges to each node representing a candidate point at that stage. The length of each such edge is set equal to the actual shortest path up to that node. (This length is set to zero for the nodes in the first stage.) Subsequently, the shortest paths are extended to the nodes in the next stage through a one-to-many shortest path computation.

Schulze et al. [27] proposed a route-based map matching algorithm for highly imprecise trajectories made up of the locations of connected cellular base stations. For each pair of consecutive locations, the algorithm constructs a transition graph that covers the road segments within an area encompassing the two locations. The transition graphs are then joined together to form a search graph. The shortest paths in the search graph between a set of nodes closest to the first location and a set of nodes closest to the last one are computed. Subsequently, the shortest among those paths is selected as the output.

The above methods choose the candidate path that qualifies as the best route without considering its proximity and similarity to the input trajectory. This may sometimes result in a situation where a path that is highly consistent with the trajectory is rejected in favour of another that is considered a better route based on some criteria. It is, of course, not uncommon for a vehicle to travel along a non-optimal route. Therefore, some route-based map matching algorithms ensure that the proximity and similarity of a path to the trajectory are factored in while choosing the best candidate path.

Giovannini [71] presented a solution in which a variable number of best candidate road segments corresponding to each location measurement are

retained on the basis of a positioning probability measure that depends on geometric criteria such as proximity and heading similarity. Subsequently, the shortest paths between each pair of candidate road segments belonging to two consecutive time steps are determined. Finally, the global optimal path from the first time step to the last one is found based on the minimum-travel-time criteria.

Srivatsa et al. [72] proposed a map matching algorithm that first finds k shortest paths between the end points of the trajectory. For each of these k candidate paths, a proximity score, expressed as the sum of the minimum distances from each raw location in the trajectory to the candidate path, is calculated. The path that is closest to the trajectory is selected as the output. A similar approach was adopted by Kuijpers et al. [73], who computed k shortest paths within an area encompassing all the location measurements in the trajectory and selected the path closest to the trajectory based on the summation of ordinal proximity weights assigned to each road segment in the path.

Unlike the above algorithms that aim to find a globally optimal route corresponding to the entire trajectory, Miwa et al. [74] proposed a route-based online map matching algorithm that operates on the trajectory in a piecewise manner. Their algorithm considers three location measurements in the trajectory at a time in order to map match the second (i.e. the middle) raw location. Using a shortest path algorithm, several candidate paths that begin and end near the first and the third raw locations, respectively, are computed. The shortest path computation considers not only the length of each road segment, but also its proximity to the trajectory. Subsequently, a logit route choice model is used to select the best candidate path. The route choice model considers a number of path attributes such as length, travel time, number of road class changes, number of turns and the proportion of expressways.

It is intuitive that algorithms that consider both route optimality and proximity to the trajectory are better suited for noisier non-GPS trajectory data, the focus of this thesis, compared to algorithms that rely solely on either of these criteria. However, given the uncertainties inherent in the location measurements

and the choice of routes, the approach of choosing the best path, spanning several location measurements, based on deterministic criteria leaves room for improvement. The weakness of this approach becomes more apparent when applied to highly imprecise trajectories that result in a large number of feasible candidate paths, as indicated by the results in [27].

2.2.3 Probabilistic Map Matching Algorithms

Most of the state-of-the-art map matching algorithms model the uncertainties caused by the location measurement errors and the low sampling rates as probability distributions and use a probabilistic framework for finding the most likely path given an input trajectory. A vast majority of the probabilistic map matching algorithms in the literature conform to a Probabilistic Graphical Model (PGM) [75] framework. A PGM is basically a graph representation of the dependencies between random variables. There are two broad categories of PGMs, namely, Bayesian networks that are modelled as directed acyclic graphs and Markov random fields that are modelled as undirected graphs. Hidden Markov Model (HMM) belonging to the former category and Conditional Random Field (CRF) belonging to the latter have been applied for the map matching problem. Solutions proposed in the literature based on HMM, CRF and other probabilistic approaches, respectively, are reviewed below.

HMM-Based Map Matching Algorithms

Hidden Markov models [76] are widely used for modelling processes with unknown (hidden) *states* that generate a temporal sequence of *observations*. In a HMM, time and state space are treated as discrete. There are two important assumptions in a HMM: First, the probability of an observation is assumed to depend only on the state from which it was generated and not on any other states or observations. Second, the process is assumed to satisfy the Markov property according to which the probability of a particular state depends only on the previous state and not on the states before the previous state. There are two

probability distributions defined in a HMM: *emission probability* and *transition probability*. The emission probability distribution expresses the probability of an observation being generated from a state. The transition probability distribution represents the probability of moving to a state given the previous state. Typically, several plausible state sequences exist for a given observation sequence. The most likely state sequence in a HMM can be found using the Viterbi algorithm [77], a dynamic programming algorithm described in the next chapter.

In the map matching context, the movement of a vehicle along a path that produced the location measurements at discrete time steps can be modelled as a HMM in the following manner [78]:

- The observations are the raw location measurements of the vehicle.
- The hidden states are the unknown on-road points on which the vehicle was when the location measurements were made.
- The emission probability of a location measurement for a given on-road candidate point represents the probability of observing the location measurement if the vehicle is on the candidate point.
- The transition probability represents the probability of the vehicle moving from a given candidate point at a time step to a candidate point at the next time step.

The HMM-based map matching framework is illustrated using a simple example in Figure 2.4, where the input trajectory is made up of a sequence of location measurements (i.e. observations) o_1, o_2 and o_3 corresponding to time steps 1, 2 and 3, respectively. The error region of each location measurement is marked by the dashed circles in Figure 2.4(a). The squares indicate the candidate points (i.e. states), which are points closest to the location measurement on each of the road segments that lie either partially or fully within the error region. The k^{th} state at time step t is denoted as $s_{t,k}$. Figure 2.4(b) shows a HMM representation of the given example. The dashed arrowed lines denote the

(a) Location measurements and candidate points on a simple road network.

(b) HMM representation of the above example where solid and dashed edges are associated with transition and emission probabilities, respectively.

Figure 2.4: Illustration of the HMM framework for map matching.

conditional dependency of the observations on the states defined by the emission probability distribution. The solid arrowed lines indicate the conditional dependency of the states at a time step on the states at the previous time step defined by the transition probability distribution.

At least 20 map matching algorithms broadly conforming to the above-described HMM framework have been proposed in the last 10 years. They mostly

differ in the definition of the emission and transition probability distributions.

Table 2.1 summarises a selection of HMM-based map matching algorithms.

Algorithm	Year	Emission probability factors	Transition probability factors
Krumm et al. [59]	2007	Distance between the location measurement and candidate point (<i>'Distance to location'</i>).	Difference between actual and estimated travel times.
Newson and Krumm [79]	2009	Distance to location.	Difference between inter-candidate-point driving distance and inter-location distance (<i>'Distance difference'</i>).
Lou et al. [80]	2009	Distance to location.	Ratio of inter-location distance and inter-candidate-point driving distance. Similarity between the actual average speed and road speed limits.
Thiagarajan et al. [81]	2009	Distance to location.	Equal transition probability for all directly connected roads. (Gap between successive raw locations are interpolated.)
Raymond et al. [82]	2012	Distance to location.	Shortest path distance between candidate points.
Goh et al. [83]	2012	Distance to location. Road width. Difference between measured speed and road speed limit.	Difference between inter-candidate-point driving distance and deduced travel distance based on elapsed time. Momentum change (to penalize paths with many abrupt turns).
Song et al. [84]	2012	Distance to location. Road speed limit (Travel on major roads more likely.)	Distance difference measure.
Mohamed et al. [85]	2014	Distance to location. Similarity between directions of trajectory and road segment.	Distance difference measure. A bias to favour staying on the same road
Asgari et al. [29]	2016	Distance to location.	The type, length and average speed of each edge in the shortest path between candidate points.

Table 2.1: A summary of selected HMM-based map matching algorithms.

One of the earliest HMM-based solutions for finding the most likely path corresponding to a trajectory was proposed by Krumm et al. [59] in 2007. They modelled the emission probability distribution as a function of the distance between the location measurement and an on-road candidate point based on the assumption that the location measurement error follows a zero-mean two-dimensional Gaussian distribution. (All the algorithms reviewed below rely on this assumption to model the emission probability distribution unless otherwise stated.) Krumm et al. calculated the transition probability between two candidate points based on the difference between the actual elapsed time and the estimated travel time between them. They modelled this difference as a Gaussian distribution.

Newson and Krumm [79] argued that the approach of assigning transition probabilities based on travel time differences adopted by Krumm et al. [59] is sensitive to traffic conditions and therefore less reliable. Instead, they compared the shortest driving distance between a pair of candidate points with the straight-line distance between the corresponding location measurements. The difference between the above two distances, known as the *distance difference* measure, was modelled as an exponential distribution. This work by Newson and Krumm, published in 2009, remains very influential in the area of HMM-based map matching, with several of the subsequent algorithms using it as a baseline and seeking to improve it.

Raymond et al. [82] pointed out that the model proposed by Newson and Krumm violates the independence assumptions in a HMM because the probability of transitioning to a state not only depends on the previous state as it should, but also on the current and previous observations. They proposed a modification where the transition probabilities depend solely on the shortest path distance between the candidate points. In a subsequent paper, Osogami and Raymond [86] computed the transition probabilities between candidate states based on the cost of the optimal path between them that minimizes the convex combination of the travel distance and the number of turns. This idea was further

extended by Yin et al. [87] who also included the proximity of the road segments to the trajectory as a cost measure in the optimal path computation.

The transition probability model of Newson and Krumm has been criticised by some researchers [84, 88] for another weakness: At longer sampling intervals, the distance difference measure typically becomes large, causing the transition probability to be very small. This creates a strong bias in favour of shorter, direct paths even if the location measurements are closer to another circuitous path. Obviously, this behaviour degrades the accuracy of the algorithm at longer sampling intervals. Song et al. [84] addressed this issue by empirically determining the optimal values of the parameter of the transition probability distribution for different sampling intervals. Wei et al. [88], in their HMM-like framework, balanced the equivalents of the emission and transition probabilities by considering the sampling interval in the computation of the optimal state sequence.

Goh et al. [83] adopted a machine learning approach for computing transition probabilities in their HMM-based map matching algorithm. They assigned the transition probability from one candidate state to another by considering the following features of the shortest path between them: (i) the difference between the driving distance of the path and the deduced travelling distance based on the elapsed time and (ii) a measure that seeks to penalize paths with many abrupt turns. A Support Vector Machine (SVM) classifier, trained using labelled instances, was used to give the probability that a given input feature vector represents a ‘correct’ transition. Li et al. [89] presented a solution in which the parameters of the transition probability model are learned from a training dataset. In their model, the transition probability between two on-road candidate points is obtained by summing over the probabilities of *all* paths that exist between them. The probability of any such path depends on the probability of the turns involved in it and the probability of the path being traversed in the time elapsed between the location measurements.

Thiagarajan et al. [81] proposed a HMM-based map matching method that differs from the solutions described above in one key aspect: it does not require the shortest path (or paths) between a pair of candidate points to be computed in order to estimate the transition probability between them. Instead, the gap between successive location measurements is first interpolated along the straight line between them by assuming a constant speed. Then, the interpolated points are fed as the input for map matching. The transition probabilities are assigned based on the assumption that it is equally probable for a vehicle to remain on the same road segment or transition to another road segment directly connected to it. This approach was later extended by Szwed and Pekala [90], who allowed transitions from one road segment to another with equal probability as long as they are linked by a sequence of not more than a certain number of road segments.

Most of the research on HMM-based map matching has been directed towards improving the transition probability model while adopting the standard approach of modelling the emission probability using a Gaussian function of the distance between a location observation and a candidate point. However, some emission probability models that consider additional information have been proposed. Song et al. [84] assumed that a vehicle is more likely to travel on major roads with higher speed limits and therefore factored in the speed limits of the candidate road segments while calculating the emission probability. Goh et al. [83] augmented the Gaussian function used to model the location measurement error by accounting for the road width. In addition, they used a penalty function based on the difference between the instantaneous speed present in a GPS measurement and the speed limits of the candidate road segments. (However, instantaneous speed information is not always available in trajectory data.) Oran and Jaillet [91] argued against relying on the closest distance between a raw location and a candidate road segment for calculating the emission probability. Instead, they used a line-integral based cumulative measure of the proximity of the raw location to the whole road segment.

A few map matching algorithms in the literature broadly conform to the HMM framework, although the authors do not explicitly describe them as such [80, 88, 92-94]. They typically use a weight function that is analogous to a combination of the emission and transition probabilities. The optimal sequence of candidate road segments corresponding to the input trajectory is found using a dynamic programming approach. The most influential among such algorithms is the one named ST-Matching proposed by Lou et al. [80], in which the transition probability consists of a spatial component and a temporal component. The spatial component is given as the ratio of the Euclidean distance between successive location measurements and the length of the shortest path between a corresponding pair of candidate points. The temporal component is obtained by comparing the average speed along the shortest path with the typical speeds of the roads along it. An improvement to this solution was proposed by Yuan et al. [92] based on the idea that the matching of a location measurement to an on-road point can be constructively influenced by its neighbouring location measurements. They proposed an interactive voting scheme that captures the mutual influences among the location measurements in order to improve the weight function used by the ST-Matching method.

The works described above focussed mainly on GPS trajectory data and the development of the proposed solutions appears to be influenced by the characteristics of such data. Their effectiveness for noisier non-GPS trajectories has not been adequately evaluated. In recent years, a few researchers have proposed HMM-based map matching algorithms that are specifically targeted towards highly imprecise non-GPS trajectories.

Mohamed et al. [85] focussed on the task of map matching coarse-grained trajectories generated by cellular network positioning. The input trajectory is first passed through a preprocessing stage for filtering outliers and interpolating large gaps before being map matched in an incremental manner. The standard emission probability model is used with two refinements: First, the standard deviation of the Gaussian distribution is dynamically estimated using the average

error of a certain number of previously matched location measurements. Second, road segments that run in the opposite direction of the input trajectory are assigned a lower emission probability. As for transition probability, the widely-followed model proposed in [79] is used with one change: To reduce abrupt transitions caused by noisy locations in the trajectory, the transition probability is biased to favour staying on the same road segment.

Lv et al. [6] proposed a map matching algorithm for cellular-positioning based trajectories using a HMM similar to the one previously proposed in [81] described earlier. Asgari et al. [29] proposed a HMM-based algorithm that maps cellular trajectories to the most likely path in a multimodal transportation network consisting of a road network and a metro/train network. In this algorithm, the emission probability for a state is expressed as a decreasing function of the distance between itself and the observed base station. The transition probability between two states depends on the type, length and average speed of each edge in the shortest path between the states.

A recent work by Algizawy et al. [30] applied HMM in a rather unconventional way to infer the path corresponding to an imprecise trajectory made up of cellular base station locations. In contrast to almost all existing solutions, the authors consider one common set of states for all time steps. However, they limit the state space by considering only the road segments that lie *at* the cell boundaries as states. Also, transitions are permitted only between states that are within a certain distance of each other. The transition probability depends on the shortest driving distance between the states. The emission probability for a given state and an observed base station location is defined as the ratio of the distance between them to the sum of the distances from the state to all other base stations that lie within a certain range. There are a few other solutions in the literature that use HMM for map matching cellular-positioning based trajectories, but they utilize additional data from the smartphone's inertial sensors such as accelerometer and magnetometer (e.g. [18] and [95]).

HMM is by far the most commonly followed approach for map matching imprecise trajectories and is widely regarded as the state of the art. However, HMM-based map matching has two well-known drawbacks. The first drawback is the inability to model complex, long-range dependencies spanning more than two time steps. This is due to the independence assumptions in a HMM that permit only local dependencies between states and observations. The second drawback is known as the *label bias* problem. In an ideal HMM, the probabilities of all transitions emanating from a state should sum to 1. This results in a bias towards states with fewer valid transitions to states at the next time step. The implications of these drawbacks for the map matching problem and ways of overcoming them are discussed later in this thesis. Despite the substantial volume of literature on HMM-based map matching published over the last decade, it remains an active research area with several papers published in recent months (e.g. [30], [96], [97] and [98]).

CRF-based Map Matching Algorithms

A Conditional Random Field (CRF) [99, 100] is an undirected probabilistic graphical model that directly represents the probabilities of possible state sequences given an observation sequence. Figure 2.5 shows the graphical model corresponding to a type of CRF known as a linear-chain CRF [100]. The unshaded nodes in the graph denote the state variables (also known as labels) and the shaded nodes represent the observed variables, respectively, at different time steps. Each undirected link in the graph indicates that the pair of nodes connected by it are part of a local function, known as a feature function.

Figure 2.5: A linear-chain conditional random field where unshaded and shaded nodes represent state and observed variables, respectively.

Given a sequence of states $S = \{s_1, s_2, \dots, s_T\}$ and a sequence of observations $O = \{o_1, o_2, \dots, o_T\}$ corresponding to time steps $1, 2, \dots, T$, the probability of the state sequence conditioned on the observation sequence is given by

$$p(S|O) = \frac{1}{Z(O)} \prod_{t=1}^T \exp \left\{ \sum_{k=1}^K \theta_k f_k(s_t, s_{t-1}, o_t) \right\} \quad (2.1)$$

where $Z(O)$ is an input-specific normalizing term, f_k is one among K feature functions and θ_k is the weight corresponding to f_k . The feature functions are commonly divided into two groups: state feature functions and transition feature functions, broadly analogous to the emission and transition probabilities in a HMM, respectively. However, as it is not required to normalize the probabilities of transitions from each state, CRFs do not suffer from the label bias problem seen in HMMs. The weights of the feature functions are commonly learned from data using maximum likelihood estimation. The state sequence in a linear-chain CRF with maximum probability conditioned on the observation sequence can be found using the Viterbi algorithm [77].

Hunter et al. [60] used a linear-chain CRF to model the problem of inferring the travelled path from sparsely sampled location measurements. They used a zero-mean Gaussian function of the distance between a location measurement and an on-road candidate point as the equivalent of a state feature function. Their transition feature function depends on a number of features such as length of the shortest path between the states considered, estimated travel time, number of traffic signals, number of turns and speed limits. The model parameters are determined through a training procedure that applies standard machine learning techniques on trajectory data.

Xu et al. [101] proposed a CRF-based map matching method in which the state feature function is the same as in the solution proposed by Hunter et al [60]. They used two transition feature functions based on the following: (i) the spatial consistency between the Euclidean distance between two consecutive location

measurements and the shortest driving distance between two corresponding on-road candidate points and (ii) the temporal consistency between the elapsed travel time between two consecutive location measurements and the expected travel time between two corresponding on-road candidate points. Subsequently, the transition feature functions are complemented by adding another function that quantifies the preference of a driver for a sub-path between consecutive candidate points. The latter function is calculated using information mined from labelled historical trajectories.

Yang and Meng [102] investigated the problem of optimal feature selection in CRFs, as using more features than necessary may increase the risk of over-fitting and increase the computation cost. They considered a number of state and transition features and used the L_1 regularization technique to reduce the weights of irrelevant features to zero during training. Their evaluations on trajectory data obtained from a taxi fleet showed that the following three features were most relevant: (i) the distance between an on-road candidate point and a location measurement, (ii) the difference between the length of a candidate sub-path and the distance between corresponding location measurements and (iii) the degree of road class changes along a candidate sub-path.

Recently, Liu et al. [103] have proposed a model named ST-CRF, whose feature functions include the spatial and temporal analysis functions originally proposed in the ST-Matching algorithm [80]. In addition, they also included two additional feature functions based on the following: (i) the closeness of the ‘middle point’ that lies in between two location measurements and the middle point between two candidate points and (ii) the consistency between the driving direction and the direction indicated by the two location measurements.

In general, CRFs are not limited by the independence assumptions followed in HMMs. It is therefore possible to model long-range context dependencies involving states and observations spanning more than two time steps using a higher-order CRF. However, as noted by Xu et al. [101], the exact inference of parameters in higher-order CRFs is intractable. All the existing CRF-

based map matching methods use a linear-chain CRF model, whose structure (i.e. its factorisation) is similar to the local dependency structure in a HMM, except that in some of the CRF-based algorithms, the transition feature function also depends on the corresponding pair of observations. (However, some of the HMM-based algorithms such as [79] and [85] also model the transition probabilities in this manner, thus violating the assumption of independence between transitions and observations.) It has been shown that for a linear-chain CRF whose transition feature functions do not depend on observations, it is possible to derive a set of HMM parameters such that the two models are exactly equivalent [104]. It can be seen from the above that the existing CRF-based map matching algorithms suffer from the same inability as HMM-based solutions to utilize contextual information about past and future states in the transition model.

Other Probabilistic Map Matching Algorithms

A small number of researchers have proposed probabilistic methods that do not conform to a probabilistic graphical model framework such as HMM and CRF. Bierlaire et al. [105] presented an algorithm that computes the probability of a sequence of GPS locations given a candidate path. This computation involves two factors: (i) a model of the location measurement error assuming a Rayleigh distribution and (ii) a model of the travel speed along a path assuming a mixture of an exponential distribution and a log normal distribution. The probabilities are computed using integrals that span all positions along portions of the given candidate path that lie within a certain range of the location measurements. A path generation algorithm is used to generate multiple candidate paths for the input trajectory. Unlike most map matching algorithms that output the best candidate path, the solution proposed by Bierlaire et al. output a set of potential paths along with the likelihood of the trajectory for each path.

Westgate [106] proposed a map matching algorithm that has some similarity to the work by Bierlaire et al. He used a Markov chain Monte Carlo method to generate a sample of paths for an input trajectory and adopted a Bayesian approach to estimate the posterior probabilities of each candidate path.

The approach utilizes the following probabilistic models: (i) two exponential distributions to model the persistent bias and the independent component, respectively, in the GPS measurement error, (ii) A multinomial logit choice model to incorporate a preference for paths with shorter travel time and (iii) a log-normal distribution of the travel time between successive location measurements.

The approach of generating a set of candidate paths and computing their likelihood followed by the above works [105, 106] is suitable for some specific applications such as route choice modelling. However, it has not been widely adopted as a general solution for the map matching problem, probably because the computations involved in it are more complex compared to the simpler formulation of algorithms based on probabilistic graphical models.

2.3 Discussion and Summary

There is an increasing awareness about the huge potential for improving urban mobility by exploiting trajectory data. With copious amounts of trajectory data being generated by millions of smartphone users every day, the problem of map matching imprecise trajectories has received strong attention from the research community in recent years. As it can be seen from the review presented above, a large number of solutions have been proposed for the map matching problem. However, the problem continues to be actively researched as there is a clear need to develop algorithms that are more robust to high degrees of imprecision in the trajectory data.

The published map matching algorithms fall into three broad categories, namely, geometrical, route-based and probabilistic. In almost all the papers on map matching, the proposed algorithm is compared with a baseline algorithm and some improvement is reported. However, it is rather difficult to draw firm conclusions from the reported results because of the lack of uniformity in the test datasets, choice of algorithm parameters and evaluation metrics. However, based

on a careful study of the literature, some broad conclusions can be drawn about the state of the art:

- The results of comparative analyses performed by several researchers clearly indicate that probabilistic algorithms achieve substantially higher accuracy with sparse and noisy trajectory data compared to deterministic approaches based on geometric or route-optimality based criteria [30, 60, 66, 73]. This finding is consistent with the fact that a vast majority of the map matching algorithms proposed in the last 5 years are based on a probabilistic graphical model framework.
- Among map matching algorithms based on probabilistic graphical models, HMM-based algorithms are significantly more influential than CRF-based ones in terms of number of published solutions as well as citations. While the linear-chain CRF model used for map matching is similar to a HMM in many aspects, some researchers [60, 103] have pointed out that the former does not suffer from the label bias problem as the latter. However, workarounds have been proposed in the literature for eliminating label bias in HMM-based map matching [81].
- Almost all the algorithms proposed for map matching highly noisy non-GPS trajectories rely on HMM [6, 18, 29, 30, 85]. The only known solution that adopted a deterministic route-based approach for map matching non-GPS trajectories [27] has been found to perform poorly compared to a HMM-based algorithm [30]. It is evident that the HMM framework lends well for modelling the uncertainties corresponding to the measurement and sampling errors in the trajectory data.

While HMM-based map matching is widely regarded as the state of the art, there are a number of issues in this area that needs further research. Based on a comprehensive study of the literature, the following focus areas have been identified for further exploration in this thesis:

- **Improved probabilistic models for sparse, non-GPS trajectories:** Most of the emission and transition probability models in the existing HMM-based map matching algorithms were designed for and tested with GPS trajectories. These models do not perform adequately when applied to trajectories generated by alternative positioning technologies with much higher measurement noise. This issue has started receiving research attention only in recent years. There exists a need to develop probabilistic models tailored for temporally sparse and highly noisy trajectories. This task is undertaken in the next chapter.
- **Integrating route-preference criteria with the HMM framework:** As seen earlier, HMM (as well as linear-chain CRF) models do not permit long-range context dependencies spanning more than two time steps to be modelled. While paths travelled by vehicles are generally influenced by the drivers' choices of optimal routes, the HMM framework is incapable of considering route optimality while finding the most likely path corresponding to a trajectory. It appears likely that complementing HMM-based map matching algorithms with some method for evaluating the preferability of candidate paths would lead to positive results. This issue is dealt with in Chapter 4.
- **Optimizations for efficient online map matching:** Map matching algorithms based on probabilistic graphical models involve a substantial amount of computation, which makes it hard to use them for applications that involve on-the-fly map matching of large scale trajectory data. This problem becomes especially difficult when highly-noisy data such as cellular-positioning based trajectories are used, as they require a large state space to be considered for each location observation. The output latency of the algorithms is also an important consideration in real-time applications that require timely map matching. There is a strong need to develop optimization techniques to improve the computational efficiency and timeliness of the map matching process. Reducing the computation

time and/or output latency of map matching algorithms have been the focus of several papers (e.g. [107], [108], [84] and [109]). A discussion of such works is provided in Chapter 5, which deals with this issue in detail.

Chapter 3

A Probabilistic Map Matching Algorithm

As discussed in the last chapter, hidden Markov models are generally considered as the most effective framework for solving the problem of map matching imprecise trajectories, especially those generated by noisy non-GPS positioning technologies. In this chapter, a HMM-based map matching algorithm, which includes new emission and transition probability models tailored for noisy and sparse trajectories, is proposed. Subsequently, the proposed algorithm is evaluated with real as well as synthetic trajectory data under varying levels of measurement noise and temporal sparsity. The solution described in this chapter and early versions of the results have been published in [32] and [33].

3.1 Problem Statement

A *location measurement* (or *location observation*) o_t consists of the measured latitude, longitude and timestamp corresponding to a mobile user's location at time step t . A *trajectory* $o_{1:N} = \{o_1, \dots, o_N\}$ is a time-ordered sequence of N location observations.

A *road network* is a directed graph $G = (V, E)$, where V is a set of nodes representing intersections or endpoints of the road segments and E is a set of edges representing road segments. The geometry of each road segment's centerline is stored as a sequence of latitude-longitude pairs representing a

piecewise linear curve. Each road segment has a number of attributes including road class, length and free-flow travel time.

A *path* p between nodes u and v is a sequence of connected road segments (i.e. edges) $e_1, \dots, e_n$ such that u is the start node of e_1 and v is the end node of e_n . In this work, a path does not necessarily start and end at nodes. It could start or end at any point that lies along the centerline of any road segment.

Given an observed trajectory $o_{1:N} = \{o_1, \dots, o_N\}$ and a road network G , the map-matching problem is to find a path p in G corresponding to $o_{1:N}$.

3.2 The HMM-Based Map Matching Algorithm

The proposed solution conforms to the HMM framework described in Section 2.2.3 in which the inaccurate location measurements are treated as observations and the corresponding on-road candidate points are treated as hidden states. In theory, all points along the road network should be considered as hidden states. However, in practice, it is sufficient to consider only those road segments that lie within a fixed error range of the observed location. Furthermore, for each such road segment, only the point along its centerline that is closest to the measured location is considered as a state. Figure 3.1(a) shows a trajectory consisting of a sequence of observations $\{o_1, o_2, o_3\}$ and the states corresponding to them on a simplified road network. The k^{th} state at time step t is denoted as $s_{t,k}$. For clarity, each observation is shown to have only two possible states.

It is convenient to visualise the states in a HMM and the transitions between them as a trellis graph. Figure 3.1(b) shows the HMM trellis graph representation for the example above. Each node in the trellis graph represents a state and each edge in it represents a transition between a pair of states. As it can be seen, there are several feasible state sequences from the first time step to the last. (In reality, the number of feasible state sequences is much larger than in this simplified example.)

(a) An example showing observations and states.

(b) HMM trellis graph, where edges represent transitions between states.

Figure 3.1: HMM representation for map matching.

Let $S_{1:T} = \{s'_1, s'_2, \dots, s'_T\}$ be a state sequence such that s'_1 is one of the states at time step 1, s'_2 is one of the states at time step 2, and so on. Let $O_{1:T} = \{o_1, o_2, \dots, o_T\}$ be the observation sequence. The joint probability of the state sequence $S_{1:T}$ and the observation sequence $O_{1:T}$ is expressed as

$$P(S_{1:T}, O_{1:T}) = P(o_1 | s'_1) \prod_{t=2}^T P(o_t | s'_t) P(s'_t | s'_{t-1}) \quad (3.1)$$

where $P(o_t|s'_t)$ denotes the emission probability, which is the probability of the observation o_t being generated from state s'_t , and $P(s'_t|s'_{t-1})$ is the probability of transitioning from state s'_{t-1} to state s'_t . The most likely state sequence in a HMM is the one with the highest joint probability. It is found by the Viterbi algorithm, which is based on dynamic programming. The following subsections describe the details of the proposed HMM-based map matching algorithm, namely, the emission probability model, the transition probability model, the method of finding the most likely state sequence in an online manner and the procedure for estimating the model parameters.

3.2.1 Emission Probability Model

This work deals with coarse-grained location observations generated by smartphones using alternative positioning technologies that are preferred over GPS-based positioning due to a variety of reasons such as higher energy efficiency, better coverage and privacy preservation. In such a scenario, the location of a smartphone is typically estimated through one of the below two modes.

- *Cellular network positioning*: When both the GPS and Wi-Fi functions are disabled, a smartphone determines its location solely based on the cellular base stations from which it receives signals [14]. While this positioning mode is the most energy efficient, it is also the most inaccurate, with the average error in the order of several hundred meters.
- *Wi-Fi and cellular network positioning*: When the Wi-Fi function is enabled in a smartphone, its location is determined using a fingerprint comprising of both Wi-Fi and cellular signal information [15]. In most built-up areas where Wi-Fi signals are typically available from several access points, this mode of positioning has better accuracy, with the errors mostly less than 100 m. However, in areas without any Wi-Fi signals, the phone's location is determined based on cellular base station information, resulting in much higher errors.

The emission probability models for each of the above positioning modes are described below.

Emission Probability Model for Cellular Network Positioning

In general, the probability of a state that represents an on-road candidate point generating a location observation depends on the distance between the on-road candidate point and the observed location. It is intuitive that states closer to the observed location have a higher emission probability compared to those farther from it. In the case of the true state, the distance between itself and the observed location is the location measurement error. Assuming that the location measurement errors conform to a zero-mean Gaussian distribution in two-dimensional space, the emission probability for a state $s_{t,k}$ and a location observation o_t , is given as

$$P(o_t | s_{t,k}) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{g(o_t, s_{t,k})^2}{2\sigma^2}} \quad (3.2)$$

where σ is the empirically-estimated standard deviation of the measurement error and $g(o_t, s_{t,k})$ is the great-circle distance [110], which is the shortest distance along the surface of the earth, between o_t and $s_{t,k}$. Points at low latitudes that are separated by short distances can be treated as being on a two-dimensional plane and in such cases, the Euclidean distance can be used as an approximation of the great-circle distance. This approximation is used in this work as it reduces the amount of computation required.

Emission Probability Model for Wi-Fi and Cellular Network Positioning

It is assumed that the measurement errors from Wi-Fi and cellular network positioning systems, respectively, follow two different zero-mean Gaussian distributions. When both of these positioning systems are used together, the location measurements originate from either of the two distributions depending on the availability of Wi-Fi signals. However, given a raw trajectory, it is generally not possible to say which of the individual measurements were generated by

considering Wi-Fi signals. Hence, the location measurement errors are modelled as a two-component zero-mean Gaussian mixture distribution. Based on this model, the emission probability for a state $s_{t,k}$ and a location observation o_t , is given as

$$P(o_t | s_{t,k}) = \frac{w_1}{\sigma_1 \sqrt{2\pi}} e^{-\frac{g(o_t, s_{t,k})^2}{2\sigma_1^2}} + \frac{w_2}{\sigma_2 \sqrt{2\pi}} e^{-\frac{g(o_t, s_{t,k})^2}{2\sigma_2^2}} \quad (3.3)$$

where w_i and σ_i are the weight and standard deviation, respectively of the i^{th} component distribution. The weights and standard deviations of the two component distributions can be estimated from a ground-truth annotated trajectory dataset with known measurement errors, as explained later in this chapter.

3.2.2 Transition Probability Model

The transition probability from a state $s_{t-1,j}$ at time step $t - 1$ to another state $s_{t,k}$ at time step t depends on the features of the optimal path between them. In this work, the path with the minimum free-flow travel time (based on the speed limits), found using Dijkstra's algorithm [64], is considered as the optimal path. Dijkstra's algorithm is capable of finding the shortest paths in a graph from a single source node to *all* the target nodes in a single run. In the current context, the minimum-travel-time paths from any state at time step $t - 1$ to all the states at time step t is found using a single run of Dijkstra's algorithm.

The transition probability model proposed below can be seen as an improvement over the influential model presented by Newson and Krumm [79] that has been adopted or adapted by several subsequent works on map matching. As described in Section 2.2.3, Newson and Krumm relied on the idea that transitions through circuitous paths are less likely compared to direct paths. They used the difference between the driving distance along a path and the great-circle distance between the current and previous location observations as a measure of the circuitousness of the path. This measure of circuitousness has a couple of

drawbacks. First, as pointed out in [82], it violates a key assumption in a standard HMM, namely, the independence of state transitions from observations. In the case of highly noisy location measurements, the great-circle distance between the observed locations may cause the measure of circuitousness to be significantly inaccurate. The second drawback relates to the fact that the transition probabilities computed based on the above measure may vary greatly for equally plausible transition paths depending on the sampling interval [88, 108].

A new measure of circuitousness that overcomes the above drawbacks is proposed. For the optimal path between states $s_{t-1,j}$ and $s_{t,k}$, the measure of circuitousness is defined as

$$y(s_{t-1,j}, s_{t,k}) = \frac{d(s_{t-1,j}, s_{t,k}) - g(s_{t-1,j}, s_{t,k})}{\Delta T} \quad (3.4)$$

where the functions d and g give the driving distance and the great-circle distance in meters, respectively, between the states and ΔT is the time interval between time steps $t - 1$ and t in seconds. The driving distance between two on-road points is the physical length of the optimal path between them.

In addition to the above measure of circuitousness, a measure of temporal implausibility is considered in order to assign low transition probabilities to paths that cannot be traversed within the time interval ΔT unless the vehicle travels at an unreasonably high speed. The measure of temporal implausibility for states $s_{t-1,j}$ and $s_{t,k}$ is defined as

$$z(s_{t-1,j}, s_{t,k}) = \frac{\max(f(s_{t-1,j}, s_{t,k}) - \Delta T, 0)}{\Delta T} \quad (3.5)$$

where $f(s_{t-1,j}, s_{t,k})$ is the free-flow travel time, in seconds, of the optimal path between states $s_{t-1,j}$ and $s_{t,k}$. It needs to be noted that the above measure will have a non-zero value only when the actual travel time (i.e. the elapsed time) is less than the expected free-flow travel time due to driving faster than the speed limits along the road segments. The cases where the actual travel time is more than the expected free-flow travel time are not considered to be temporally

implausible, as such occurrences are commonly observed due to traffic conditions forcing a vehicle to travel slower than the free-flow speed.

Based on analysis of ground-truth annotated trajectory data, the measure of circuitousness defined in Equation 3.4 and the measure of temporal implausibility defined in Equation 3.5 are modelled as exponential distributions. Based on the above discussion, the transition probability of moving from state $s_{t-1,j}$ to state $s_{t,k}$ is defined as

$$P(s_{t,k}|s_{t-1,j}) = \lambda_y e^{-\lambda_y y(s_{t-1,j}, s_{t,k})} \lambda_z e^{-\lambda_z z(s_{t-1,j}, s_{t,k})} \quad (3.6)$$

where λ_y and λ_z are parameters of the exponential distributions to be empirically determined.

The Label Bias Problem

In a HMM, transition probabilities are normalized so that the probabilities of all the transitions emanating from a state sum to 1. However, this results in an undesirable bias towards states with fewer actual transitions to states at the subsequent time step. (The phrase “fewer actual transitions” could be better understood by considering a state $s_{t,k}$ at time step t that is situated such that moving from $s_{t,k}$ to most states at time step $t + 1$ is possible only through highly circuitous and temporally implausible paths with almost zero transition probability.) Thiagarajan et al. [81] worked around this problem by introducing a dummy ‘dead-end’ state and assigning the transition probabilities in such a way that the probability of all the transitions from a state (including the transition to the dead-end state) sums to 1. This approach is adapted in this work as follows.

It needs to be recalled that in the proposed HMM, only those road segments that lie within the error range of a location observation are considered as states at that time step. Since transitions are independent of observations in a HMM, it is incorrect to assume that a vehicle will move from a state at a time step to only the states considered at the next time step. In theory, the vehicle is free to move to states outside the error range of the next location measurement. That

is, there is a probability of the vehicle moving to a state not considered at the next time step. All such unconsidered states can be abstracted as a dead-end state. Let $s_{t,k}$ be a state at time step t and $|S_{t+1}|$ be the number of states considered at time step $t + 1$. The probabilities of each transition emanating from $s_{t,k}$, given by Equation 3.6, are multiplied by the factor $1/|S_{t+1}|$. The transition probability from $s_{t,k}$ to the dead-end state at time step $t + 1$ is set so that the sum of all transition probabilities from $s_{t,k}$ equals 1. The dead-end state, being outside the error range of the location observation, has an emission probability of zero. A dead-end state at a time step can only transition to the dead-end state at the next time step with a probability of 1. It is not hard to see that the above workaround will have the same effect as not normalizing the transition probabilities in the original HMM without the dead-end states.

3.2.3 Determination of the Most Likely State Sequence

Given an observation sequence $o_{1:N} = \{o_1, \dots, o_N\}$ and a corresponding HMM trellis graph with N time steps, the Viterbi algorithm [77] computes the most likely sequence in the HMM using the following recurrence relations.

$$V_{1,k} = P(o_1|s_{1,k}) \quad (3.7)$$

$$V_{t,k} = P(o_t|s_{t,k}) \max_j \left(V_{t-1,j} P(s_{t,k}|s_{t-1,j}) \right), \quad 1 < t \leq N \quad (3.8)$$

In the above equations, $V_{t,k}$ denotes the joint probability of the most likely state sequence ending at state $s_{t,k}$ based on the observations $o_1, \dots, o_t$. It is also referred to as the Viterbi path probability. The index j in Equation 3.8 that maximizes $V_{t,k}$ is stored as a back pointer for state $s_{t,k}$. It points to the predecessor state $s_{t-1,j}$ of the state $s_{t,k}$ in the most likely sequence ending at the latter.

In its standard form, the Viterbi algorithm outputs the most likely state sequence from time step 1 to time step N after all the N observations are received and processed. In such a case, the state $s_{N,x}$ at time step N that has the highest joint probability is selected as the final state in the most likely sequence. From

that state, the complete sequence is traced backwards using the back pointers. In the context of map matching, the most-likely state sequence needs to be transformed into a path in the road network. Such a path is obtained by interpolating the gap between each pair of successive states (i.e. on-road points) in the most likely state sequence with the optimal path between them.

In time-critical applications that require online map matching of streaming trajectory data, it is not feasible to wait until all the location observations in a trajectory are received before applying the Viterbi algorithm as described above. In such situations, it is desirable to apply the Viterbi algorithm in an online manner such that parts of the most likely state sequence are generated from time to time without waiting for all the observations to be received. However, the partial state sequences generated in this manner, when joined together, should result in the same global most likely state sequence found by the standard Viterbi algorithm. Such partial sequences can be generated based on the transitive closure concept, which defines the reachability relations in a directed graph [III]. This method, which has been previously applied for map-matching in [83], checks for a convergence condition where there remains only one state at a time step that is reachable through a chain of back pointers from every state of a subsequent time step. This idea is illustrated in Figure 3.2, where the arrows denote the back pointers stored by the Viterbi algorithm. It can be seen that tracing backward from any of the states at time step 5 leads to the shaded state $s_{3,2}$ at time step 3. In such a case, it can be said that a convergence condition has occurred with $s_{3,2}$ being the convergence state. In the given example, the partial sequence up to the convergence state (i.e. $s_{1,3}, s_{2,2}, s_{3,2}$) can be released as output, as it cannot be altered by future observations received after time step 5. Thus, this online variant of the Viterbi algorithm outputs a part of the most likely state sequence each time a convergence state is found.

Figure 3.2: An example of the convergence condition in the Viterbi algorithm.

3.2.4 Methods for Estimation of Model Parameters

The procedures for estimating the parameters of the emission and transition probability models, respectively, are given below.

Parameter Estimation for Emission Probability Models

As seen in Section 3.2.1, two different emission probability models are used depending on the type of the input trajectory data. In the case of trajectory data generated by cellular network positioning or any other positioning technology whose measurement errors conform to a single zero-mean Gaussian distribution, the emission probability model is as given in Equation 3.2. The standard deviation of the Gaussian distribution can be estimated from a trajectory dataset where the ground truth location corresponding to each location measurement is known. In such a case, for each location measurement, the error is nothing but the distance between itself and the corresponding ground truth location. In line with the practice followed by several map-matching algorithms (e.g. [79], [83], and [112]), the standard deviation of the measurement error can be estimated using the median absolute deviation, which is a robust estimator resilient to outliers. If $\{g_1, \dots, g_n\}$ is the set of all measurement errors in the ground-truth annotated

trajectory dataset and if the errors are assumed to be Gaussian distributed in two-dimensional space with zero mean, the standard deviation can be estimated as follows [113]:

$$\sigma = 1.4826 \text{ median}_i(|g_i|) \quad (3.9)$$

If the location measurements in the trajectory data are generated using both Wi-Fi and cellular signals, the emission probability model is as given in Equation 3.3. The weight and standard deviation of each of the two components in the model can be estimated using the expectation-maximization (EM) algorithm [114].

Parameter Estimation for Transition Probability Model

The transition probability model defined Equation 3.6 involves parameters λ_y and λ_z that need to be estimated from data. λ_y and λ_z , respectively, define the distributions of the measure of circuitousness and the measure of temporal implausibility. As seen before, the proposed map matching algorithm computes the abovementioned measures for the sub-paths between every state at a time step to every state at the next time step. The distributions of these measures can be obtained by computing them for a sample of *true* sub-paths extracted from a ground-truth annotated trajectory dataset. As shown later in Section 3.4, the values of the measure of circuitousness and the measure of temporal implausibility, respectively, computed from a number of true sub-paths conform to exponential distributions. The rate parameters λ_y and λ_z of the exponential distributions can be estimated as the inverse of their respective means.

3.3 The Experimental Setup

The following subsections introduce the experimental setup used for evaluating the algorithmic techniques proposed in this thesis.

3.3.1 Trajectory Datasets

In this work, a number of trajectory datasets along with corresponding ground truth information have been used for evaluating the proposed algorithms and for estimating the parameters of the models used. They are described below.

Cellular Dataset

Trajectory data generated by cellular network positioning were recorded during taxi trips made in Singapore using an Android smartphone with the GPS and Wi-Fi functions disabled. For this purpose, an application that sends the cellular base station fingerprints to the Google location server and obtains the location estimates from the server through the Android location application program interface was used. This trajectory dataset, referred to as *cellular dataset*, consists of 21807 location measurements recorded mostly at the rate of one point per second during 20 taxi trips. (i.e. The dataset contains 20 trajectories.) The trips covered a total distance of 421 km with a total travel time of 8 hours and 9 minutes. The roads covered during the taxi trips are shown in black in Figure 3.3.

The ground truth information of a trajectory consists of the true on-road points corresponding to each location measurement in it and the true path (i.e. a sequence of connected road segments) travelled by the vehicle. For determining the ground truth information of the collected trajectories, GPS trajectory data were also recorded during the taxi trips at the rate of one sample per second using another smartphone. The GPS trajectories, whose measurement errors are typically in the order of a few meters, were first map-matched using the HMM-based algorithm described in Section 3.2. The output of map matching (i.e. the inferred path and the matched on-road points) were subjected to thorough manual inspection and correction with the aid of the knowledge about the actual paths travelled during the data collection exercise. While the ground truth paths thus obtained are 100% accurate, the true on-road points corresponding to the location measurements could be off by up to a few meters due to the unavoidable GPS error. However, this is a reasonable approximation considering the fact that

Figure 3.3: Roads travelled during the collection of cellular trajectories
(In black; Base map image: Google Maps).

the typical measurement error of non-GPS trajectories used in this work is one or more orders of magnitude higher than that of GPS trajectories.

Wi-Fi-and-Cellular Dataset

Another trajectory dataset containing 10 trajectories was created through a separate set of 10 taxi trips. The trajectories and their corresponding ground truth information were created in the manner described above, except that the Wi-Fi function was enabled in the smartphone during the data collection. In places where Wi-Fi signals are available, the smartphone application provided location measurements based on a fingerprint comprising of both Wi-Fi access point and cellular base station information. In places without Wi-Fi availability, locations were determined using only cellular base station information. This trajectory dataset, referred to as *Wi-Fi-and-cellular dataset*, has a total of 6414 location measurements. A total distance of 104 km was travelled during this data collection exercise. The total duration of the 10 trips was 2 hours and 39 minutes.

Synthetic Datasets

In order to evaluate the performance of map matching algorithms under different levels of measurement noise, a number of trajectory datasets with added

synthetic noise are used in this work. These datasets were created by adding random Gaussian noise to all the ground-truth on-road points corresponding to all the 30 trajectories in the above two trajectory datasets (i.e. Cellular dataset and Wi-Fi-and-cellular trajectories.) The standard deviation of the added Gaussian noise ranged from 100 m to 500 m, in steps of 100 m. Thus, five synthetic trajectory datasets corresponding to measurement noise levels (in terms of standard deviation) 100 m, 200 m, 300m, 400 m and 500 m have been used. Each *synthetic dataset* has 30 trajectories with known ground truth information.

Paths Dataset

A larger trajectory dataset with ground truth information is used for estimating path-related parameters such as the parameters of the transition probability model explained earlier. (This dataset is also used for estimating other path-related parameters in Chapter 4.) This dataset, referred to as *paths dataset*, was provided by Quantum Inventions [115], a Singapore-based ITS company. It contains 1000 GPS trajectories that were collected from anonymized users of a mobile application. The corresponding ground truth information was determined as previously described.

3.3.2 Estimation of Parameters from the Trajectory Datasets

The standard deviation, σ , of the Gaussian distribution of measurement errors in Equation 3.2 has been estimated from the known location measurement errors in the cellular dataset. (It is worth recalling that the ground truth information is known.) For the cellular dataset, the median and worst-case measurement errors are 258 m and 4652 m, respectively. The standard deviation has been estimated to be 382 m using the median absolute deviation estimator, as given in Equation 3.9.

The median and worst-case measurement errors, respectively, in the Wi-Fi-and-cellular dataset are 56 m and 8116 m. The weights, w_1 and w_2 , and the standard deviations, σ_1 and σ_2 , of the two-component Gaussian mixture model in Equation

3.3 have been estimated using the EM algorithm. The estimated values are given in Table 3.1.

Parameter	Component distribution 1	Component distribution 2
Weight	0.26	0.74
Standard deviation (m)	393.54	66.39

Table 3.1: Estimated parameters of the Gaussian mixture distribution

For the synthetic datasets, the standard deviation of the added noise is known and hence it is not necessary to estimate it from the dataset.

As seen before, for each location observation, the state space is limited to the road segments that lie within a fixed error range. For trajectory datasets whose measurement errors conform to a single Gaussian distribution, the error range is set to be equal to 3 times the standard deviation of the measurement error. For the Wi-Fi-and-cellular dataset, whose errors are modelled as a Gaussian mixture model, the error range is set to 1500 m, as 99.9% of the known measurement errors in the dataset are within this limit.

The transition probability model parameters λ_y and λ_z in Equation 3.6 are common for all types of trajectories. These parameters were estimated based on the ground-truth paths in the paths dataset. As the transition probability model deals with sub-paths between states corresponding to two successive time steps, each ground-truth path in the paths dataset was first randomly segmented into multiple true sub-paths, whose travel durations range from 1 minute to 5 minutes. A total of 4828 such sub-paths with an average length of 2.6 km are considered. For each sub-path, the measure of circuitousness (Equation 3.4) and the measure of temporal implausibility (Equation 3.5) are calculated. It can be seen from the histogram in Figure 3.4 that the values of the measure of circuitousness for all the sub-paths conform to an exponential distribution. Taking the inverse of the mean of all such values, the rate parameter λ_y of the exponential distribution was found to be 0.69. The values of the measure of temporal implausibility for all the sub-

paths do not fit well to an exponential distribution due to the reason that for about 95% of the sub-paths, the measure of temporal implausibility is zero. (i.e. The sub-paths are temporally plausible.) However, the proposed transition probability model seeks to penalize only the temporally implausible sub-paths (i.e. sub-paths with a non-zero measure of temporal implausibility). When only such sub-paths are considered, the values of the measure of temporal implausibility broadly follow an exponential distribution as shown in Figure 3.5. The rate parameter λ_z of this distribution was estimated to be 13.35.

Figure 3.5: Histogram of the measure of circuitousness.

Figure 3.4: Histogram of the measure of temporal implausibility (with only the non-zero values considered).

3.3.3 Evaluation Method and Metrics

The proposed HMM-based map matching algorithm was implemented in C++ and evaluated using the cellular, Wi-Fi-and-cellular and synthetic trajectory datasets. While the trajectories in the datasets were originally recorded mostly at the rate of one sample per second, several temporally-sparse trajectories were derived from each of them by subsampling them at sampling intervals ranging from 60 seconds to 300 seconds. A digitized road network of Singapore containing geometric, attribute, and connectivity information of 65814 road segments is used for the evaluation. The proposed algorithm is compared with the following two map matching algorithms:

- The *closest point* algorithm: This is a simple algorithm that greedily matches each location measurement in the trajectory to a point on its closest road segment and connects successive road segments using the shortest paths between them.
- The *Newson-Krumm* algorithm: This is an online variant of the HMM-based map matching algorithm proposed in [79] that is used as a benchmark by many works on map matching sparse and noisy trajectories (e.g. [95],[6],[88],[94] and [116]). While this algorithm was originally proposed for offline map matching using the standard Viterbi algorithm, the version compared here uses the online Viterbi algorithm that does not compromise the optimality of the output.

The algorithms are evaluated in terms of accuracy, timeliness and running time. It is a common practice in the map matching literature to examine the accuracy of an algorithm in terms of its precision and recall [18, 66, 85]. In the context of map matching (with path inference), they are defined as

$$precision = \frac{L_{correct}}{L_{matched}} \quad (3.10)$$

$$recall = \frac{L_{correct}}{L_{truth}} \quad (3.11)$$

where $L_{matched}$ is the length of the path inferred by the algorithm, L_{truth} is the length of the corresponding ground truth path and $L_{correct}$ is the total length of the portions of the inferred path that overlap with the ground truth path. Some works on map matching (e.g. [95] and [30]) combine precision and recall into a single measure of accuracy known as F-score or F-measure, which is the harmonic mean of precision and recall. It is defined as

$$F_score = 2 \times \frac{precision \times recall}{precision + recall} \quad (3.12)$$

Similar to the approach adopted in [116], the map matching error is defined in this work using the F-score as follows:

$$Map\ matching\ error = 1 - F_score \quad (3.13)$$

The timeliness of the map matching algorithms is evaluated in terms of the average input-to-output latency, which is the time elapsed between the time a location measurement is generated and the time when the inferred path corresponding to it is produced by the algorithm. (It is worth recalling that the online Viterbi algorithm will produce a partial output only when the convergence condition is detected.) The running times of the map matching algorithms are expressed in terms of the average amount of time taken to process one trajectory. A HP Z420 Workstation with an Intel Xeon E5-1650 v2 (3.5 GHz) processor and 16 GB physical memory has been used for the experiments.

3.4 Evaluation Results

The accuracy of the proposed algorithm, in terms of the map matching error expressed as a percentage, is compared with the closest point and the Newson-Krumm algorithms. Figures 3.6 to 3.12, respectively, show the map matching errors obtained with the cellular dataset, the Wi-Fi-and-cellular dataset and the five synthetic datasets at different sampling intervals. Table 3.2 provides a summary of the results by showing the average error across all sampling intervals for each dataset for each algorithm.

Dataset	Closest point algorithm	Newson-Krumm algorithm	Proposed HMM algorithm
Cellular dataset	61.6%	22.4%	12.5%
Wi-Fi-and-cellular dataset	55.7%	24.3%	12.9%
Synthetic dataset (100 m noise)	57.6%	11.4%	4.1%
Synthetic dataset (200 m noise)	62.2%	18.4%	6.9%
Synthetic dataset (300 m noise)	65.0%	26.5%	9.0%
Synthetic dataset (400 m noise)	67.9%	31.1%	12.8%
Synthetic dataset (500 m noise)	69.9%	34.3%	17.0%

Table 3.2: Average map matching errors across all sampling intervals.

Figure 3.6: Comparison of map matching error on the cellular dataset.

Figure 3.7: Comparison of map matching error on the Wi-Fi-and-cellular dataset.

Figure 3.8: Comparison of map matching error on the synthetic dataset (100 m noise).

Figure 3.9: Comparison of map matching error on the synthetic dataset (200 m noise).

Figure 3.10: Comparison of map matching error on the synthetic dataset (300 m noise).

Figure 3.11: Comparison of map matching error on the synthetic dataset (400 m noise).

Figure 3.12: Comparison of map matching error on the synthetic dataset (500 m noise).

It can be seen from the results that the naïve approach of matching each location measurement to the closest point on the road network produces very poor accuracy compared to the other two probabilistic algorithms under all the conditions considered. In the worst case, the map matching error of the closest point algorithm rises to almost 80%, showing its complete unsuitability for the task of inferring the travelled path from imprecise trajectories. The proposed HMM-based algorithm is significantly more accurate than the Newson-Krumm algorithm at all sampling intervals and all noise levels. The overall map matching error of the proposed algorithm is 55% lower than that of the Newson-Krumm algorithm. The map matching error increases gradually with increase in the sampling interval as well as the noise level.

The substantial difference in the accuracies of the proposed algorithm and the compared version of the Newson-Krumm algorithm merits some discussion. Both the algorithms differ mainly in the transition probability model and the difference in the accuracy can be attributed to the following differences between their respective transition probability models. First, while the Newson-Krumm model assumes that transitions between states occur through the shortest-distance path, the proposed model relies on the minimum-travel-time path, which better represents reality. Analysis of the true sub-paths obtained from the paths dataset shows that in most cases, vehicles move from one point to another through the minimum-travel-time path along major roads even when a shorter-distance path involving minor roads exist between those points. Second, the proposed measure of circuitousness, by factoring in the sampling interval and by *not* considering the location observations, overcomes the drawbacks of the measure used by Newson and Krumm, as explained in Section 3.2.2. Finally, unlike the Newson-Krumm model that does not consider the travel times of the transition paths, the proposed model includes a measure of temporal implausibility in order to assign low transition probabilities for paths whose free-flow travel times are higher than the sampling interval.

The mean output latencies of the compared algorithms for the various trajectory datasets are shown in Figures 3.13 to 3.19. The closest point algorithm that immediately snaps each location measurement to the nearest road segment without waiting for further observations, has an output latency of zero. For the other two algorithms that apply the online Viterbi algorithm on a HMM trellis graph, the mean output latency in terms of absolute time generally increases with the sampling interval. (If the output latency is expressed in terms of the number of time steps, smaller sampling intervals would be associated with longer output latencies.) While the Newson-Krumm algorithm produces lower (i.e. better) latencies at relatively lower sampling intervals and noise levels, the proposed algorithm has better latencies at higher levels of imprecision.

Figure 3.13: Comparison of mean output latency on the cellular dataset.

Figure 3.14: Comparison of mean output latency on the Wi-Fi-and-cellular dataset.

Figure 3.15: Comparison of mean output latency on the synthetic dataset (100 m noise).

Figure 3.16: Comparison of mean output latency on the synthetic dataset (200 m noise).

Figure 3.17: Comparison of mean output latency on the synthetic dataset (300 m noise).

Figure 3.18: Comparison of mean output latency on the synthetic dataset (400 m noise).

Figure 3.19: Comparison of mean output latency on the synthetic dataset (500 m noise).

Figures 3.20 to 3.26 show the average time taken by the proposed and the Newson-Krumm algorithm for map matching one trajectory. (The naïve closest point algorithm takes only a negligible amount of computation time, less than one thousandth of the time taken by the other two algorithms, and is not shown in the graphs.) There is no significant difference in the running times of the Newson-Krumm algorithm and the proposed algorithm. In both the algorithms, more than 90% of the overall computation time for most cases is consumed by a single step: the computation of the shortest paths from each candidate point (i.e. state) at one time step to all the candidate points at the next time step. These shortest path computations have been identified by several researchers (e.g. [80], [107]) as the computational bottleneck in probabilistic map matching algorithms. If the average number of location measurements in a trajectory is N and the average number of candidate points considered at each time step is M , then a total of $(N - 1) \times M$ single-source shortest path computations need to be performed. Therefore, when the sampling interval is increased, N in the above term decreases, causing a reduction in the average running time. On the other hand, at higher levels of noise, M becomes exponentially higher, as a larger error region needs to be considered, and therefore the average running time increases rapidly.

Figure 3.20: Comparison of running time on the cellular dataset.

Figure 3.21: Comparison of running time on the Wi-Fi-and-cellular dataset.

Figure 3.22: Comparison of running time on the synthetic dataset (100 m noise).

Figure 3.23: Comparison of running time on the synthetic dataset (200 m noise).

Figure 3.24: Comparison of running time on the synthetic dataset (300 m noise).

Figure 3.25: Comparison of running time on the synthetic dataset (400 m noise).

Figure 3.26: Comparison of running time on the synthetic dataset (500 m noise).

For the maximally imprecise trajectory data used in the experiments (i.e. 300-seconds sampling interval with synthetic noise of 500 m standard deviation), the map matching error, the mean output latency, and average running time per trajectory are 32%, 783 seconds and 23 seconds, respectively. These numbers indicate the need to further improve the performance of the map matching algorithm before such trajectory data can be used for practical online applications.

3.5 Summary

A HMM-based online map matching algorithm, best suited for temporally sparse trajectories with high levels of measurement noise, has been presented in this chapter. It includes new emission and transition probability models tailored for the type of trajectory data being targeted. A novel aspect of the proposed emission probability model is the use of a Gaussian mixture distribution to model the measurement errors in coarse-grained trajectories generated by smartphones using a combination of Wi-Fi and cellular network positioning. The proposed transition probability model is based on a refined measure of path circuitousness and a new measure to penalize paths with implausible travel times. An online variant of the Viterbi algorithm is used for finding the most likely state sequence in the HMM. The procedures for estimating the model parameters from ground-truth annotated trajectory dataset have been presented.

The proposed algorithm has been evaluated using several trajectory datasets with different levels of real as well as synthetic measurement noise. It achieves substantially higher accuracy at all noise levels and sampling intervals compared to the widely-used algorithm proposed by Newson and Krumm. In terms of output latency, the proposed solution outperforms the Newson-Krumm algorithm at higher sampling intervals and noise levels. While there is no significant difference in the running times of both algorithms, it has been found that the shortest path computations between sets of candidate points at successive time steps account for most of the total running time.

Despite the superior performance of the proposed HMM-based algorithm, there is a need to further improve its accuracy, timeliness and computational efficiency, especially at higher levels of imprecision in the input trajectory data. The next two chapters aim to address this requirement.

Chapter 4

A Route Choice Model for Robust Map Matching

This chapter presents a novel technique that seeks to overcome a fundamental limitation of the HMM-based map matching framework described in the last chapter, namely, its inability to represent higher-order dependencies among states. The proposed technique involves complementing the HMM-based algorithm with a model of drivers' short-term route choice behaviour in order to improve its accuracy and robustness to imprecision in the input data. A method of modelling route choice behaviour from real trajectory data is presented. The overall map matching solution that combines the HMM with the route choice model is evaluated under different levels of imprecision in the input trajectories. This work has been previously published in [33].

4.1 A Limitation of HMM-based Map Matching

The HMM used for map matching follows the Markov assumption, which states that the probability of the next state depends only on the current state and is conditionally independent of all the previous states given the current state. In other words, only first-order dependencies between states are represented in the HMM. In the context of map matching, such a HMM is not a faithful model of the movement of a vehicle along a road network. In reality, the probability of a vehicle moving from point A to point B on a road network depends on the broader

context of route followed from an origin before point A and an intended destination after point B . Statistical analysis of large trajectory datasets have shown that movement of a vehicle along a road network is non-Markovian, especially when a vehicle intends to reach a specific destination, typically through the shortest or fastest path [72]. In other words, the paths taken by vehicles are influenced by the drivers' route choices, which are not well accounted for in the HMM framework.

The above limitation becomes more apparent when a high level of measurement noise is present in the input trajectory data. Given a noisy input trajectory, some parts of the path inferred by the HMM-based algorithm may appear implausible from a route choice perspective. This is illustrated using a simplified example in Figure 4.1 where the HMM-based approach may find the state sequence $s_{1,1}, s_{2,1}, s_{3,1}$ to be most likely, although the alternative sequence $s_{1,1}, s_{2,2}, s_{3,1}$ appears more probable from the standpoint of human route choice. A model of drivers' route choice behavior is likely to assign a higher probability to the choice of the sequence $s_{1,1}, s_{2,2}, s_{3,1}$ compared to that of the sequence $s_{1,1}, s_{2,1}, s_{3,1}$. Intuitively, it appears natural to hypothesize that the map-matching accuracy could be improved by complementing the HMM-based algorithm with a route choice model. This idea is explored and evaluated in the following sections.

Figure 4.1: A map-matched path that is improbable from a route choice perspective.

4.2 Short-Term Route Choice Modelling

Given a number of alternative paths between two points in a road network, a route choice model [117] can be used to estimate the probability of each of the alternative paths being chosen by a driver. In the following subsections, a procedure for generating a set of alternative paths, known as a *choice set*, for a given pair of on-road points is first proposed. Subsequently, a route choice model that gives the probability of a particular alternative path based on the path's attributes is presented. The process of estimating the parameters of the model is also described.

4.2.1 Choice Set Generation

For any two points in a road network, the number of all possible alternative paths between them could be very large. However, this may include many highly circuitous and unreasonable paths that would not be considered by any driver. For the purpose of route choice modelling, it is necessary to generate a choice set containing a limited number of reasonable and distinct alternative paths between a given origin and destination.

A common approach to the problem of finding multiple paths between an origin-destination pair is to apply a k -shortest-paths algorithm [118]. However, empirical studies have shown that k -shortest-paths algorithms tend to produce paths that are highly similar [119]. For instance, k -shortest-paths algorithms may produce a path along a freeway and another path that repeatedly exits and re-enters the freeway through slip roads at interchanges. The latter path may hardly be considered as an alternative to the former. Another approach, known as the link-penalty approach, performs repeated shortest-path computations and minimizes overlapping among paths by imposing a penalty on links (i.e. road segments) in paths that are already found. The proposed method for choice set generation, explained below, is broadly based on the link-penalty approach.

In the proposed improvement to the map matching algorithm, explained later in Section 4.3, a route choice model is used to estimate the route choice probabilities of each *partial* path produced by the HMM-based online algorithm and its alternatives. Such partial paths, whose travel times are typically in the order of a few minutes, may not correspond to the complete trips made by drivers (e.g. from home to work). Rather, they connect pairs of points that lie along a complete path. The time points at which the vehicle was at the start and end points of a partial path and therefore the elapsed time between them are known. Analysis of true paths corresponding to real trajectory data shows that between points separated by a few minutes of travel, there are rarely more than 3 or 4 reasonable alternative paths. Therefore, in this work, the choice set is limited to a maximum of 5 alternative paths.

The first alternative path in the choice set is pre-identified and made available. In the current context, the HMM-generated partial path is considered as the pre-identified first alternative path in the choice set. As paths found based on the minimum-travel-time criterion are known to account for most of the paths chosen by real drivers [120], it needs to be ensured that the minimum-travel-time path is present in the choice set. Therefore, the minimum-travel-time path is computed using Dijkstra's algorithm and added to the choice set as the second alternative path if it is different from the first alternative path.

Subsequently, a penalty is applied to the links in the above two paths. This is done by inflating the travel times of the relevant links by a factor in order to reduce the chances of them being part of subsequent paths found based on the minimum-travel-time criterion. However, as pointed out in [119], applying a penalty to links that are close to the start point or end point of the paths may cause unreasonable paths to be found subsequently. Therefore, while inflating the travel time of a link that lies along a path in the choice set, the distances along the path from the start point to the link and from the link to the end point are taken into consideration. The inflated travel time of link e , with start node u and end node v , lying along path p_i between start point q and end point r is given by

$$\tau'(e) = \tau(e) + \tau(e) \omega \frac{\min(d_i(q,u), d_i(v,r))}{d_i(q,r)} \quad (4.1)$$

where $\tau(e)$ is the original travel time of link e , ω is the inflation factor and the function d_i gives the driving distance between two points along path p_i . An inflation factor value of 5 has been empirically found to produce the best results.

Subsequently, the minimum-travel-time path is computed thrice. After each path computation, the travel times of the links in the path are inflated as explained above. The computed path is added to the choice set only if it meets two conditions: (i) It does not overlap any of the alternative paths already present in the choice set by more than a proportion α , which is empirically set to 50%, and (ii) It should be temporally possible. Allowing for a high degree of speeding, a path is considered to be temporally possible if its free-flow travel time is not more than thrice the actual elapsed time between its start and end points. In this manner, the choice set is populated with a maximum of 5 alternative paths. Figure 4.2 shows an example case of 3 alternative paths generated by the above procedure on the Singapore road network. (The abbreviations in the picture denote the path attributes explained in the next subsection.) The steps involved in the choice set generation method are presented in Algorithm 4.1.

Figure 4.2: An example of multiple alternative paths produced by the choice set generation method (Base map image: Google Maps).

Algorithm 4.1: Choice Set Generation

Input:

- Start point q ;
- End point r ;
- A pre-identified path between q and r , p_p .

Output:

A choice set with up to 5 alternative paths, C .

1. Add pre-identified path p_p to choice set C .
 2. Compute the minimum-travel-time path p_m between q and r .
 3. Add p_m to C if different from p_p .
 4. Inflate the travel times of all links in p_p and p_m .
 5. Compute the current minimum-travel-time path p_m between q and r .
 6. Add p_m to C if it does not overlap any path in C by more than 50% and is temporally possible.
 7. Inflate the travel times of all links in p_m .
 8. Repeat steps 5-7 twice.
 9. Reset the travel times of all links to original values.
-

4.2.2 Route Choice Model

In most route choice models, a driver's preference for an alternative path is quantified by a value called *utility* and the driver is assumed to choose the alternative path with the highest utility. The utility of an alternative path depends on the attributes of the alternative path. It may also depend on the characteristics of the driver and the trip. However, such information is generally not obtainable from anonymized trajectory data and, therefore, not considered in this work.

Given a choice set, C , consisting of several alternative paths, the utility of alternative path $p_i \in C$ is given by

$$U_i = V_i + \varepsilon_i \quad (4.2)$$

where V_i is a deterministic term and ε_i is a random term that captures the uncertainty due to unobserved factors that affect the utility but are not considered in the utility calculation. The deterministic term V_i is expressed as a

linear-in-parameters function of the attributes of alternative path p_i . In this work, the following attributes that are known to influence drivers' route choices are considered.

- Free-flow travel time (FTT; in seconds)
- Number of traffic signals (NTS)
- Average road class (ARC; road classes are numbered from 1, starting with the highest class)
- Number of class changes (NCC)

The deterministic term of the utility for alternative path p_i is given by

$$V_i = \beta_{FTT}FTT_i + \beta_{NTS}NTS_i + \beta_{ARC}ARC_i + \beta_{NCC}NCC_i \quad (4.3)$$

where FTT_i is the FTT attribute of alternative path p_i , β_{FTT} is a coefficient parameter corresponding to the FTT attribute, and so on. Equation 4.3 can be rewritten as

$$V_i = \boldsymbol{\beta}' \mathbf{x}_i \quad (4.4)$$

where $\boldsymbol{\beta}$ is a vector of parameters $(\beta_{FTT}, \beta_{NTS}, \beta_{ARC}, \beta_{NCC})$ and $\mathbf{x}_i$ is a vector of attributes $(FTT_i, NTS_i, ARC_i, NCC_i)$ corresponding to the alternative path p_i .

Multinomial logit models [117] are widely used for choice modelling due to their simplicity and tractability. In a multinomial logit model, the random term ε_i in Equation 4.2 is assumed to be an independent and identically distributed random variable that follows the Gumbel distribution. The scale factor of the random term is irrelevant to the choice of the alternative with the highest utility [121] and is therefore typically normalized to one. The mean of the random term represents the average impact of all other factors that are not included in the calculation of the deterministic term. Without loss of generality, the mean of the random term can be assumed to be zero if an alternative-specific constant is included in the calculation of the deterministic term of the utility. The alternative-specific constant, which is an unknown parameter to be estimated, represents the bias towards an alternative due to factors that are not considered

in the model. However, the notion of alternative-specific constants is not intuitively meaningful in the context of route choice, where alternative paths in different situations may not have anything in common with each other. For instance, it does not make sense to suppose that the k^{th} alternative path in one choice set has the same level of bias as that of the k^{th} alternative path in another choice set. Therefore, alternative-specific constants are not included in the proposed route choice model. That is, the unobserved factors are assumed to have the same effect on all the alternative paths. Overall, the proposed route choice model can be considered as a reasonable approximation of a multinomial logit model. Based on this model, the probability of a driver choosing alternative path p_i from a choice set C can be estimated as

$$P(p_i|C) = \frac{e^{\beta'x_i}}{\sum_{p_j \in C} e^{\beta'x_j}} \quad (4.5)$$

4.2.3 Estimation of Model Parameters

The β parameters in the multinomial logit model (Equations 4.3 to 4.5) can be estimated from a sample of choice observations. In the context of route choice, the choice observations correspond to path choices made by real drivers. A choice observation n consists of a choice set C_n with several alternative paths and information about which alternative path was actually chosen by a driver. Given a path chosen by a driver between a pair of points during a real drive, it is included as the pre-identified first alternative path in a choice set. Subsequently, other feasible alternative paths between the pair of points are generated using the choice set generation method. This process is repeated for a number of chosen paths obtained from ground-truth labelled trajectory datasets.

Maximum likelihood estimation is the most common method used for estimating the parameters of route choice models. It estimates the values of the parameters for which the sample of choice observations is most likely to have occurred [122]. Given a sample of N choice observations, the likelihood function for a general choice model is given by

$$\mathcal{L}^* = \prod_{n=1}^N \prod_{p_i \in C_n} P(p_i | C_n)^{y_{in}} \quad (4.6)$$

where y_{in} is an indicator variable that equals 1 if p_i is the chosen alternative in choice set C_n , 0 otherwise. For a multinomial logit model, the likelihood function is given by

$$\mathcal{L}^* = \prod_{n=1}^N \prod_{p_i \in C_n} \left(\frac{e^{\beta' x_{in}}}{\sum_{p_j \in C_n} e^{\beta' x_{jn}}} \right)^{y_{in}} \quad (4.7)$$

where β' is the vector of parameters to be estimated and x_{in} is the vector of attributes of alternative path p_i in choice set C_n . It is convenient to take the logarithm of $\mathcal{L}^*$ and use the log-likelihood function given by

$$\mathcal{L} = \sum_{n=1}^N \sum_{p_i \in C_n} y_{in} \left(\beta' x_{in} - \ln \sum_{j \in C_n} e^{\beta' x_{jn}} \right) \quad (4.8)$$

The β parameters corresponding to the maximum of $\mathcal{L}$ are estimated by differentiating $\mathcal{L}$ with respect to each β parameter and setting the partial derivatives to zero. This involves solving a system of equations using computationally intensive methods. However, there are open source software packages such as Biogeme [123] for estimating the parameters of choice models including the multinomial logit model.

4.3 The Improved Map Matching Algorithm

The improved solution for map matching combines the route choice model described in the previous section with the HMM-based online map matching algorithm presented in the last chapter in the following manner.

Location observations from each vehicle-based user are received and processed on the fly. Each time a convergence condition is detected by the online Viterbi algorithm, the partial path up to the convergence state is released for

further processing. This path is referred to as the HMM-generated partial path. A convergence state represents a point on the road network where a location observation is matched with a relatively high degree of confidence. Therefore, it can be said that the start and end points of the HMM-generated partial paths are generally accurate, while the paths themselves may have a degree of error associated with them. Multiple alternative paths are generally plausible between the start and end points of the HMM-generated partial path.

Let q and r be start and end points, respectively, of a HMM-generated partial path. The choice set generation method described in Section 4.2.1 is applied to generate a number of reasonable and largely distinct alternative paths between q and r . As seen earlier, the choice set unconditionally includes the HMM-generated partial path as one of the alternatives.

The task now is to identify the most-likely path from among the several alternative paths in the choice set. The alternative path that has a high probability of being chosen by a driver *and* is highly consistent with the sequence of location measurements is considered as the most-likely path. Towards this end, two probability measures that are referred to as the route choice probability and the observation generation probability, respectively, are estimated for each alternative path.

The route choice probability $P(p_i|C)$ is the probability of a driver choosing the alternative path p_i within the choice set C , as given in Equation 4.5. The observation generation probability of an alternative path is the probability that the given location observations were generated while traveling along that path. Let $o_{1:K}$ be a sequence of K location observations corresponding to the alternative paths in the choice set. If the location measurement errors are assumed to follow a single zero-mean Gaussian distribution, the observation generation probability for a given alternative path p_i is defined as

$$P(o_{1:K}|p_i) = \prod_{k=1}^K \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{g_c(o_k, p_i)^2}{2\sigma^2}} \quad (4.9)$$

where σ is the standard deviation of the location measurement error and $g_c(o_k, p_i)$ is the great-circle distance between o_k and the point closest to it on p_i . If the location measurement errors are assumed to follow a two-component zero-mean Gaussian mixture distribution, the observation generation probability for a given alternative path p_i is defined as

$$P(o_{1:K}|p_i) = \prod_{k=1}^K \left(\frac{w_1}{\sigma_1 \sqrt{2\pi}} e^{-\frac{g_c(o_k, p_i)^2}{2\sigma_1^2}} + \frac{w_2}{\sigma_2 \sqrt{2\pi}} e^{-\frac{g_c(o_k, p_i)^2}{2\sigma_2^2}} \right) \quad (4.10)$$

where w_i and σ_i are the weight and standard deviation, respectively of the i^{th} component distribution.

Among all the alternative paths in the choice set, the one that maximizes the product of the route choice probability and the observation generation probability is selected as the most likely path taken by the vehicle between the start and end points. This path is released as the partial output of the improved map-matching algorithm.

The high-level steps in the proposed overall solution are summarized below.

1. If a convergence condition is detected in the online Viterbi algorithm, produce the HMM-generated partial path.
2. Generate a choice set of multiple alternative paths including the HMM-generated partial path.
3. For each alternative path, compute the route choice probability and the observation generation probability.
4. Output the alternative path that maximizes the product of the route choice and observation generation probabilities and go to step 1.

The improved map matching algorithm described above, besides overcoming the inability of HMM to consider long term dependencies, also has another benefit: While HMM-based map matching assumes that transitions between two neighbouring states (i.e. candidate points at successive time steps)

occur only through the optimal path (e.g. minimum-travel-time path) between them, the use of the route choice model allows multiple alternative paths to be considered. This point could be better understood by considering a situation where the convergence condition is detected by the Viterbi algorithm at two successive time steps. In such a situation, the HMM-generated partial path would be the optimal path between the two convergence states. The improved algorithm will generate and assess multiple alternatives to the optimal path.

4.4 Evaluation

The parameters of the route choice model have been estimated based on the ground-truth paths in the paths dataset. As seen before, in the proposed solution, the route choice model is applied to partial paths that typically span a few time steps. Therefore, the ground-truth paths in the paths dataset have been randomly segmented into a number of partial paths, whose travel durations range from 1 minute to 10 minutes. A total of 2016 such paths, with an average length of 4.2 km, are used as input for estimating the parameters. Each of these paths is included as the first alternative path in a choice set. (i.e. The first alternative path in each choice set is the actual chosen path.) The choice set generation method is used to generate up to 4 alternatives to the chosen path. Thus, each choice observation consists of a choice set with up to 5 alternatives. The β parameters of the route choice model, estimated by maximum likelihood using the Biogeme [123] software package, are presented in Table 4.1. All the estimated parameters are statistically significant ($p < 0.05$).

Parameter	Corresponding attribute	Estimated value	<i>t</i> -test value
β_{FTT}	Free-flow travel time (s)	-0.019	-6.97
β_{NTS}	Number of traffic signals	-0.100	-5.48
β_{ARC}	Average road class	-0.244	-2.25
β_{NCC}	Number of class changes	-0.272	-4.68

Table 4.1: Estimated parameters of the route choice model.

The improved map matching algorithm is evaluated using the experimental setup described in the last chapter. For clarity, the proposed overall solution that includes the HMM and the route choice model is referred to as the *HMM+RCM* algorithm in the following discussion. The HMM-based algorithm presented in the last chapter, which serves as the baseline, is referred to as the *HMM-only* algorithm. The map matching error of the HMM+RCM algorithm is compared with that of the HMM-only algorithm at different sampling intervals and noise levels. It needs to be noted that there is no difference in the output latencies of the two algorithms as both rely on the same online Viterbi algorithm to output partial paths. As for the running time, the difference between the two algorithms is negligible (less than 1% in most cases). Therefore, issues related to output latency and running time are not discussed below. However, they are the focus of the next chapter.

Figures 4.3 to 4.9, respectively, show the map matching errors obtained with the cellular dataset, the Wi-Fi-and-cellular dataset and the five synthetic datasets at different sampling intervals. Averaging across all datasets and all sampling intervals, the map matching errors of the HMM-only algorithm and the HMM+RCM algorithm are 10.7% and 9.4% (i.e. a 12% reduction in the error). In general, the HMM+RCM algorithm has significantly lower map matching errors compared to the HMM-only algorithm at higher levels of measurement noise and at higher sampling intervals. For instance, considering only those synthetic datasets with a noise level of 300 m or higher at a sampling interval of 180 seconds or higher, the map matching errors of the HMM-only algorithm and the HMM+RCM algorithm are 15.9% and 13.6%, respectively. This is not surprising because, as explained in Section 4.1, the HMM-only algorithm is more likely to produce unreasonable partial paths when the degree of imprecision in the input data is high. For such input data, the inclusion of the route choice model in the map matching algorithm proves more useful.

When the input trajectory data has lower levels of imprecision, the HMM+RCM algorithm does not produce a considerable and consistent reduction

in the map matching error. In the case of Wi-Fi-and-cellular dataset, whose median measurement error (56 m) is lower than that of all the other datasets, the HMM+RCM algorithm has the same accuracy as the HMM-only algorithm for all the sampling intervals. Although a small proportion of highly noisy location observations are present in this dataset (corresponding to places without Wi-Fi signals), the use of the route choice model does not seem to make any difference. However, this is a small dataset with only 10 short trajectories and as such, it may not be advisable to draw firm conclusions from this small sample.

Figure 4.3: Evaluation of the improved algorithm on the cellular dataset.

Figure 4.4: Evaluation of the improved algorithm on the Wi-Fi-and-cellular dataset.

Figure 4.5: Evaluation of the improved algorithm on the synthetic dataset (100 m noise).

Figure 4.6: Evaluation of the improved algorithm on the synthetic dataset (200 m noise).

Figure 4.7: Evaluation of the improved algorithm on the synthetic dataset (300 m noise).

Figure 4.8: Evaluation of the improved algorithm on the synthetic dataset (400 m noise).

Figure 4.9: Evaluation of the improved algorithm on the synthetic dataset (500 m noise).

Despite the generally positive results achieved, it is necessary to note that there are situations in which a vehicle is driven along routes that may not make sense from a route choice perspective (e.g. unoccupied taxis looking for passengers [72] or a person driving in an unfamiliar area). Map matching based on route-choice criteria may be ineffective or even counterproductive for trajectories from such trips. The trajectories used for evaluations in this work correspond to actual taxi trips. It is known that taxi drivers are generally adept at choosing optimal routes and are less likely to take circuitous paths compared to other drivers. As part of future work, it would be informative to evaluate the proposed solution in the presence of more diverse route choice behaviour.

4.5 Summary

While HMM-based algorithms are widely considered to be the state of the art for map matching sparse and noisy trajectory data, they are limited by the Markov assumption that permits only first-order dependencies between states at neighbouring time steps. This prevents the map matching algorithms from examining state sequences spanning more than two time steps from the perspective of route optimality or driver preference. However, in most cases, movements of vehicles occur along routes that are considered optimal according to the drivers' route preferences. Against this background, this chapter has examined the novel idea that complementing a HMM-based map matching algorithm with a model of drivers' route choice behaviour may help to improve the map matching accuracy.

The proposed approach to modelling short-term route choice behaviour includes a choice set generation procedure for generating a number of reasonable alternative paths for a given pair of on-road points. A simplified multinomial logit model is used to estimate the probability of an alternative path being chosen based on several path attributes. The parameters of the model have been successfully estimated from actual travelled paths corresponding to real trajectory data.

An improved map matching algorithm, which adopts a route-choice based approach to refine the output of the HMM-based algorithm presented in the last chapter, has been proposed. In the improved algorithm, the route choice model is used to reassess each HMM-generated partial path along with a set of feasible alternative paths. The proposed solution has been applied to several sets of trajectory data with different levels of imprecision and its accuracy has been compared with that of the baseline HMM-based algorithm. The results show that the proposed refinement helps to achieve considerably better map matching accuracy, particularly for trajectory data with high levels of measurement noise and longer sampling intervals.

Chapter 5

Optimization Techniques for Online Map Matching

Time-critical smart mobility applications require a large volume of streaming trajectory data to be processed on the fly. This in turn requires the trajectories to be map matched in a timely and efficient manner. However, as seen in Chapter 3, the state-of-the-art map matching algorithms have substantial output latencies and computation times, especially when applied to highly imprecise input data. This chapter presents optimization techniques for improving the efficiency of online map matching in terms of output latency and computation time. Early versions of this work have been previously published and awaiting publication, respectively, in [34] and [35].

5.1 Optimization for Improving Output Latency

In the following, a brief review of the existing approaches to reduce the output latency of probabilistic map matching algorithms is first presented. Subsequently, an optimization technique for output latency reduction is proposed and evaluated.

5.1.1 Background and Related Work

The state-of-the-art map matching algorithms rely on a probabilistic graphical model such as a HMM and find the most likely state sequence in it using the

Viterbi algorithm. In its standard form, the Viterbi algorithm outputs the most likely state sequence corresponding to a trajectory after all the location observations in the trajectory are received. Obviously, this mode of operation incurs long output delays and is therefore unsuitable for time-critical online applications such as traffic condition estimation where the currency of the output estimates is of high importance. Therefore, many of the published map matching algorithms include techniques for reducing the latency of the Viterbi algorithm for online operation.

A simple latency-reduction method is use a fixed-sized sliding window to divide the trajectory into short sequences of location observations and to find the most likely state sequence for each such observation sequence [80, 85]. However, in order to achieve good accuracy, larger windows are needed, resulting in higher output latencies [83]. Another approach known as lagged smoothing [60] involves processing a fixed-number, say k , of subsequent observations before outputting the state corresponding to a given observation. Lagged smoothing is performed using the forward-backward algorithm [76], which computes the probability of a state at time step t , given the observations up to time step $t + k$. However, a fixed lag may not suit all situations, as in some cases a higher than usual number of subsequent observations may be needed to resolve the uncertainty at a time step. While the abovementioned techniques limit the output latency to a fixed number of time steps, they are prone to producing suboptimal solutions.

As explained in Section 3.2.3, the map matching algorithm proposed in this work uses an online variant of the Viterbi algorithm that helps to achieve lower output latencies without compromising the optimality of the output. The online Viterbi algorithm [83, 124, 125] outputs a partial state sequence up to a particular state $s_{t,j}$ at time step t if the chains of back pointers from *all* the states at a subsequent time step $t + m$ converge at $s_{t,j}$. In such a case, the output latency is m , which varies depending on the situation. Thus, the online Viterbi algorithm waits for a variable number of time steps for a convergence state to be detected before it outputs a partial state sequence.

In practice, it is not always necessary to wait until a convergence state is detected by the online Viterbi algorithm. Often, it is possible to infer with a high degree of certainty that a particular state would be part of the eventual most likely state sequence even before it is confirmed with absolute certainty by a convergence state. In line with this idea, Narasimhan et al. [III] proposed a method of estimating the probability of a state at time step t being part of the eventual most likely sequence based on observations up to any subsequent time step $t + k$. Based on this probabilistic estimate, they calculate an uncertainty measure such that the uncertainty measure is small when the above probabilistic estimate is concentrated on a single state. (i.e. One state has a much higher probability of being part of the eventual most likely sequence compared to all other states.) Narasimhan et al. assume that the uncertainty measure decreases as more future observations are seen. Based on this assumption, they proposed an ‘online-step’ method that chooses a future time step $t + k$ for generating the partial sequence up to time step t such that a combination of the latency and the uncertainty measure is minimized. This method effectively places an upper bound on the latency. An approach similar to the one described above was later applied for HMM-based map matching by Wang and Zimmermann [109].

5.1.2 The Proposed Method

The method used in this work for reducing the output latency of the map matching algorithm is based on the probabilistic estimate proposed by Narasimhan et al. [III], but differs from their work in the choice of the uncertainty measure and in not enforcing an upper bound on the output latency. Unlike [III] and [109], the proposed method does not seek a trade-off between accuracy and output latency. Rather, the aim here is to minimize output latency with negligible loss of accuracy. In keeping with this aim, no upper bound is placed on the output latency, as doing so may lead to significant loss of accuracy in some situations. For instance, in a situation where two long parallel roads fit the location observations with almost equal probability, it is necessary to wait for more observations until the two roads are sufficiently disambiguated.

As seen in Chapter 3, the Viterbi algorithm computes the Viterbi path probability for each state at each time step. For any state $s_{t,k}$ at time step t , the Viterbi path probability is the joint probability of the most likely state sequence ending at state $s_{t,k}$ based on the observations $o_1, \dots, o_t$. It is expressed as

$$V_{t,k} = P(o_t | s_{t,k}) \max_j (V_{t-1,j} P(s_{t,k} | s_{t-1,j})) \quad (5.1)$$

where $P(o_t | s_{t,k})$ and $P(s_{t,k} | s_{t-1,j})$ are given by the emission and transition probability models, respectively, of the HMM. The index j in Equation 5.1 that maximizes $V_{t,k}$ is stored as a back pointer for state $s_{t,k}$. The back pointer points to the state $s_{t-1,j}$ at the previous time step that lies along the most likely sequence ending at $s_{t,k}$. Figure 5.1 shows a simple example where each state at each time step, except the first one, points to a state at the previous time step. In this example, there is no convergence state and it is not possible to say with absolute certainty which of the states at each time step will be part of the eventual most likely state sequence when all the future observations are considered.

Figure 5.1: An illustration of back pointers in the Viterbi algorithm.

Let $\bar{\delta}(s_{t,k}, s_{t-m,j})$ be a function that indicates if the state $s_{t-m,j}$ at a past time step $t - m, m \geq 1$, is reachable through a chain of back pointers from state $s_{t,k}$ at time step t . It is defined as

$$\bar{\delta}(s_{t,k}, s_{t-m,j}) = \begin{cases} 1 & \text{if } s_{t-m,j} \text{ is reachable from } s_{t,k} \\ 0 & \text{otherwise} \end{cases} \quad (5.2)$$

Let $\mathbf{o}_{1:T} = \{o_1, \dots, o_T\}$ be the *complete* sequence of location observations in a trajectory. Let $\mathbf{o}_{1:t} = \{o_1, \dots, o_t\}$, $t < T$ be the partial sequence of location observations received up to time step t . Let $\mathbf{s}_{1:T}^* = \{s_1^*, \dots, s_T^*\}$ be the *eventual* most likely state sequence when the complete sequence of location observations $\mathbf{o}_{1:T}$ is taken into consideration. However, at time step t , only the partial observation sequence $\mathbf{o}_{1:t}$ is available. In such a situation, the probability that a particular state at time step $t - m$ would be part of the eventual most likely state sequence is estimated as follows:

$$P(s_{t-m,j} = s_{t-m}^* | \mathbf{o}_{1:t}) = \sum_{k=1}^{N_t} V_{t,k} \bar{\delta}(s_{t,k}, s_{t-m,j}) \quad (5.3)$$

In the above equation, N_t denotes the number of states at time step t and $P(s_{t-m,j} = s_{t-m}^* | \mathbf{o}_{1:t})$ represents the probability of $s_{t-m,j}$ being the state at time step $t - m$ on the eventual most likely state sequence $\mathbf{s}_{1:T}^*$, given the partial observation sequence $\mathbf{o}_{1:t}$ received up to time step t .

If a particular state at time step $t - m$, say $s_{t-m,j}$, is a convergence state, then only that state will have a non-zero probabilistic estimate as given by Equation 5.3. (This is because the reachability function $\bar{\delta}$ will return zero for all other states.) In such a case, the partial state sequence up to $s_{t-m,j}$ can be released without any loss of global optimality. However, when more than one state at time step $t - m$ has a non-zero probabilistic estimate, it is reasonable to examine if one of the states has a much higher probability of being part of the eventual most likely state sequence compared to all other states. If so, the partial state sequence up to that dominant state may be released in advance without generally incurring any significant loss of optimality.

Let s'_{t-m} and s''_{t-m} be the states at time step $t - m$ that are estimated to have the highest and second-highest probabilities, respectively, of being part of the eventual most likely state sequence. Let h_{t-m} be a measure of *certainty* associated with time step $t - m$ such that higher values of h_{t-m} indicate that the probabilistic estimate given by Equation 5.3 is more concentrated on a single state at that time step. This means that when the difference between the probabilistic estimates of s'_{t-m} and s''_{t-m} is higher, the measure of certainty h_{t-m} is also higher. Based on this insight, h_{t-m} is defined as the ratio of the highest probabilistic estimate among all the states of time step $t - m$ to the second-highest probabilistic estimate.

$$h_{t-m} = \frac{P(s'_{t-m} = s^*_{t-m} | \mathbf{o}_{1:t})}{P(s''_{t-m} = s^*_{t-m} | \mathbf{o}_{1:t})} \quad (5.4)$$

In the proposed method, the partial state sequence up to the dominant state s'_{t-m} is released as output if the certainty measure h_{t-m} exceeds a threshold τ . The numerator in Equation 5.4 cannot be less than the denominator by definition and therefore, the minimum possible value of h_{t-m} is 1. Setting the threshold τ to be 1 causes the state with the highest probabilistic estimate to be chosen greedily at each time step. While this minimizes the output latency, it will lead to a significant loss of accuracy. A higher value of the threshold τ will cause the latency to increase, but will generally produce more accurate outputs. The optimal value of the threshold τ is to be determined empirically.

5.1.3 Evaluation

The evaluation of the proposed latency reduction technique is carried out using the same experimental setup used in the previous chapters. In order to determine the optimal value of the threshold τ , the proposed method is applied to all the trajectory datasets with different values of τ . The output latency and map matching error values, averaged across all datasets and all sampling intervals, are presented in Table 5.1. The HMM-based algorithm presented in the Chapter 3 is used as the baseline for assessing the effect of the proposed latency reduction

method. Averaging across all datasets and all sampling intervals, the baseline algorithm has a map matching error and mean output latency of 10.74% and 473.77 seconds, respectively. The percentage change figures in Table 5.1 are with respect to the above baseline values. Not surprisingly, the lowest mean output latency is obtained when τ is set to 1. (i.e. The state with the highest probabilistic estimate is greedily chosen irrespective of how well it outscores the other states.) This ensures that a partial output is generated at every time step with a latency of one time step. The observed mean latency of 181 seconds in this case is in fact equal to the average duration of one time step (i.e. the average sampling interval) in the experiments. However, setting τ equal to one increases the map matching error considerably, by 29% over the baseline. The error improves (i.e. reduces) when τ is increased. However, it can be seen from the Table 5.1 that when τ is increased beyond 30, the improvement in the map matching error is not consistent and significant. Therefore, τ is set to 30 in this work. At this threshold value, the increase in the overall map matching error with respect to the baseline is negligible ($< 0.5\%$), but the overall reduction in the mean output latency is considerable (33%).

Threshold τ	Mean output latency (seconds)	% Change in mean output latency	Map matching error (%)	% Change in map matching error
1	181.07	-61.78%	13.90	+29.34%
10	281.72	-40.54%	11.04	+2.78%
20	302.36	-36.18%	10.83	+0.81%
30	315.11	-33.49%	10.79	+0.44%
40	321.25	-32.19%	10.80	+0.51%
50	324.95	-31.41%	10.77	+0.22%

Table 5.1: Overall mean output latency and map matching error for different values of threshold τ across all datasets and sampling intervals.

Table 5.2 shows a comparison of the mean output latencies for the baseline algorithm and the optimized algorithm that includes the proposed latency

reduction technique (with τ set to 30) for each dataset, averaged across the five sampling intervals ranging from 60 seconds to 300 seconds. The highest percentage reduction in the output latency is observed for the Wi-Fi-and-cellular dataset. It is worth recalling that although the Wi-Fi-and-cellular dataset has a low median measurement error of 56 m, a large error region of 1500 m (leading to a large state space) is considered in order to accommodate the outlier observations corresponding to places without Wi-Fi signals. This arguably causes the Viterbi algorithm to maintain many highly improbable state sequences. However, such state sequences, which have very little chance of being part of the eventual most-likely state sequence, would be eliminated by the proposed method, leading to a substantial reduction in the output latency. The results obtained for the five synthetic datasets with added Gaussian noise suggest that the effectiveness of the proposed technique, in terms of percentage reduction in the output latency, gradually increases with the noise level.

	Mean output latency of the baseline algorithm (seconds)	Mean output latency of the optimized algorithm (seconds)	% Change
Cellular dataset	521.41	356.62	-31.60%
Wi-Fi-and-cellular dataset	496.00	248.46	-49.91%
Synthetic – 100 m noise dataset	288.67	216.72	-24.92%
Synthetic – 200 m noise dataset	368.69	262.30	-28.86%
Synthetic – 300 m noise dataset	463.22	327.81	-29.23%
Synthetic – 400 m noise dataset	539.43	363.67	-32.58%
Synthetic – 500 m noise dataset	639.00	430.22	-32.67%

Table 5.2: Mean output latencies for each dataset, averaged across all sampling intervals.

Figures 5.2 to 5.8, respectively, show the mean output latencies obtained with each dataset at different sampling intervals. In absolute terms, the reduction in the latency against the baseline is fairly consistent across all the sampling intervals. In terms of percentage reduction, the proposed method is more effective at lower sampling intervals. It needs to be noted that the mean output latency of the baseline algorithm, in terms of the number of time steps, is higher at lower sampling intervals. For instance, the cellular dataset has a mean output latency of about 6 and 2 time steps, respectively, for the 60-seconds and 300-seconds sampling intervals. Hence, it could be argued that there exists more potential for reducing the output latency at lower sampling intervals.

Figure 5.2: Comparison of mean output latency on the cellular dataset.

Figure 5.3: Comparison of mean output latency on the Wi-Fi-and-cellular dataset.

Figure 5.4: Comparison of mean output latency on the synthetic dataset (100 m noise).

Figure 5.5: Comparison of mean output latency on the synthetic dataset (200 m noise).

Figure 5.6: Comparison of mean output latency on the synthetic dataset (300 m noise).

Figure 5.7: Comparison of mean output latency on the synthetic dataset (400 m noise).

Figure 5.8: Comparison of mean output latency on the synthetic dataset (500 m noise).

5.2 Optimization for Improving Computation Time

For the problem of map matching with path inference, algorithms based on probabilistic graphical models such as HMM have emerged as the most accurate solution. However, this accuracy comes at a substantial computational cost. It is widely recognized that the bulk of the computation time in map matching algorithms based on HMM or similar models is spent calculating the transition probabilities, which involves computing the shortest paths from all the states (i.e. candidate points) at a time step to all the states at the next time step [80, 107, 126]. In other words, the *many-to-many shortest path* (MSP) computation step is

the bottleneck in such map matching algorithms. The number of candidate points at each time step increases exponentially with the level of measurement noise in the input trajectory (because a larger error region needs to be considered). This means that the MSP computations become intractable for highly noisy trajectories. For instance, on the cellular dataset, the average time taken to process one trajectory on a high-end personal computer is 27 seconds out of which 24 seconds are spend on the MSP computation step. Clearly, such high computation times are unacceptable for real-time, large-scale map matching. In the following, works in the literature that dealt with this issue are first reviewed. Subsequently, an efficient algorithm is proposed for the MSP problem. The effectiveness of the proposed algorithm for accelerating the map matching of imprecise trajectories is evaluated.

5.2.1 Related Work

The standard way of solving the MSP problem from a set of source nodes to a set of target nodes in a graph is to perform a *single-source shortest path* (SSP) computation for each source node. This is generally performed by repeatedly running Dijkstra’s algorithm [64], which can find the shortest paths from a source node to all the nodes in the network. However, such a solution, which is referred to as the Dijkstra-based MSP algorithm in this thesis, incurs an unacceptably large computation time when the number of source nodes is high [127]. While a number of optimization techniques have been proposed in the literature for speeding up *one-to-one* shortest path computations in road networks [128], they are not directly applicable for MSP computations. It is generally counterproductive to transform a MSP problem between a set of source nodes S and a set of target nodes T into $|S| \times |T|$ one-to-one shortest-path computations, when the Dijkstra-based MSP algorithm can solve it by performing $|S|$ SSP computations.

Some researchers working on the map matching problem have proposed optimizations for reducing the computation time of the MSP computation step.

A commonly used optimization is to limit the search space of Dijkstra’s algorithm by the maximum possible distance that can be travelled from the source node within the time elapsed between successive time steps [107, 129]. However, this causes the search space to be much larger than actually required. A better alternative is to keep track of the target nodes *settled* by Dijkstra’s algorithm (explained in Appendix A) and to stop the algorithm once all the target nodes are settled. A method of terminating the shortest path search early based on the Viterbi path probabilities has been proposed in [126]. However, this method appears to be applicable only if the transition probabilities in the HMM are related to the shortest path distance in a certain way.

Heuristic pruning methods that limit the shortest path search space and the number of states at each time step have been used for accelerating probabilistic map matching algorithms. [34, 80, 129]. However, they do not guarantee optimality and in some cases, produce erroneous outputs. An optimization that reduces the number of transition probability calculations (and thus the number of SSP computations) without compromising optimality has been proposed in [108]. It involves reformulating the problem of finding the most likely state sequence in a HMM trellis graph as a one-to-one shortest path problem and finding the state sequence without having to explore all the states in the trellis graph. However, this optimization is not applicable for online map matching as it requires the complete HMM trellis graph to be available.

Some recent works speed up the MSP computation step by precomputing the shortest paths between all pairs of nodes in the road network that lie within a certain distance and storing the results in a hash table [130-132]. While such an approach has been reported to produce impressive speedups, two issues need to be noted. First, the above cited works were evaluated with GPS trajectories, where only a small number of candidate points are considered at each time step. In contrast, for highly noisy inputs such as cellular trajectories, it is common to have more than 1000 candidate points at a time step. In such a case, for each SSP computation, the precomputation approach would require 1000-plus queries to

the hash table, while a single run of Dijkstra algorithm would find the shortest paths to all the 1000-plus targets. Therefore, the benefits of precomputing the shortest paths appear uncertain for noisy trajectories. The second issue is that it is not practical to frequently repeat the all-pairs shortest path computation in response every change in the road network (e.g. road closures). Also, the precomputation approach is not suitable for scenarios where the shortest paths need to be computed on the basis of dynamic (i.e. time-varying) edge lengths.

The MSP problem is encountered not only in map matching, but also in other areas (e.g. in logistics applications). It has been studied by a few researchers as a standalone problem. Knopp et al. [127] applied a hierarchical acceleration technique to the MSP problem. While this solution is up to 3 orders of magnitude faster than the Dijkstra-based MSP algorithm for large problem instances, it has an important drawback: It involves an offline preprocessing step in which a highway hierarchy is constructed. As in the case of precomputing the shortest paths, the preprocessing step makes it impractical to apply this solution to road networks in which the connectivity and edge lengths change dynamically.

A solution to the MSP problem that does not involve any preprocessing has been proposed by Shibuya [133]. This solution first computes a bounded backward search space from all the target nodes and then uses a lower-bound estimate of the shortest path length to the set of target nodes to speed up the search from each source node. This yields a reduction in computation time of about 30% compared to the Dijkstra-based MSP algorithm for normal cases and about 70% when the source nodes and target nodes are located in two distant clusters. Another algorithm that solves the MSP problem without requiring any preprocessing has been proposed in [134]. This algorithm is based on the concept of *landmarks* [135], which helps to estimate a lower bound on the shortest-path distance between any node and a target node in order to direct the search towards the latter. The landmarks-based MSP algorithm yields a speed-up of up to four times for large problem instances when the source nodes and target nodes are confined to two clusters.

5.2.2 Algorithm for Clustered MSP Computation

In the proposed map matching algorithm as well as similar algorithms, the shortest paths need to be computed from a cluster of candidate points lying within the error region of a location measurement to another cluster of candidate points lying within the error region of the next location measurement. This problem, referred to as the clustered MSP problem, is formulated as follows.

As before, let $G = (V, E)$ be a directed and weighted graph representing a road network with a set of nodes V and a set of edges $E \subset V \times V$. Each edge $(u, v) \in E$ represents a road segment with a non-negative length $l(u, v)$. In this work, edge lengths represent the free-flow travel time of the road segment. The graph is represented using the compact forward and reverse star representation [136], which is a data structure that allows efficient computation of shortest paths in both the forward and the reverse directions. A path in the graph is a sequence of nodes $(u_1, u_2, \dots, u_n)$ such that $(u_i, u_{i+1}) \in E$ for all $i, 0 < i < n$. The length of a path is the sum of the lengths of the edges in it. The shortest-path distance $d(v, w)$ between nodes v and w is the length of the shortest path between them.

Let $S = \{s_1, s_2, \dots, s_m\}$, $S \subset V$, be a set of source nodes and $T = \{t_1, t_2, \dots, t_n\}$, $T \subset V$, be a set of target nodes such that $S \cap T = \emptyset$. All the source nodes in S lie within a bounded area A , which is referred to as the source region. (In the map matching context, the source region is nothing but the circular error region.) Let all the target nodes in T lie outside the source region A . The clustered MSP problem is to find the lengths of the shortest paths between all pairs of source nodes and target nodes. That is, the shortest-path distances $d(s_i, t_j)$ for all (i, j) , $0 < i \leq |S|$, $0 < j \leq |T|$ need to be computed. (In the proposed map matching algorithm and other similar algorithms, the composition of the shortest paths is not needed for the calculation of transition probabilities.)

The proposed algorithm for the clustered MSP problem is inspired by a simple observation: Shortest paths in a road network originating from a number of source nodes within a region typically exit the source region's boundary at a

relatively small number of points. This is partly due to the hierarchical nature of road networks, where a route originating from a minor road quickly joins one of a small number of nearby major roads and proceeds along the latter. In the following description, the last node that lies along a shortest path before it exits the source region is referred to as an *exit node*. In the context of clustered MSP computation, the shortest path from a source node to a target node can be divided into two parts: the shortest path between the source node and an exit node and the shortest path between that exit node and the target node. In line with this insight, the proposed algorithm involves the following high-level steps:

1. Identify the exit nodes in the source region and compute the shortest-path distances between each exit node and each target node.
2. Compute the shortest-path distances between each source node and each exit node.
3. Find the shortest-path distances between each source node and each target node based on the results of the above two steps.

Detailed description of the above high-level steps follows.

Step 1: Let V_A be the set of all nodes lying within the source region A . Let X be the set of exit nodes, which is initialized to be empty. Let $G' = (V', E')$ be an abstract graph with a set of nodes $V' = S \cup T \cup X$ and a set of edges $E' \subset V' \times V'$. At the start of the algorithm, the abstract graph G' has no nodes or edges, but they are incrementally added to it as the algorithm progresses. In order to identify the set of exit nodes, a set of all *potential exit nodes* $P = \{p_1, p_2, \dots, p_r\}$ is formed first. A node p is considered a potential exit node if there exists an edge $(p, q) \in E$ such that $p \in V_A, q \notin V_A$. In other words, a potential exit node is a node that lies within the source region and has an adjacent node that lies outside the source region as illustrated in Figure 5.9. Obviously, any shortest path originating from the source region A to a target node outside it must pass through one of the potential exit nodes. However, depending on the source nodes and target nodes, only a subset of the potential exit nodes serve as exit nodes. The actual exit nodes are identified as described below.

Figure 5.9: A simple grid network showing potential exit nodes (shaded) for a source region (within the dotted circle).

An SSP computation is performed with each potential exit node $p_i \in P$ as the source in order to find the shortest-path distances to all the target nodes in T . The SSP computation is performed using an augmented version of Dijkstra's algorithm that keeps track of the last potential exit node (LPEN) in the shortest path between the source and each node. Since the source p_i is a potential exit node, its LPEN value is set equal to itself (i.e. $LPEN[p_i] = p_i$). In Dijkstra's algorithm, a node is considered *settled* if the shortest path from the source to that node has been found. After a node u is settled, its LPEN value is updated as follows: If u is a potential exit node, $LPEN[u]$ is set equal to u . Else, the previous value of $LPEN[u]$ is retained. The SSP computation is terminated when all the target nodes are settled (i.e. the shortest-path distances from p_i to all the target nodes are found). Subsequently, for each target node $t_j \in T$, it is checked if $LPEN(t_j)$ is equal to p_i . If so, p_i is identified as an exit node and the following actions are performed:

- If $p_i \notin X$, add p_i to X .
- If $p_i \notin V'$, add p_i to V' .
- If $t_j \notin V'$, add t_j to V' .
- Add (p_i, t_j) to E' with edge length $l'(p_i, t_j)$ equal to $d(p_i, t_j)$, the shortest-path distance from p_i to t_j in graph G found by the SSP computation.

At the end of Step 1, the set X contains all the exit nodes for the set of source nodes S and the set of target nodes T . The abstract graph $G' = (V', E')$ has outgoing edges from each exit node to one or more target nodes.

Step 2: This step computes the shortest-path distances from each source node $s_i \in S$ to each exit node $x_j \in X$. In most instances of the clustered MSP problem, $|S| \gg |X|$. Therefore, it is more efficient to perform $|X|$ SSP computations in the reverse direction. Therefore, with each exit node $x_i \in X$ as the source, a reverse SSP computation is performed using a modified version of Dijkstra's algorithm. Unlike the standard Dijkstra's algorithm that expands the search space by exploring the outgoing edges of nodes starting from the source node, the modified version explores the incoming edges of nodes. The shortest path from node v to node u found by the reverse SSP computation is the same as the actual shortest path from node u to node v . The reverse SSP computation is terminated when all the nodes in $|S|$ are settled. Subsequently, for each node $s_j \in S$, the following actions are carried out:

- If $s_j \notin V'$, add s_j to V' .
- Add (s_j, x_i) to E' with edge length $l'(s_j, x_i)$ set equal to $d(s_j, x_i)$, the shortest-path distance from s_j to x_i in graph G found by the reverse SSP computation.

At the end of Step 2, the formation of the abstract graph G' is complete. G' can be viewed as a three-stage graph, with each stage consisting of nodes belonging to the sets S , X and T respectively as shown in Figure 5.10. Each node in X (the middle stage in G') has incoming edges from all the nodes in S (the first stage) and is connected to one or more nodes in T (the last stage) through outgoing edges. As noted earlier, the length of an edge (u, v) in G' is equal to the shortest-path distance between u and v in the original graph G . The abstract graph G' retains all the essential information needed to find the shortest-path distance from any source node in S to any target node in T .

Figure 5.10: Simplified representation of the abstract graph G' . Each edge represents the shortest path between its end nodes in the original graph G .

A point needs to be clarified here. The exit nodes, by definition, lie inside the source region and it is possible for exit nodes to be source nodes. In fact, in the MSP computation step in map matching, all the nodes inside the source region, including the exit nodes, are considered as source nodes (i.e. $X \subset S$). Hence, the actual organization of the abstract graph G' is somewhat different in practice with outgoing edges connecting each exit node with all other exit nodes. The three-stage graph in Figure 5.10 is a simplified representation that illustrates the concept clearly.

Step 3: This step finds the shortest-path distances in the abstract graph G' from each source node $s_i \in S$ to each target node $t_j \in T$. This is done by performing an SSP computation for each source node. While the standard Dijkstra's algorithm can be used for this purpose, it is slightly more efficient to use a customized version that takes advantage of the following observations:

- The shortest path in G' from any source node in S to any target node in T consists of at most two edges. Hence, for any source node s_i , it is sufficient to explore the outgoing edges of s_i and the outgoing edges of all the exit nodes connected to it.

- It is not necessary to maintain a priority queue of nodes and extract the minimum-distance node from it as the order in which the nodes are processed does not affect the final result.

For any pair of nodes, the length of the shortest path between them in G' is the same as the length of the shortest path between them in the original graph G , as shown in the proof below.

Proof of Correctness

Lemma 5.1. *For any source node $s_i \in S$ and target node $t_j \in T$, the shortest-path distance $d'(s_i, t_j)$ between them in the abstract graph G' is equal to the shortest-path distance $d(s_i, t_j)$ between them in the original graph G .*

Proof: Recall that P is the set of all potential exit nodes, which are nodes located within the source region A with an adjacent node outside it. Since the source node s_i lies inside A and the target node t_j lies outside it, the shortest path between them must exit A through a potential exit node $p_x \in P$ (i.e., p_x is the last potential exit node in the shortest path from s_i to t_j). Since p_x lies along the shortest path from s_i to t_j in G , $d(s_i, t_j) = d(s_i, p_x) + d(p_x, t_j)$.

In Step 1 of the algorithm, since p_x is the last potential exit node in the shortest path from p_x to t_j , p_x is identified as an exit node and the edge (p_x, t_j) with edge length $d(p_x, t_j)$ is added to G' . In Step 2, the shortest-path distance in G from each source node to each exit node is found and an edge representing the shortest path is added to G' . Therefore, it is certain that the edge (s_i, p_x) with edge length $d(s_i, p_x)$ is added to G' . It follows that there exists a path in G' from s_i to t_j comprising of edges (s_i, p_x) and (p_x, t_j) with path length $d(s_i, p_x) + d(p_x, t_j)$, which is equal to the shortest-path distance $d(s_i, t_j)$ in G . Since the set of nodes V' in G' is a subset of the set of nodes V in G and the edge lengths in G' represent the shortest-path distances in G , it is not possible for a shorter path from s_i to t_j to exist in G' with path length less than $d(s_i, t_j)$. Hence, the shortest-

path distance $d'(s_i, t_j)$ from s_i to t_j in the abstract graph G' is equal to shortest-path distance $d(s_i, t_j)$ in the original graph G .

Applying the Clustered MSP Algorithm for Map matching

The proposed clustered MSP algorithm can be used as subroutine for performing the MSP computation step in the map matching algorithm. Let o_t and o_{t+1} be two location measurements obtained at successive time steps t and $t + 1$, respectively. Let R be the error range of the location measurements. Let A_t and A_{t+1} be two circular error regions of radius R with o_t and o_{t+1} as their respective centers. The map-matching algorithm requires that the shortest-path distances from each candidate point inside the region A_t to each candidate point inside the region A_{t+1} be found.

Some details need to be explained here. A candidate point is the point that is closest to the location measurement on a road segment. As such, it may lie anywhere along the road segment, not necessarily on the nodes at either end of the road segment. However, only the nodes are considered as sources and targets for shortest path computation. An offset is maintained for each candidate point to indicate its position along the road segment and this offset is used to appropriately adjust the computed shortest-path distances. Also, it needs to be clarified that since edge lengths represent free-flow travel times in this work, the shortest-path distance between two nodes is the minimum free-flow travel time between them. However, for computing the transition probability, the proposed map matching algorithm also requires the physical length of the shortest path found based on the minimum-travel-time criterion. Therefore, the algorithms used in this work for SSP computation also track the physical lengths of the shortest paths. Doing so does not incur any significant additional computation.

The proposed clustered MSP algorithm cannot be directly used in the map matching algorithm due to the following reason. The formulation of the clustered MSP problem presented earlier in this section assumes that all the target nodes lie outside the source region. However, this is not always true in the map

matching context. It is common for error regions corresponding to two consecutive location measurements to overlap as shown in Figure 5.11, especially when the sampling interval is small and the error regions are large. In such a situation, some of the source nodes in the MSP computation are also target nodes. This issue is handled as follows: If the circular regions A_t and A_{t+1} do not overlap each other, the proposed clustered MSP algorithm is directly applied by considering all the nodes within A_t as the set of source nodes S and all the nodes

Figure 5.11: Error regions of consecutive location measurements may overlap.

within A_{t+1} as the set of target nodes T . However, if A_t and A_{t+1} partially overlap each other, a set of nodes $S' = S - T$ is formed. That is, S' is the set of source nodes in the portion of the source region A_t that is not overlapped by the target region A_{t+1} . The set of source nodes lying inside the overlapping region is denoted as $S'' = S \cap T$. The clustered MSP algorithm is used to find the shortest-path distances from each node in S' to each node in T . The conventional Dijkstra-based MSP algorithm is used for finding the shortest-path distances from each node in S'' to each node in T .

5.2.3 Evaluation

The optimized map matching algorithm that uses the proposed clustered MSP algorithm as described above is evaluated with the different trajectory datasets.

A version of the map matching algorithm that relies fully on the conventional Dijkstra-based MSP algorithm for the MSP computation step is used as the baseline. The two compared versions of the map matching algorithm are functionally equivalent (i.e., the map-matched paths are identical in all the experiments). They differ only in the running time. Figures 5.12 to 5.18 show the average time taken by the two versions for map matching one trajectory.

Figure 5.12: Comparison of running time on the cellular dataset.

Figure 5.13: Comparison of running time on the Wi-Fi-and-cellular dataset.

Figure 5.14: Comparison of running time on the synthetic dataset (100 m noise).

Figure 5.15: Comparison of running time on the synthetic dataset (200 m noise).

Figure 5.16: Comparison of running time on the synthetic dataset (300 m noise).

Figure 5.17: Comparison of running time on the synthetic dataset (400 m noise).

Figure 5.18: Comparison of running time on the synthetic dataset (500 m noise).

The proposed optimization based on the clustered MSP algorithm improves the running time at all sampling intervals and noise levels considered. However, the degree of improvement varies considerably with the sampling interval and noise level, and the running time savings range from 5% to 78%. For a given level of measurement noise, the speed-up produced by the optimization based on the clustered MSP algorithm increases with the sampling interval. One reason for this is that, at low sampling intervals, it is more likely for the error regions containing the sources and targets, respectively, to overlap each other, thereby reducing the usage of the clustered MSP algorithm. The results obtained for the synthetic datasets indicate that when the sampling interval is kept fixed, the speed-up increases with the noise level only up to a point and then declines. Based on closer analysis, it appears that the effectiveness of the clustered MSP algorithm depends on the number of source nodes and the separation between the error regions containing the sources and targets, respectively. It may be noted that if the sampling interval is kept fixed, the average separation between the error regions decreases with increasing noise, making it more likely for consecutive error regions to overlap.

An apparent anomaly in the results (Figures 5.12 to 5.18) needs to be explained. The baseline and optimized algorithms differ only in the way in which shortest paths are computed. The rest of the steps remain the same. However, in some cases, the times taken by the two algorithms for the rest of the steps are considerably different. This is likely due to cache memory issues. It appears that in some cases, the invocation of the clustered MSP algorithm causes data residing in the cache memory to be displaced, causing the program to slow down. This effect is more pronounced at smaller sampling intervals where a higher number of MSP computations are performed. It may be possible to eliminate or mitigate this issue by optimizing the source code in a cache-aware manner.

The landmarks-based MSP algorithm [134], which is the best-performing solution in the literature for the MSP problem without any preprocessing, was also evaluated as part of this study. However, it does not provide a consistent

improvement with respect to the Dijkstra-based MSP algorithm [35]. The landmarks-based MSP algorithm, which conforms to the goal-directed search approach, does not appear to be effective in the map matching context, where the source nodes and target nodes are generally not very distant from each other.

5.3 Summary

Output latency and computation speed are important considerations when map matching large-scale trajectory data in real time for smart mobility applications. There exists a need to devise novel techniques to improve the efficiency of the proposed map matching algorithm in order to enable its use for practical online applications. This need has been addressed to a significant extent in this chapter.

A simple yet effective technique has been presented for reducing the input-to-output latency of the online Viterbi algorithm used to find the most likely state sequence in a HMM or a similar model. This technique enables the early output of partial state sequences based on an estimate of the probability of a state being part of the eventual most likely sequence. Evaluations on real as well as synthetic trajectory datasets show that the proposed optimization produces an overall reduction in output latency of 33% with negligible loss of accuracy.

An efficient algorithm has been proposed for speeding up a bottleneck step present in probabilistic map matching algorithms, namely, the computation of many-to-many shortest paths between clusters of candidate locations. The proposed clustered MSP algorithm involves identifying a set of exit nodes in the region that encompasses the source nodes. The MSP computation is accelerated by breaking it down into two stages: computing the shortest paths between the exit nodes and the target nodes and doing the same between the source nodes and the exit nodes. An abstract graph representation is used to optimally combine the results of the above computations. When the clustered MSP algorithm is incorporated into the map matching algorithm, it yields a significant reduction of up to 78% in the computation time.

Chapter 6

Traffic Condition Estimation From Imprecise Trajectories

This chapter examines the effectiveness of the proposed map matching (i.e. path inference) solution for a high-impact smart mobility application, namely, traffic condition estimation from imprecise trajectory data. As wrongly-matched portions of the inferred paths lead to erroneous traffic condition estimates, a confidence measure for increasing the precision of map matching is proposed. A method of estimating the traffic condition of the links in the inferred paths, in terms of travel speed, is presented. The overall solution that includes the map matching and link speed estimation modules is evaluated with different levels of imprecision in the input trajectory data.

6.1 Overview of Traffic Condition Estimation

With the severity of the traffic congestion problem continuing to increase, real-time information about road traffic conditions has become essential for motorists in cities. According to a 2015 study, an urban commuter in the USA experiences, on average, travel delays of 42 hours every year due to traffic congestion [137]. In general, drivers who plan their route by considering the prevailing traffic conditions are less likely to experience travel delays compared to those who do not do so. Currently, many drivers use Web-based services providing live traffic information and traffic-aware driving directions such as Google Maps [138] or

Figure 6.1: Screen capture from Google Maps showing live traffic conditions.

Bing Maps [139]. Figure 6.1 shows a snapshot of the traffic conditions in a part of Singapore provided by Google Maps using a colour coding scheme.

The conventional approach towards traffic data collection is to use sensing technologies such as inductive loop detectors and video image detection systems, which have a number of drawbacks. Due to high installation and maintenance costs, they are generally not implemented throughout the road network, leading to low coverage. Also, they have several reliability issues and, in some cases, their performance is affected by adverse weather conditions [140]. The mass adoption of wirelessly connected mobile devices in the late 1990s led to the emergence of a technique known as Floating Car Data (FCD). In general, an FCD system receives some form of location sequence information from a wirelessly connected device inside a moving vehicle, infers the travelling path of the vehicle, and estimates the travel times or speeds along the traversed road segments. Estimates from a number of vehicles are aggregated to determine the traffic state of each road segment. FCD systems are seen as an attractive alternative to conventional traffic data collection methods, as the former can potentially provide much higher coverage at a low operational cost.

Most FCD systems rely on GPS trajectory data, obtained either from vehicle-based GPS devices (e.g. [141]) or from GPS-enabled mobile phones (e.g. [49]). There have also been a number of systems that use anonymized cellular *handover* data obtained from cellular operators to estimate traffic conditions. However, the latter systems generally have limited accuracy compared to GPS-based FCD systems and are not considered developed enough to the level needed for large scale utilization [142]. It needs to be noted that cellular handover data are temporally irregular, as they are generated only when a mobile phone moves from one cell to another during an ongoing call.

The feasibility of using smartphone-generated non-GPS trajectories for traffic condition estimation has not been adequately studied in the literature. However, as explained in Chapter 1, the use of such trajectories offer several advantages over GPS trajectories such as substantially higher energy efficiency, improved coverage and better privacy for the smartphone user providing the data. Furthermore, the consumption of the smartphone's battery power and bandwidth can be kept at negligible levels by sampling the locations very sparsely. This is particularly important for crowdsensing systems where users may not be willing to contribute data if they incur energy and monetary costs [16]. It is reasonable to suppose that the use of smartphone-generated coarse-grained trajectories would induce more users to participate, thereby leading to improved coverage of the traffic condition estimation system. Therefore, it is important to study if such non-GPS trajectories, which generally have a high degree of imprecision, are usable for the task of traffic condition estimation.

If smartphone-generated trajectories corresponding to non-vehicular modes of travel are detected [25, 31] and filtered out beforehand and only trajectories from vehicle-based users are considered, the process of estimating traffic conditions from them consists of the following steps [143, 144].

- (i) Map matching (with path inference)
- (ii) Travel time allocation.
- (iii) Travel time/speed aggregation

The first step matches each location measurement in an input trajectory to a point on the road network and identifies the path travelled by the vehicle. It can be effectively performed by the solution presented in the previous chapters. As the location measurements are sparsely sampled, it is common for a vehicle to traverse several road segments between two consecutive samples. Figure 6.2 shows an example where m_1 and m_2 denote two consecutive map-matched points on a road network and the dashed line indicates the part of the overall path that lies between them. The travel time between two consecutive map-matched points is nothing but the difference between the timestamps of the corresponding location measurements. The second step, travel time allocation, performs the task of allocating the observed travel time to individual links (i.e. road segments) in the traversed path. This is not a straightforward task because the link travel times depend on the attributes of the links such as the speed limit and the stopping times at signalized intersections [5, 143]. The last step, travel time/speed aggregation, produces an overall estimate of travel time/speed for each link by combining all the estimates for that link obtained from all the input trajectories.

In the traffic condition estimation system evaluated later in this chapter, only the first two of above-described steps are implemented. (The travel time aggregation step is relevant only if each link is traversed by several vehicles during

Figure 6.2: An example showing two consecutive map-matched points and the part of the overall path that lies between them (Base map: Bing Maps)

each time period considered. That would require a much larger trajectory dataset than the one used for the evaluations.) In this work, the output of the second step is expressed in terms of link running speeds, which are intuitively more meaningful than link travel times. The running speed of a link is the average speed maintained over that link while the vehicle is in motion. (i.e. The stopping delays at traffic signals are excluded from the speed calculation.) In most cases, the level of traffic congestion on a link can be correctly inferred from the estimated running speed. Thus, the traffic condition estimation system in this work can be seen as comprising of two components: a map matching module and a link speed estimation module as shown in Figure 6.3.

The *precision* of map matching is particularly important for traffic condition estimation. In this context, precision is the proportion of the inferred path that is correct. If a portion of the path inferred by the map matching algorithm is incorrect, it would lead to wrongly estimating travel speeds for road segments that were never actually traversed by the vehicle. Several researchers have noted the need to identify and discard incorrect portions of the map-matched path before estimating traffic conditions [13, 71]. Thiagarajan et al. [18] explored the idea of using a confidence predictor in order to improve the precision of map matching at the cost of its recall. In the next section, an effective confidence measure that helps to filter out likely incorrect portions of the map-matched path is presented.

The simplest way of allocating the observed travel time between two consecutive location measurements among the traversed links is to apportion it based on the link lengths, assuming that the vehicle travelled at a constant speed

Figure 6.3: Overview of the traffic condition estimation system used in this work.

throughout [71]. A somewhat more informed method is to allocate the observed travel time based on the free flow travel times of the links, which are generally based on their respective speed limits [145]. Hunter et al. [146] presented a probabilistic model of link travel times and an expectation maximization algorithm for learning the model parameters from a GPS trajectory dataset. Zheng and Van Zuylen [147] proposed a neural network model for link travel time estimation. They used simulation-based data to train the model and to evaluate its performance.

The above methods for link travel time estimation fail to separately account for the amount of time the vehicle might have spent stopping at signalized intersections. It is not hard to see that, in many cases, not accounting for the intersection delays would lead to gross errors in the estimation of the running speeds of the links. Hellinga et al. [143] proposed a model that represents the travel time of a link as three components: free flow travel time, delay due to congestion and stopping time. Their method of travel time allocation finds the most likely travel time for each link on a traversed path based on a likelihood function, whose parameters are empirically set. Jenelius et al. [5] represented the travel time along a path as a combination of the running travel times along links and the delays at intersections. They proposed a statistical model for link travel time estimation that considers a number of variables including link attributes and trip conditions. The parameters of the model are estimated from a GPS trajectory dataset using maximum likelihood.

Similar to [143] and [5], the method of estimating link travel times proposed in this work treats the stopping delays at intersections as separate from the running times along the links. However, it is simpler and more generic compared to the methods cited above. While the model in [5] relies on a number explanatory variables including those representing trip conditions and a corresponding number of parameters, the proposed method requires only minimal information about the road network. Also, it overcomes some of the constraints of [143] such as the unrealistic assumption that a vehicle will stop at

most once along the path traversed between two consecutive location measurements. The proposed method is explained in detail in Section 6.3.

6.2 A Confidence Measure for Map Matching

It is worth recalling that the proposed map matching solution finds the most likely state sequence in a HMM trellis graph using an online variant of the Viterbi algorithm. The online Viterbi algorithm outputs a partial state sequence from time to time with an output latency of at least one time step. Once the partial state sequence up to a certain time step is output, the stages of the HMM trellis graph up to that time step can be deleted as they are no longer needed for further computations. Each state in the HMM represents a candidate point on a road segment. A partial state sequence is transformed into a path in the road network by interpolating the gap between each pair of successive states (i.e. on-road points) in the sequence with the optimal path between them.

Let the HMM trellis graph maintained at a point of time cover T time steps. Let $o_{1:T} = \{o_1, \dots, o_T\}$ be the corresponding sequence of observations (i.e. location measurements). Let $\gamma_t(s_{t,i}) = P(s_{t,i} | o_{1:T})$ represent the posterior probability of the state $s_{t,i}$ at time step t , given the observation sequence $o_{1:T}$, $1 \leq t < T$. It can be expressed as follows [76].

$$\gamma_t(s_{t,i}) = \frac{\alpha_t(s_{t,i})\beta_t(s_{t,i})}{\sum_{j=1}^{N_t} \alpha_t(s_{t,j})\beta_t(s_{t,j})} \quad (6.1)$$

where $\alpha_t(s_{t,i})$ and $\beta_t(s_{t,i})$, respectively, are known as the forward probability and backward probability, which are defined shortly. N_t is the number of states at time step t . The denominator in Equation 6.1 is a normalization term that can be ignored for the task at hand, namely, the calculation of a confidence measure. It is convenient to use the unnormalized probability $\gamma'_t(s_{t,i})$ defined below.

$$\gamma'_t(s_{t,i}) = \alpha_t(s_{t,i})\beta_t(s_{t,i}) \quad (6.2)$$

The forward probability $\alpha_t(s_{t,i})$ represents the joint probability of the state $s_{t,i}$ at time step t and the observation sequence $o_{1:t}$. The forward probability for the states in the first time step is initialized as $\alpha_1(s_{1,i}) = P(o_1|s_{1,i})$, for all i . For the states in the subsequent time steps, the forward probability is recursively calculated as

$$\alpha_t(s_{t,k}) = P(o_t|s_{t,k}) \sum_{j=1}^{N_{t-1}} P(s_{t,k}|s_{t-1,j}) \alpha_{t-1}(s_{t-1,j}) \quad (6.3)$$

where $P(o_t|s_{t,k})$ and $P(s_{t,k}|s_{t-1,j})$, respectively, represent the emission and transition probabilities in the HMM. The backward probability $\beta_t(s_{t,i})$ is the probability of the partial observation sequence $o_{t+1:T}$ after time t , given the state $s_{t,i}$, at time step t . The backward probability for all the states in the final time step T is defined as 1. That is, $\beta_T(s_{T,i}) = 1$, for all i . For the states in the previous time steps, it is recursively calculated as

$$\beta_t(s_{t,k}) = \sum_{j=1}^{N_{t+1}} P(s_{t+1,j}|s_{t,k}) P(o_{t+1}|s_{t+1,j}) \beta_{t+1}(s_{t+1,j}) \quad (6.4)$$

Let it be supposed that based on the observation sequence $o_{1:T}$, a partial state sequence $s_{1:u}^* = \{s_1^*, \dots, s_u^*\}$, $1 < u < T$, is generated by the online Viterbi algorithm as output. Let p_u be the partial path in the road network corresponding to the partial state sequence $s_{1:u}^*$. Let s_t^* denote the state in the partial state sequence $s_{1:u}^*$ at time step t , $1 \leq t \leq u$. This means that s_t^* represents the on-road point to which the observation o_t is map matched. (To be clear, $s_t^* = s_{t,i}$ for some value of i .) Let c_t be a confidence measure associated with s_t^* . The confidence measure c_t is based on the posterior probability of the state s_t^* at time step t , given the observation sequence $o_{1:T}$. That is, c_t depends on $\gamma_t'(s_t^*)$, which is calculated using Equations 6.2 to 6.4. Let $\hat{s}_t$ be another state at time step t such that the candidate on-road point represented by it does not lie along the path p_u and $\gamma_t'(\hat{s}_t)$ is the highest among all the states at time step t representing candidate

on-road points not lying along the path p_u . The confidence measure c_t associated with s_t^* is defined as follows.

$$c_t = \frac{\gamma'_t(s_t^*)}{\gamma'_t(\hat{s}_t)} \quad (6.5)$$

The rationale behind the choice of the above confidence measure is easy to see. If the probability of the map-matched on-road point at a particular time step, given the sequence of observations up to a future time step, is substantially higher than that of any alternative candidate points, it is reasonable to be very confident about the map-matched point. The confidence measure defined in Equation 6.5 captures this relationship. By comparing the confidence measure of the map-matched points against a suitable threshold, likely erroneous parts of the path inferred by the map matching algorithm can be identified and discarded in order to improve the precision of the algorithm.

6.3 Link Speed Estimation Method

The basic unit of output provided by the map matching module (i.e. the input of the link speed estimation module) consists of a pair of consecutive map-matched points and the sub-path traversed by the vehicle between them. More specifically, the input to the link speed estimation module contains the following information:

- Two consecutive map-matched points, m_{t-1} and m_t , respectively, corresponding to time steps $t - 1$ and t . Each map-matched point m_t is specified as a tuple $\langle r_i, o_t, c_t \rangle$, where r_i is the directed road segment on which m_t lies, o_t is an offset that specifies the position of m_t along the road segment and c_t is the confidence measure associated with m_t .
- A sub-path p_t along the road network between the map-matched points m_{t-1} and m_t . The sub-path p_t is specified as a sequence of connected road segments $\{r'_1, \dots, r'_n\}$, where r'_i represents the portion of the road segment r_i that is covered by p_t .

Figure 6.4: An example of the input to the link speed estimation method.

Figure 6.4 shows an example of the input provided to the link speed estimation module. The dashed line in the figure indicates the sub-path p_t and the dark circles represent the intersections between road segments. For clarity, the other roads that join at the intersections are not shown.

The link speed estimation method described below processes an input such as the one shown in Figure 6.4 only if the confidence measure of both m_{t-1} and m_t (i.e. c_{t-1} and c_t) are above a certain threshold. Otherwise, the input is discarded.

Let δ_t denote the known time interval between time steps $t - 1$ and t . This is, of course, the same as the travel time from m_{t-1} and m_t along the sub-path p_t . This known travel time can be seen as being composed of two components as expressed below:

$$\delta_t = \tau(p_t) + \omega(p_t) \quad (6.6)$$

where $\tau(p_t)$ represents the total running travel time along road segments r'_1 through r'_n that make up the sub-path p_t and $\omega(p_t)$ represents the total stopping delay, due to traffic signals at the intersections, along the sub-path. Both $\tau(p_t)$ and $\omega(p_t)$ are unknown. A method of estimating the total stopping delay $\omega(p_t)$ is presented in Section 6.3.1. If δ_t and $\omega(p_t)$ are known, it is trivial to determine

the total running travel time $\tau(p_t)$ based on Equation 6.6. The process of allocating $\tau(p_t)$ among the constituent links (i.e. road segments) of sub-path p_t and estimating the running speeds of the links is described in Section 6.3.2.

6.3.1 Stopping Delay Estimation

The actual stopping delays at signalized intersections depend on a number of factors such as prevailing traffic flows and the traffic signal cycles. The amount of time a vehicle would spend stopping at a particular intersection is subject to a stochastic process. Furthermore, there are substantial differences in the stopping delays associated with different turning manoeuvres. Given the large number of intersections in a typical urban road network, it is not practical to estimate and use a separate stopping delay distribution for each intersection, for each turning manoeuvre. This difficulty is handled in this work by grouping all the intersections in the road network into the following 3 broad types: major intersections, minor intersections and hybrid intersections. In each of the above intersection types, it is possible to make one of the following turning manoeuvres: left turn, right turn and going straight. (U-turns, which occur infrequently, are grouped together with right turns. The road network used in this work conforms to the left-hand traffic system.) Therefore, a total of 9 combinations of intersection types and turning manoeuvres are considered.

An offline preprocessing step is used to identify the intersection type and turning manoeuvre associated with each pair of directly connected road segments in the road network in the following manner. Based on the road-class attribute, each road segment is first classified as either major (e.g. expressways and arterial roads) or minor (e.g. residential streets). If all the road segments meeting at an intersection are major, then it is considered a major intersection. An intersection with all minor roads is a minor intersection, and one where both major and minor roads meet is a hybrid intersection. The turning manoeuvre involved between two road segments is determined based on the angle made by them at the intersection. The intersection type and turning manoeuvre for each valid pair of

directly-connected road segments are thus predetermined and stored in a table that is accessed by the link speed estimation module during runtime.

For any combination of intersection type and turning manoeuvre, the *mean* stopping delay can be estimated from a densely sampled, ground-truth annotated trajectory dataset as explained in Section 6.4. Let $\hat{\omega}(r_j, r_k)$ denote the mean stopping delay at the intersection between two directly-connected road segments r_j and r_k . It is determined based on the intersection type and turning manoeuvre of the pair of road segments (r_j, r_k) . For instance, if the two road segments connect at a major intersection and involve a right turn, then $\hat{\omega}(r_j, r_k)$ is set equal to the pre-estimated mean stopping delay of the major-intersection-and-right-turn combination.

Given a sub-path p_t made up of connected road segments $\{r'_1, \dots, r'_n\}$, a simple way of estimating the total stopping delay along it is to sum up the mean stopping delays between each pair of consecutive road segments in the sub-path (i.e. $\sum_{i=1}^{n-1} \hat{\omega}(r_i, r_{i+1})$). However, doing so overlooks an important issue: The vehicle might have stopped (for the traffic signal) along the first road segment (i.e. r_1) *before* the map-matched point m_{t-1} . (It is not uncommon for a vehicle to stop at a considerable distance before an intersection due to a long queue.) In such a case, the mean stopping delay at the first intersection along p_t should *not* be considered while estimating the total stopping delay. Similarly, the vehicle might have stopped along the last road segment (i.e. r_n) before the map-matched point m_t . In such a case, the mean stopping delay at the intersection at the downstream end of road segment r_n should also be considered even though it is not covered by the sub-path p_t .

It is not possible to be certain if the vehicle stopped before or after a particular point along a road segment. Therefore, this work relies on a probabilistic model that defines the probability of a vehicle stopping for the traffic signal at a point whose position along a road segment is specified by its distance to the downstream end of the road segment (i.e. its distance to the upcoming intersection). For a map-matched point m_k whose position along road segment

r_i is specified by an offset o_k from the start of the road segment, its distance to the upcoming intersection is denoted by $o'_k = l(r_i) - o_k$, where $l(r_i)$ is the length of r_i . It is generally assumed that a vehicle is more likely to stop closer to the downstream end of a road segment than farther from it [143]. However, analysis of traffic signal stops observed in real trajectory data shows that while the above assumption is generally true, vehicles are actually less likely stop extremely close to the downstream end of a road segment. This is because the stop line is generally situated several meters before the point where the centreline of the road segment meets the centrelines of the other road segments that join at the intersection. The distribution of stopping probabilities, as a function of the distance to the upcoming intersection o'_k , broadly follows a lognormal distribution such as the one shown in Figure 6.5. It is given by

$$f_o(o'_k) = \frac{1}{o'_k \sigma \sqrt{2\pi}} \exp\left(-\frac{(\log(o'_k) - \mu)^2}{2\sigma^2}\right) \quad (6.7)$$

where μ and σ , respectively, are the mean and standard deviation of $\log(o'_k)$ to be estimated. (Here, $\log(o'_k)$ denotes the natural logarithm of o'_k .)

Given a map-matched point m_k whose distance to the upcoming intersection is o'_k , it is useful to calculate the probability of a vehicle stopping between m_k and the upcoming intersection. This probability is given by the

Figure 6.5: Stopping probability as a function of distance to the upcoming intersection.

cumulative distribution function (CDF) of the lognormal distribution, which is defined as

$$F_o(o'_k) = \frac{1}{2} \operatorname{erfc} \left(- \frac{\log(o'_k) - \mu}{\sigma\sqrt{2}} \right) \quad (6.7)$$

where $\operatorname{erfc}(\cdot)$ is the complementary error function.

As before, let m_{t-1} and m_t be two consecutive map-matched points connected by a sub-path p_t made up of connected road segments $\{r'_1, \dots, r'_n\}$. (It may be helpful to refer to Figure 6.4, where $n = 5$). Let $\hat{\omega}(r_i, r_{i+1})$ denote the mean stopping delay at the intersection between r_i and r_{i+1} . Based on the above discussion, the total stopping delay along p_t is estimated as follows if $n > 1$.

$$\omega(p_t) = F_o(o'_{t-1}) \hat{\omega}(r_1, r_2) + \sum_{i=2}^{n-1} \hat{\omega}(r_i, r_{i+1}) + (1 - F_o(o'_t)) \hat{\omega}(r_n, *) \quad (6.8)$$

The first term on the right hand side of Equation 6.8 represents an estimate of the stopping delay encountered at the first intersection that follows m_{t-1} . The second term sums up the mean stopping delays at the subsequent intersections along the sub-path p_t . The third term is an estimate of the possible stopping delay encountered along road segment r_n (i.e. before m_t) as a result of stopping for the traffic signal at the next intersection. The symbol $*$ in the term indicates that the next road segment after r_n is not known at this point. Hence, the turning manoeuvre the vehicle intends to make at the next intersection is also not known. In such a situation, it is reasonable to take the average of the stopping delays associated with all the 3 turning manoeuvres for that intersection type. $\hat{\omega}(r_n, *)$ represents this value.

A special case arises when the sub-path p_t consists of a single road segment (i.e. $n = 1$). In such a case, both the map-matched points m_{t-1} and m_t lie on the same road segment r_1 as shown in Figure 6.6. This calls for the probability of the vehicle stopping between m_{t-1} and m_t to be considered along with the mean

Figure 6.6: Two consecutive map-matched points on the same road segment.

stopping delay at the intersection at the downstream end of road segment r_1 . Thus, the total stopping delay along p_t for case where $n = 1$ is

$$\omega(p_t) = (F_o(o'_{t-1}) - F_o(o'_t)) \hat{\omega}(r_n, *) \quad (6.8)$$

6.3.2 Travel Time Allocation and Speed Estimation

Given δ_t , the elapsed time between map-matched points m_{t-1} and m_t , and $\omega(p_t)$, the total stopping delay along the sub-path p_t between those map-matched points, the total running travel time along p_t is

$$\tau(p_t) = \delta_t - \omega(p_t) \quad (6.9)$$

In this work, $\tau(p_t)$ is apportioned among the road segments $\{r'_1, \dots, r'_n\}$ covered by p_t on the basis of their free flow travel times. The free flow travel time of road segment r'_i is defined as

$$q(r'_i) = \frac{l(r'_i)}{c(r'_i)} \quad (6.10)$$

where $l(r'_i)$ is the length of r'_i and $c(r'_i)$ is the speed limit of r'_i . The free flow travel time of the sub-path p_t is simply the sum of the free flow travel times of all the road segments covered by it.

$$q(p_t) = \sum_{i=1}^n q(r'_i) \quad (6.11)$$

The running travel time along road segment r'_i is calculated as

$$\tau(r'_i) = \frac{q(r'_i)}{q(p_t)} \tau(p_t) \quad (6.12)$$

The running speed of road segment r'_i is

$$v(r'_i) = \frac{l(r'_i)}{\tau(r'_i)} \quad (6.13)$$

It needs to be recalled that r'_i represents the portion of the road segment r_i that is covered by sub-path p_t . It is possible that another portion of r_i is covered by a different sub-path generated at the previous or subsequent time step. Therefore, it is common to have multiple running speed estimates for the same road segment. Let $v_t(r_k)$ denote the running speed estimate produced by the proposed method for road segment r_k at time step t . Let $l_t(r_k)$ denote the length of the portion of r_k that is used for estimating $v_t(r_k)$. The overall running speed of the road segment is estimated by calculating the length-based weighted average.

$$\hat{v}(r_k) = \frac{\sum_t l_t(r_k) v_t(r_k)}{\sum_t l_t(r_k)} \quad (6.14)$$

It is worth noting that the estimate of the total stopping delay $\omega(p_t)$ used in Equation 6.9 is based on the *mean* stopping delays along the sub-path. Of course, this estimate could be higher or lower than the actual total stopping delay encountered by the vehicle. In particular, overestimation of $\omega(p_t)$ would lead to underestimation of the total running travel time $\tau(p_t)$, which in turn would lead to the running speeds of the road segment to be overestimated, sometimes excessively so. Therefore, an upper bound is placed on the running speed estimates given by Equation 6.14. This upper bound is set to be 20% above the speed limit of the road segment, as motorists are generally unlikely to drive at a speed higher than that.

6.4 Evaluation

The experimental evaluations in this chapter consist of the following parts, each of which are described in the subsequent subsections.

- Evaluation of the effectiveness of the proposed confidence measure for improving the precision of map matching.
- Estimation of mean stopping delays and the parameters of the stopping probability distribution from a real trajectory dataset.
- Evaluation of the overall traffic condition estimation system under different levels of sparsity and noise in the input trajectory data.

6.4.1 Evaluation of the Confidence Measure

The effectiveness of the confidence measure for filtering out likely incorrect sub-paths produced by the map matching module is evaluated with the 5 synthetic-noise trajectory datasets. The first step is to empirically determine a suitable threshold such that sub-paths associated with confidence measures lower than the threshold are discarded. This is done by analysing the trade-offs between the precision and recall of map matching observed with different values of the threshold. Without confidence-based filtering, the map matching algorithm achieves an overall precision of 93.45% and a recall of 91.02%, when the results are aggregated across all noise levels and sampling intervals. Table 6.1 shows the precision and recall values, as well as the percentage change with respect to the above-mentioned numbers, achieved with different settings of the confidence measure threshold. It can be seen that increasing the threshold produces a consistent increase in precision, but at a substantial cost of recall. However, a high degree of map matching precision is necessary for traffic condition estimation and therefore the loss of recall needs to be tolerated. Based on these results, a threshold value of 1.1 is chosen, as a higher threshold only produces a marginal increase in precision.

In the context of traffic condition estimation, it is more useful to express the error of the map matching step in terms of the *false inference rate*, which is defined as the proportion of the inferred path that is *incorrect*. As precision is the proportion of the inferred path that is correct, the false inference rate is equal to

1 minus precision. (The false inference rate is analogous to the concept of false discovery rate used in null hypothesis testing.) Figures 6.7 to 6.11 show the false inference rate, expressed as a percentage, for each of the synthetic-noise datasets at different sampling intervals, with and without confidence-based filtering. It can be seen that confidence-based filtering produces a consistent and significant reduction in the false inference rate. The highest false inference rate of 12.5% is observed at the highest level of imprecision in the input trajectory data (i.e. 500 m noise and 300-second sampling interval.) This worst-case value corresponds to a map matching precision of 87.5%. According to one study [18], a median precision of 85% or above is considered useful for accurate traffic condition estimation. The results indicate that the proposed confidence measure helps to improve the precision of the map matching algorithm to a level acceptable for traffic condition estimation, despite using highly imprecise trajectories as input.

Confidence measure threshold	Precision(%)	% Change in precision	Recall (%)	% Change in recall
0.8	94.38	1.00%	82.43	-9.44%
0.9	94.65	1.29%	80.34	-11.73%
1.0	95.10	1.77%	75.82	-16.70%
1.1	96.75	3.54%	61.56	-32.37%
1.2	96.85	3.64%	54.59	-40.03%

Table 6.1: Precision and recall of map matching for different values of confidence measure threshold, averaged across all noise levels and sampling intervals.

Figure 6.7: Evaluation of the confidence measure on the synthetic dataset (100 m noise)

Figure 6.8: Evaluation of the confidence measure on the synthetic dataset (200 m noise).

Figure 6.9: Evaluation of the confidence measure on the synthetic dataset (300 m noise).

Figure 6.10: Evaluation of the confidence measure on the synthetic dataset (400 m noise).

Figure 6.11: Evaluation of the confidence measure on the synthetic dataset (500 m noise).

6.4.2 Estimation of Mean Stopping Delays and Stopping Probability Parameters

The mean stopping delays at different types of intersections for different turning manoeuvres are estimated using the paths dataset that contains 1000 ground-truth annotated GPS trajectories sampled mostly at the rate of one point per second. The dataset also contains GPS-derived instantaneous speeds for each GPS location in the trajectories. This makes it possible to calculate the total time spent stopping at each intersection along each trajectory and find the mean stopping delay for each combination of intersection type and turning manoeuvre. Based on actual stopping delays observed in about 14,000 instances of traversing an intersection, the mean stopping delays have been found as given in Table 6.2.

	Left turn	Right turn	Straight
Major intersection	8.93	23.04	14.11
Minor intersection	3.80	7.39	3.20
Hybrid intersection	7.82	17.40	3.93

Table 6.2: Mean stopping delays in seconds for different combinations of intersection types and turning manoeuvres, estimated from a real trajectory dataset.

The parameters μ and σ of the lognormal distribution given by Equation 6.7, which defines the stopping probability, have been estimated based on the distances between the traffic signal stops observed in paths dataset and their respective upcoming intersections. The parameters μ and σ have been estimated to be 3.727 and 0.993, respectively.

6.4.3 Evaluation of the Traffic Condition Estimation System

The overall traffic condition estimation system, which consists of the map matching and link speed estimation modules, has been implemented in C++ and evaluated using the synthetic-noise trajectory datasets at sampling intervals ranging from 60 seconds to 300 seconds. The link speed estimation module processes the path inferred by the map matching algorithm in a piecewise manner, with each piece consisting of a pair of consecutive map-matched points and the sub-path between them. For each road segment in the inferred path, an estimate of the running speed is produced. The true running speed experienced at each road segment is calculated from the ground truth information associated with each trajectory. It may be recalled that the ground truth information includes the true location and speed of the vehicle at every second. The accuracy of the running speed estimates generated by the proposed system is evaluated in terms of the mean absolute error (MAE), in line with the practice followed by several works on traffic condition estimation (e.g. [49], [116] and [148]). In this context, the MAE is simply equal to the average of the absolute differences between the true running speed and the estimated running speed, considering all the road segments in all the trajectories.

It can be seen from the description of the link speed estimation method in Section 6.3 that the error in the estimation of the stopping delays at signalized intersections is the major source of error in the estimates of running speeds. Intuitively, it appears that the running speeds estimated for expressways, which do not have signalized intersections, would have lower errors compared to other roads with traffic signals. Therefore, it is useful to evaluate the above two

categories of roads separately. Tables 6.3, 6.4 and 6.5, respectively, present the MAE, in km/h, of the traffic condition estimation system for expressways, non-expressway roads and all roads. Each column in the tables corresponds to a noise level in the input trajectory data. Each row corresponds to a sampling interval.

	Noise: 100 m	Noise: 200 m	Noise: 300 m	Noise: 400 m	Noise: 500 m
Sampling interval: 60 s	6.63	8.34	9.59	10.35	10.50
Sampling interval: 120 s	6.96	7.33	8.00	9.11	9.19
Sampling interval: 180 s	7.69	8.75	10.27	9.17	9.57
Sampling interval: 240 s	7.21	6.99	7.11	10.19	8.12
Sampling interval: 300 s	8.09	8.63	8.38	9.81	9.02

Table 6.3: Mean absolute errors (km/h) of running speed estimates for expressways.

	Noise: 100 m	Noise: 200 m	Noise: 300 m	Noise: 400 m	Noise: 500 m
Sampling interval: 60 s	15.90	18.05	18.11	20.08	20.22
Sampling interval: 120 s	16.31	15.39	17.15	17.48	19.05
Sampling interval: 180 s	15.50	14.74	16.46	17.03	16.88
Sampling interval: 240 s	15.96	14.71	17.66	16.37	19.56
Sampling interval: 300 s	16.49	15.74	13.12	16.78	20.46

Table 6.4: Mean absolute errors (km/h) of running speed estimates for non-expressway roads.

	Noise: 100 m	Noise: 200 m	Noise: 300 m	Noise: 400 m	Noise: 500 m
Sampling interval: 60 s	11.28	12.65	13.02	13.63	13.93
Sampling interval: 120 s	11.05	10.40	11.11	11.65	11.72
Sampling interval: 180 s	10.72	10.69	12.20	11.53	11.81
Sampling interval: 240 s	10.56	9.72	9.74	11.85	9.33
Sampling interval: 300 s	10.88	10.86	9.77	11.66	11.40

Table 6.5: Mean absolute errors (km/h) of running speed estimates for all roads.

As expected, the MAE values for expressways are significantly lower than those of non-expressway roads. While MAE values for expressways range from 6.6 to 10.5 km/h, the corresponding values for non-expressway roads range from 14.7 to 20.5 km/h. Considering both types of roads together, the errors range from 9.7 to 13.9 km/h, with the overall average being 11.3 km/h.

The effect of the imprecision in the input trajectory data on the quality of the traffic condition estimates is interesting. Broadly speaking, for a given sampling interval, the speed estimation errors appear to increase moderately with increasing measurement noise in the input data. This can be explained by the fact that higher noise levels generally translate into higher errors in the location of the map-matched points, even if the overall path is inferred correctly. The errors in the location of the map-matched points introduce errors in the lengths of the sub-paths between them, which affect accuracy of the running speed estimates (Equation 6.13). For a given noise level, increasing the sampling interval does not produce a consistent increase in the error. In fact, the overall error at the 60-second sampling interval is higher than the error observed at longer sampling intervals. There are two possible reasons for this. First, the errors in the lengths of the sub-paths caused by the measurement noise, become less significant (as a proportion of the length) when longer sampling intervals are considered. Second, at longer sampling intervals, the vehicle is likely to pass through more intersections between two consecutive measurements, thereby making it more likely that errors in the estimation of stopping delays at individual intersections cancel out each other.

The primary question here is this: Are the observed errors in the estimation of road-segment-wise running speeds, tabulated above, acceptable for applications that rely on road traffic condition data? To answer this, one needs to consider the fact that exact speed estimates are not essential for most applications that consume traffic data. Some studies have concluded that errors in speed estimates of up to 20% are acceptable for traveler information applications [149]. This is reflected in the granularity of traffic speed data disseminated by various

data providers. For instance, the Land Transport Authority of Singapore provides real-time data on traffic speeds only in the form of speed bands such as 0 to 20 km/h, 20 to 40 km/h and so on [150]. Based on the above points and knowledge of the typical speeds on different types of roads, it could be inferred that most of the MAE values shown in Tables 6.3 to 6.5 are within the acceptable range.

Furthermore, an important point needs to be noted. The system described in this chapter produces a running speed estimate for a road segment based on a single trajectory corresponding to a single vehicle. In a real-world traffic condition estimation system, trajectory data is obtained from a much larger number of vehicles so that each road segment is traversed by several vehicles during each time period (e.g. a 5-minute interval). In such a case, multiple estimates of the running speed of the road segment are generated and the average of those estimates is disseminated as output. If positive and negative errors are equally likely, then they tend to cancel out when averaged. Therefore, it is important to examine the *distribution* of the estimation errors. The histogram in Figure 6.12 shows the distribution of the estimation errors for all road segments under all experimental conditions. It can be seen that the estimation errors roughly follow a normal distribution with zero mean. (Specifically, the mean and standard deviation of the distribution are -0.5 km/h and 15.5 km/h, respectively.) If the estimation errors are thus distributed, the error in the average of several estimates will tend to be close to zero. This indicates that the proposed traffic

Figure 6.12: Histogram of the running-speed estimation error.

condition estimation system, when used with a large quantum of input trajectory data, is capable of achieving high accuracy despite the high levels of measurement noise and temporal sparsity in the data.

6.5 Summary

Trajectory-based traffic condition estimation is arguably the most important application that relies on map matching. Accurate and timely data about the road traffic conditions is highly useful to drivers who seek to avoid congested routes and minimize the travel time. The idea of estimating traffic conditions, such as road-segment-wise running speeds, from coarse-grained non-GPS trajectories is attractive because it offers several advantages over the standard option of relying on GPS trajectories. This chapter has evaluated the feasibility of using the proposed map matching solution, along with imprecise trajectory data, for traffic condition estimation. Towards this goal, a system consisting of a map matching module and a link speed estimation module has been presented.

As the precision of map matching is particularly important for traffic condition estimation, a confidence-based filtering approach has been adopted to improve precision at the cost of recall. An effective confidence measure, based on the posterior probabilities of map-matched points given a sequence of location observations, has been proposed. Experiments performed under different levels of noise and sparsity in the trajectory data show that the confidence-based filtering method produces a consistent and significant reduction in the proportion of the map-matched path that is incorrect.

A method of estimating the running speeds of the road segments in each partial path output generated by the map matching module has been proposed. This method involves a detailed procedure for estimating the total stopping delay at signalized intersections along the partial path. This procedure relies on mean stopping delays for different combinations of intersection types and turning manoeuvres estimated from a real ground-truth labelled trajectory dataset. It also

depends on a model that gives the probability of a vehicle stopping at a location specified by the distance to the upcoming intersection. The total running travel time along the partial path is allocated among the road segments on the basis of their free flow travel times in order to calculate their running speeds. A detailed experimental evaluation of the accuracy of the running speed estimates has been conducted. Analysis of the mean absolute errors observed under various conditions and the distribution of the errors indicates that the proposed system is capable of generating acceptably accurate traffic condition estimates from substantially imprecise trajectory data.

Chapter 7

Conclusions and Future Work

7.1 Conclusions

This thesis is a step towards unlocking the full potential of trajectory data, generated by ubiquitous smart mobile devices in copious amounts, for enabling smart mobility in urban transport networks. It presents effective and efficient techniques for map matching imprecise trajectories generated by alternatives to the standard option of GPS such as Wi-Fi positioning and cellular network positioning, which, despite their high degree of inaccuracy, are preferable to GPS due to a number of reasons. These reasons include the substantially higher energy consumption of GPS, its unavailability in areas such as urban canyons and tunnels, and the privacy risks associated with sharing GPS trajectories. Guided by a critical study of the existing body of research on the topic of map matching, this work makes several significant contributions to the topic, as recapitulated below.

A probabilistic algorithm based on a hidden Markov model has been proposed for map matching sparse, coarse-grained trajectories in an online manner (Chapter 3). The emission and transition probability models in the HMM have been carefully designed in order to achieve a high degree of robustness against the imprecision in the input data. In particular, the proposed transition probability model overcomes several shortcomings of existing HMM-based map matching algorithms such as the reliance on the shortest-distance path between candidate points, flawed measures of circuitousness and temporal consistency,

and susceptibility to the label-bias problem. Experimental evaluation with real as well as synthetic-noise trajectory datasets shows that the accuracy of the proposed algorithm is substantially better than a widely-used baseline algorithm [79] at all levels of measurement noise and sparsity, with the map matching error of the former less than half of the latter in most cases. It is worth noting that the superior accuracy of the proposed HMM-based algorithm is not achieved at the cost of its output latency and running time, which remain comparable to the baseline algorithm.

A novel approach based on a route choice model has been developed for overcoming the inability of the HMM-based map matching framework, which permits only first-order dependencies among states, to consider route context information (Chapter 4). It also helps to overcome a restriction in HMM-based map matching, which allows the transition between two candidate points to occur only through the optimal path between them. In the proposed approach, a multinomial-logit route choice model is used to assess the HMM-generated partial paths from the perspective of route choice. A choice set generation procedure has been proposed to generate a number of reasonable alternative paths between any two points on a road network. The route choice model, which gives the probability of a path being chosen among a set of alternatives based on several path attributes, has been estimated from a real trajectory dataset. Complementing the HMM-based map matching algorithm with the route choice model produces a notable improvement in accuracy, except at relatively low levels of noise and sparsity. The results are consistent with the intuitive observation that HMM-based map matching is more likely to produce unrealistic paths as output at higher degrees of imprecision in the input data.

Considerable research effort has been directed towards improving the timeliness and computational efficiency of the developed map matching solution in order to meet the requirements of real-time smart mobility applications (Chapter 5). An optimization technique that enables early generation of partial state sequences in the HMM has been presented. This technique relies on an

estimate of the probability of a state being part of the eventual most likely sequence. It reduces the overall output latency by about one-third with negligible loss of accuracy. The output latency can be reduced further at the cost of some loss of accuracy by changing a threshold parameter. In order to improve the computation time of the map matching algorithm, an innovative method has been proposed to speed-up its bottleneck step, namely, the many-to-many shortest path computation between clusters of candidate points. This method relies on an interesting insight: shortest paths originating from a cluster of source nodes typically pass through a small number of exit nodes, which can be used to subdivide and simplify the computation of shortest paths. When this new method of computing many-to-many shortest paths is incorporated into the map matching algorithm, a major reduction in the total computation time is observed. The proposed optimization techniques can also be used to improve the efficiency of other state-of-the-art map matching algorithms that are based on a probabilistic graphical model such as HMM or CRF.

A prototype traffic condition estimation system that relies on the proposed map matching algorithm has been developed and evaluated in this work (Chapter 6). It incorporates several salient features including the ability to filter out likely incorrect portions of the map-matched path based on a confidence measure and a method of estimating the total stopping delay at the intersections between a pair of map-matched points. The traffic condition estimation system, which estimates the running speed along each road segment in the map-matched path, has been evaluated under different levels of temporal sparsity and measurement noise. Despite using imprecise trajectory data as input, the accuracy of the running speed estimates has been found to be generally within the range considered acceptable for applications that consume traffic data. While, in general, the accuracy of the estimated running speeds degrades moderately with increasing noise levels, the sampling interval does not have a consistent effect on the accuracy. Furthermore, the distribution of the estimation errors (i.e. a zero-mean normal distribution) suggests that the overall accuracy would greatly improve when each road segment is traversed by several vehicles, with positive

and negative errors cancelling out each other when averaged. This makes the proposed traffic condition estimation system particularly suitable for projects involving large scale trajectory data.

The above findings lend support to a broad conclusion: Coarse-grained smartphone trajectory data, despite being substantially inferior in terms of positioning accuracy and temporal density, can still be utilized for smart mobility applications. With over 70% of the population in many developed countries owning a smartphone [151], an immense amount of crowdsourced trajectory data is potentially obtainable. However, many users may be willing to share data only if the data sampling is done without imposing any significant energy, bandwidth or privacy costs [16]. Against this background, the use of sparsely-sampled non-GPS trajectories, made feasible by the proposed map matching solution, is likely to lead to increased user participation, which is crucial for the success of crowdsourcing-based systems.

It can be confidently stated that this thesis advances the state of the art in map matching, particularly for noisy and sparse trajectories. While the vast majority of the map matching literature is focussed on GPS trajectories, this is one of the few works targeting non-GPS trajectories with a much higher degree of measurement noise. This is the first known work that augmented a probabilistic-graphical-model based map matching framework by incorporating route choice criteria into it. The map matching accuracies achieved in this research are superior to the accuracies reported by other works in the literature that dealt with comparable levels of measurement noise (e.g. [18] and [27]). Despite being oriented towards non-GPS trajectories, the proposed techniques are nevertheless applicable to GPS trajectories as well. It is envisaged that the high levels of map matching accuracy and efficiency achieved in this work would lead to more effective utilisation of trajectory data in order to improve urban mobility.

7.2 Future Work

It is, of course, possible to further develop many aspects of the work presented in this thesis and broaden its scope. A few future research directions for extending this work are briefly discussed below.

- **Reducing uncertainty using additional sensor data:** This research considered only minimal trajectory data, with each location measurement consisting of a latitude, longitude and a timestamp. Some works in the literature have demonstrated that the map matching accuracy of HMM-based algorithms can be significantly improved by considering additional data from the smartphone's inertial sensors such as accelerometer and gyroscope [18, 95]. While such sensor data may not be always available, it may, however, be useful to have the ability to utilize such data (when available) in order to reduce the uncertainty in the input trajectory data. Therefore, it is worth devising an effective method of optionally incorporating observations from inertial sensors into the proposed HMM framework.
- **Mining historical trajectory data for improved map matching:** Given a large historical trajectory dataset where the ground truth path of each trajectory is known with reasonable certainty, it is possible to mine that dataset and extract useful information about travel patterns. For instance, the most commonly travelled path between certain locations could be extracted and stored in order to aid the map matching process [152]. It would be interesting to examine ways in which such travel pattern information can be used to augment the transition probability model and the route choice model used in this work.
- **Extending the solution to multimodal travel:** Transport authorities in many cities such as Singapore [3] are strongly promoting the use of public transport, while discouraging car ownership. Against this background, a large proportion of smartphone-generated trajectories may correspond to

multimodal journeys, involving several modes such as bus, train, taxi, bicycle and walk. The capability to accurately infer the travelled paths from such trajectories is useful for many purposes, including travel demand estimation and travel behaviour analysis. The challenging problem of map matching multimodal trajectories has received some research attention in recent years [29, 153]. While the map matching algorithm proposed in this work is designed for unimodal trajectories corresponding to car travel, extending it to work with multimodal trajectories would be a valuable contribution.

- **Developing an adaptive location sampling method:** All the experiments in this thesis were carried out with trajectories such that the raw locations in each trajectory are subject to the same type of measurement noise and are sampled at a fixed rate. However, in practice, a situation-aware adaptive sampling strategy may help to achieve an optimal trade-off between the map matching accuracy and the energy costs incurred by the smartphone user [154]. For instance, in a dense and complex part of the road network where it is difficult to infer the path of a vehicle with confidence, it may be beneficial to reduce the sampling interval or temporarily use a more accurate positioning technology in order to resolve the ambiguity. It is useful to develop an effective confidence-based method for dynamically altering the sampling scheme and study the accuracy-energy trade-offs.
- **Developing a unified probabilistic model for map matching:** The proposed map matching solution relies on a two-stage approach involving a HMM and a route choice model. A more elegant approach would be to develop a unified probabilistic model that is capable of capturing the benefits of both the HMM and the route choice model. In theory, a higher-order probabilistic graphical model such as a higher-order CRF, which permits the modelling of long term context dependencies spanning several time steps, can be used for the above purpose. Doing so would make it

possible to represent route preference criteria within the probabilistic graphical model framework. However, such an approach has not yet been explored in the literature. Some researchers (e.g. [101]) have acknowledged the merit of using a higher-order CRF for map matching, while pointing out that it is quite complicated to realise such a model. Nevertheless, it is a worthy research idea to explore.

References

- [1] Centre for Economics and Business Research, "The future economic and environmental costs of gridlock in 2030," London, 2014.
- [2] United Nations Department of Economic and Social Affairs, "The World's Cities in 2016 - Data Booklet," 2016.
- [3] Land Transport Authority, "Smart Mobility 2030 - ITS Strategic Plan for Singapore," Singapore, 2014.
- [4] Y. Zheng, "Trajectory data mining: an overview," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 6, p. 29, 2015.
- [5] E. Jenelius and H. N. Koutsopoulos, "Travel time estimation for urban road networks using low frequency probe vehicle data," *Transportation Research Part B: Methodological*, vol. 53, pp. 64-81, 2013.
- [6] M. Lv, L. Chen, X. Wu, and G. Chen, "A road congestion detection system using undedicated mobile phones," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, pp. 3060-3072, 2015.
- [7] E. Kazagli, J. Chen, and M. Bierlaire, "Individual Mobility Analysis Using Smartphone Data," in *Mobile Technologies for Activity-Travel Data Collection and Analysis*, S. Rasouli and H. Timmermans, Eds., ed: IGI Global, , 2014.
- [8] N. Dhakar and S. Srinivasan, "Route choice modeling using GPS-based travel surveys," *Transportation Research Record: Journal of the Transportation Research Board*, pp. 65-73, 2014.
- [9] J. Yuan, Y. Zheng, X. Xie, and G. Sun, "T-drive: Enhancing driving directions with taxi drivers' intelligence," *IEEE Transactions on Knowledge and Data Engineering*, vol. 25, pp. 220-232, 2013.
- [10] W. Luo, H. Tan, L. Chen, and L. M. Ni, "Finding time period-based most frequent path in big trajectory data," in *Proceedings of the 2013 ACM SIGMOD international conference on management of data*, 2013, pp. 713-724.

- [11] J. Eisner, S. Funke, A. Herbst, A. Spillner, and S. Storandt, "Algorithms for Matching and Predicting Trajectories," in *ALLENEX*, 2011, pp. 84-95.
- [12] D. Qiu, P. Papotti, and L. Blanco, "Future locations prediction with uncertain data," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 2013, pp. 417-432.
- [13] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk, "On map-matching vehicle tracking data," in *Proceedings of the 31st international conference on very large data bases*, 2005, pp. 853-864.
- [14] P. A. Zandbergen, "Accuracy of iPhone locations: A comparison of assisted GPS, WiFi and cellular positioning," *Transactions in GIS*, vol. 13, pp. 5-25, 2009.
- [15] Android Developers. (2017, 9 December 2017). *Location Strategies*. Available: <https://developer.android.com/guide/topics/location/strategies.html>
- [16] R. K. Ganti, F. Ye, and H. Lei, "Mobile crowdsensing: current state and future challenges," *IEEE Communications Magazine*, vol. 49, 2011.
- [17] I. Constandache, S. Gaonkar, M. Sayler, R. R. Choudhury, and L. Cox, "Enloc: Energy-efficient localization for mobile phones," in *INFOCOM 2009, IEEE*, 2009, pp. 2716-2720.
- [18] A. Thiagarajan, L. Ravindranath, H. Balakrishnan, S. Madden, and L. Girod, "Accurate, low-energy trajectory mapping for mobile devices," in *Proceedings of the 8th USENIX conference on Networked systems design and implementation*, 2011, pp. 20-20.
- [19] Y. Tsuda, Q. Kong, and T. Maekawa, "Detecting and correcting WiFi positioning errors," in *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing*, 2013, pp. 777-786.
- [20] P. Zhou, Y. Zheng, and M. Li, "How long to wait?: predicting bus arrival time with mobile phone based participatory sensing," in *Proceedings of the 10th international conference on Mobile systems, applications, and services*, 2012, pp. 379-392.
- [21] C. R. Vicente, D. Freni, C. Bettini, and C. S. Jensen, "Location-related privacy in geo-social networks," *IEEE Internet Computing*, vol. 15, pp. 20-27, 2011.
- [22] M. Wernke, P. Skvortsov, F. Dürr, and K. Rothermel, "A classification of location privacy attacks and approaches," *Personal and ubiquitous computing*, vol. 18, pp. 163-175, 2014.

- [23] C.-Y. Chow and M. F. Mokbel, "Trajectory privacy in location-based services and data publication," *ACM Sigkdd Explorations Newsletter*, vol. 13, pp. 19-29, 2011.
- [24] C. A. Ardagna, M. Cremonini, E. Damiani, S. D. C. Di Vimercati, and P. Samarati, "Location privacy protection through obfuscation-based techniques," in *IFIP Annual Conference on Data and Applications Security and Privacy*, 2007, pp. 47-60.
- [25] M. Mun, D. Estrin, J. Burke, and M. Hansen, "Parsimonious mobility classification using GSM and WiFi traces," in *Proceedings of the Fifth Workshop on Embedded Networked Sensors (HotEmNets)*, 2008.
- [26] Y. Kanza, "Uncertainty in geosocial data: friend or foe?," *SIGSPATIAL Special*, vol. 8, pp. 3-10, 2016.
- [27] G. Schulze, C. Horn, and R. Kern, "Map-matching cell phone trajectories of low spatial and temporal accuracy," in *Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on*, 2015, pp. 2707-2714.
- [28] R. Mohamed, H. Aly, and M. Youssef, "Accurate real-time map matching for challenging environments," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, pp. 847-857, 2017.
- [29] F. Asgari, A. Sultan, H. Xiong, V. Gauthier, and M. A. El-Yacoubi, "CT-Mapper: mapping sparse multimodal cellular trajectories using a multilayer transportation network," *Computer Communications*, vol. 95, pp. 69-81, 2016.
- [30] E. Algizawy, T. Ogawa, and A. El-Mahdy, "Real-Time Large-Scale Map Matching Using Mobile Phone Data," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 11, p. 52, 2017.
- [31] T. Sohn, A. Varshavsky, A. LaMarca, M. Chen, T. Choudhury, I. Smith, *et al.*, "Mobility detection using everyday GSM traces," *UbiComp 2006: Ubiquitous Computing*, pp. 212-224, 2006.
- [32] G. R. Jagadeesh and T. Srikanthan, "Probabilistic map matching of sparse and noisy smartphone location data," in *Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on*, 2015, pp. 812-817.
- [33] G. R. Jagadeesh and T. Srikanthan, "Online Map-Matching of Noisy and Sparse Location Data With Hidden Markov and Route Choice Models," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, pp. 2423-2434, 2017.

- [34] G. R. Jagadeesh and T. Srikanthan, "Heuristic optimizations for high-speed low-latency online map matching with probabilistic sequence models," in *Intelligent Transportation Systems (ITSC), 2016 IEEE 19th International Conference on*, 2016, pp. 2565-2570.
- [35] G. R. Jagadeesh and T. Srikanthan, "Fast Computation of Clustered Many-to-Many Shortest Paths and its Application to Map Matching," *ACM Transactions on Spatial Algorithms and Systems*, 2018 (Under review).
- [36] G. R. Jagadeesh and T. Srikanthan, "Robust real-time route inference from sparse vehicle position data," in *Intelligent Transportation Systems (ITSC), 2014 IEEE 17th International Conference on*, 2014, pp. 296-301.
- [37] K. Chin and C. Lee, "TrafficScan-bringing real-time travel information to motorists," *Journeys*, pp. 7-14, 2009.
- [38] J. Yuan, Y. Zheng, C. Zhang, W. Xie, X. Xie, G. Sun, *et al.*, "T-drive: driving directions based on taxi trajectories," in *Proceedings of the 18th SIGSPATIAL International conference on advances in geographic information systems*, 2010, pp. 99-108.
- [39] M. Rahmani, H. N. Koutsopoulos, and A. Ranganathan, "Requirements and potential of GPS-based floating car data for traffic management: Stockholm case study," in *Intelligent Transportation Systems (ITSC), 2010 13th International IEEE Conference on*, 2010, pp. 730-735.
- [40] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, *et al.*, "Big data: The next frontier for innovation, competition, and productivity," McKinsey Global Institute, 2011.
- [41] H. Almuhimedi, F. Schaub, N. Sadeh, I. Adjerid, A. Acquisti, J. Gluck, *et al.*, "Your location has been shared 5,398 times!: A field study on mobile app privacy nudging," in *Proceedings of the 33rd annual ACM conference on human factors in computing systems*, 2015, pp. 787-796.
- [42] J. Willaredt, "WiFi and Cell-ID based positioning-protocols, standards and solutions," Technische Universität Berlin, 2011.
- [43] E. Kaplan and C. Hegarty, *Understanding GPS: principles and applications*: Artech house, 2005.
- [44] United States Federal Aviation Administration, "Global Positioning System Standard Positioning Service Performance Analysis Report," 96, 2017.
- [45] F. van Diggelen and P. Enge, "The worlds first gps mooc and worldwide laboratory using smartphones," in *Proceedings of the 28th International Technical Meeting of The Satellite Division of the Institute of Navigation (ION GNSS+ 2015)*, 2015, pp. 361-369.

- [46] WiGLE. (12 December 2017). *WiGLE: Wireless Network Mapping*. Available: <https://wigle.net/>
- [47] J. Zhou, K.-K. Chu, and J.-Y. Ng, "Providing location services within a radio cellular network using ellipse propagation model," in *Advanced Information Networking and Applications, 2005. AINA 2005. 19th International Conference on*, 2005, pp. 559-564.
- [48] E. Trevisani and A. Vitaletti, "Cell-ID location technique, limits and benefits: an experimental study," in *Mobile computing systems and applications, 2004. WMCSA 2004. Sixth IEEE workshop on*, 2004, pp. 51-60.
- [49] A. M. Bayen and A. D. Patire, *Mobile Millennium final report*: California Center for Innovative Transportation, Institute of Transportation Studies, University of California, Berkeley, 2011.
- [50] A. Allström, J. Archer, A. Bayen, S. Blandin, J. Butler, D. Gundlegård, *et al.*, "Mobile Millennium Stockholm," in *2nd International Conference on Models and Technologies for Intelligent Transportation Systems*. Leuven, Belgium, 2011.
- [51] T. Abdulazim, H. Abdelgawad, K. M. N. Habib, and B. Abdulhai, "Using Smartphones and Sensor Technologies to Automate Collection of Travel Data," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2383, pp. 44-52, 2013.
- [52] C. E. White, D. Bernstein, and A. L. Kornhauser, "Some map matching algorithms for personal navigation assistants," *Transportation research part c: emerging technologies*, vol. 8, pp. 91-108, 2000.
- [53] M. A. Quddus, W. Y. Ochieng, L. Zhao, and R. B. Noland, "A general map matching algorithm for transport telematics applications," *GPS solutions*, vol. 7, pp. 157-167, 2003.
- [54] S. Syed and M. E. Cannon, "Fuzzy logic-based map matching algorithm for vehicle navigation system in urban canyons," in *ION National Technical Meeting, San Diego, CA*, 2004, pp. 26-28.
- [55] M. Fu, J. Li, and M. Wang, "A hybrid map matching algorithm based on fuzzy comprehensive judgment," in *Intelligent Transportation Systems, 2004. Proceedings. The 7th International IEEE Conference on*, 2004, pp. 613-617.
- [56] 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*, vol. 15, pp. 312-328, 2007.

- [57] 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*, vol. 48, pp. 153-165, 2014.
- [58] Google. (12 December 2017). *Google Scholar*. Available: <https://scholar.google.com/>
- [59] J. Krumm, J. Letchner, and E. Horvitz, "Map matching with travel time constraints," in *SAE World Congress*, 2007.
- [60] 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*, vol. 15, pp. 507-529, 2014.
- [61] J.-s. Yang, S. Kang, and K.-s. Chon, "The map matching algorithm of GPS data with relatively long polling time intervals," *Journal of the Eastern Asia Society for Transportation Studies*, vol. 6, pp. 2561-2573, 2005.
- [62] Y. Zheng and M. A. Quddus, "Weight-based shortest-path aided map-matching algorithm for low-frequency positioning data," in *Transportation Research Board 90th Annual Meeting*, 2011.
- [63] H. Yin and O. Wolfson, "A weight-based map matching method in moving objects databases," in *Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on*, 2004, pp. 437-438.
- [64] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische mathematik*, vol. 1, pp. 269-271, 1959.
- [65] H. Alt and M. Godau, "Computing the Fréchet distance between two polygonal curves," *International Journal of Computational Geometry & Applications*, vol. 5, pp. 75-91.
- [66] 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*, 2013, pp. 434-437.
- [67] T. Griffin, Y. Huang, and S. Seals, "Routing-based map matching for extracting routes from GPS trajectories," in *Proceedings of the 2nd International Conference on Computing for Geospatial Research & Applications*, 2011, p. 24.
- [68] D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 10, pp. 112-122, 1973.

- [69] M. Rahmani and H. N. Koutsopoulos, "Path inference from sparse floating car data for urban networks," *Transportation Research Part C: Emerging Technologies*, vol. 30, pp. 41-54, 2013.
- [70] G. Sarlas, "Processing low-frequency floating car data for transportation applications," Master's thesis, KTH Royal Institute of technology, 2013.
- [71] L. Giovannini, "A novel map-matching procedure for low-sampling GPS data with applications to traffic flow analysis," PhD thesis, Università di Bologna, 2011.
- [72] M. Srivatsa, R. Ganti, J. Wang, and V. Kolar, "Map matching: facts and myths," in *Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, 2013, pp. 474-477.
- [73] B. Kuijpers, B. Moelans, W. Othman, and A. Vaisman, "Uncertainty-based map matching: the space-time prism and k-shortest path algorithm," *ISPRS International Journal of Geo-Information*, vol. 5, p. 204, 2016.
- [74] T. Miwa, D. Kiuchi, T. Yamamoto, and T. Morikawa, "Development of map matching algorithm for low frequency probe data," *Transportation Research Part C: Emerging Technologies*, vol. 22, pp. 132-145, 2012.
- [75] D. Koller, N. Friedman, and F. Bach, *Probabilistic graphical models: principles and techniques*: MIT press, 2009.
- [76] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, pp. 257-286, 1989.
- [77] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, pp. 260-269, 1967.
- [78] A. Thiagarajan, "Probabilistic models for mobile phone trajectory estimation," Phd thesis, Massachusetts Institute of Technology, 2011.
- [79] 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*, 2009, pp. 336-343.
- [80] 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*, 2009, pp. 352-361.

- [81] A. Thiagarajan, L. Ravindranath, K. LaCurts, S. Madden, H. Balakrishnan, S. Toledo, *et al.*, "VTrack: accurate, energy-aware road traffic delay estimation using mobile phones," in *Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems*, 2009, pp. 85-98.
- [82] R. Raymond, T. Morimura, T. Osogami, and N. Hirose, "Map matching with hidden Markov model on sampled road network," in *Pattern Recognition (ICPR), 2012 21st International Conference on*, 2012, pp. 2242-2245.
- [83] C. Goh, J. Dauwels, N. Mitrovic, M. Asif, A. Oran, and P. Jaillet, "Online map-matching based on Hidden Markov model for real-time traffic sensing applications," in *Intelligent Transportation Systems (ITSC), 2012 15th International IEEE Conference on*, 2012, pp. 776-781.
- [84] 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*, 2012, pp. 605-608.
- [85] 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*, 2014, pp. 401-404.
- [86] T. Osogami and R. Raymond, "Map Matching with Inverse Reinforcement Learning," in *IJCAI*, 2013, pp. 2547-2553.
- [87] Y. Yin, R. R. Shah, and R. Zimmermann, "A general feature-based map matching framework with trajectory simplification," in *Proceedings of the 7th ACM SIGSPATIAL International Workshop on GeoStreaming*, 2016, p. 7.
- [88] 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*, 2013.
- [89] M. Li, A. Ahmed, and A. J. Smola, "Inferring movement trajectories from GPS snippets," in *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining*, 2015, pp. 325-334.
- [90] P. Szwed and K. Pekala, "An Incremental Map-Matching Algorithm Based on Hidden Markov Model," in *Artificial Intelligence and Soft Computing*, 2014, pp. 579-590.
- [91] A. Oran and P. Jaillet, "An HMM-based map matching method with cumulative proximity-weight formulation," in *Connected Vehicles and Expo (ICCVE), 2013 International Conference on*, 2013, pp. 480-485.

- [92] J. Yuan, Y. Zheng, C. Zhang, X. Xie, and G.-Z. Sun, "An interactive-voting based map matching algorithm," in *Mobile Data Management (MDM), 2010 Eleventh International Conference on*, 2010, pp. 43-52.
- [93] A. Javanmard, M. Haridasan, and L. Zhang, "Multi-track map matching," in *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, 2012, pp. 394-397.
- [94] B. Y. Chen, H. Yuan, Q. Li, W. H. Lam, S.-L. Shaw, and K. Yan, "Map-matching algorithm for large-scale low-frequency floating car data," *International Journal of Geographical Information Science*, vol. 28, pp. 22-38, 2014.
- [95] 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*, 2015, p. 5.
- [96] K. Kempinska, T. Davies, J. Shawe-Taylor, and P. Longley, "Probabilistic Map-Matching for Low-Frequency GPS Trajectories," in *Proceedings of GIS Ostrava*, 2017, pp. 209-221.
- [97] C. Song and X. Yan, "Hidden Markov Model for Floating Car Trajectory Map Matching," in *Advances in Computer Science and Ubiquitous Computing*, ed: Springer, 2017, pp. 553-559.
- [98] X. Zhou, Y. Ding, H. Tan, Q. Luo, and L. M. Ni, "HIMM: An HMM-Based Interactive Map-Matching System," in *International Conference on Database Systems for Advanced Applications*, 2017, pp. 3-18.
- [99] J. Lafferty, A. McCallum, and F. C. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *18th International Conference on Machine Learning 2001*, 2001, pp. 282-289.
- [100] C. Sutton and A. McCallum, "An introduction to conditional random fields," *Foundations and Trends® in Machine Learning*, vol. 4, pp. 267-373, 2012.
- [101] M. Xu, Y. Du, J. Wu, and Y. Zhou, "Map Matching Based on Conditional Random Fields and Route Preference Mining for Uncertain Trajectories," *Mathematical Problems in Engineering*, vol. 2015, 2015.
- [102] J. Yang and L. Meng, "Feature Selection in Conditional Random Fields for Map Matching of GPS Trajectories," in *Progress in Location-Based Services 2014*, ed: Springer, 2015, pp. 121-135.
- [103] X. Liu, K. Liu, M. Li, and F. Lu, "A ST-CRF Map-Matching Method for Low-Frequency Floating Car Data," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, pp. 1241-1254, 2017.

- [104] E. Fosler-Lussier, Y. He, P. Jyothi, and R. Prabhavalkar, "Conditional random fields in speech, audio, and language processing," *Proceedings of the IEEE*, vol. 101, pp. 1054-1075, 2013.
- [105] M. Bierlaire, J. Chen, and J. Newman, "A probabilistic map matching method for smartphone GPS data," *Transportation Research Part C: Emerging Technologies*, vol. 26, pp. 78-98, 2013.
- [106] B. S. Westgate, "Vehicle travel time distribution estimation and map-matching via Markov chain monte carlo Methods," Phd thesis, Cornell University, 2013.
- [107] 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*, 2012, pp. 613-616.
- [108] H. Koller, P. Widhalm, M. Dragaschnig, and A. Graser, "Fast Hidden Markov Model Map-Matching for Sparse and Noisy Trajectories," in *Proceedings of the 2015 IEEE 18th International Conference on Intelligent Transportation Systems*, 2015, pp. 2557-2561.
- [109] 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*, 2014, pp. 33-42.
- [110] K. Gade, "A non-singular horizontal position representation," *The journal of navigation*, vol. 63, pp. 395-417, 2010.
- [111] M. Narasimhan, P. Viola, and M. Shilman, "Online decoding of markov models under latency constraints," in *Proceedings of the 23rd international conference on Machine learning*, 2006, pp. 657-664.
- [112] Y. Wang, Y. Zhu, Z. He, Y. Yue, and Q. Li, "Challenges and opportunities in exploiting large-scale GPS probe data," *HP Laboratories, Technical Report HPL-2011-109*, vol. 21, 2011.
- [113] P. J. Rousseeuw and C. Croux, "Alternatives to the median absolute deviation," *Journal of the American Statistical association*, vol. 88, pp. 1273-1283, 1993.
- [114] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the royal statistical society. Series B (methodological)*, pp. 1-38, 1977.

- [115] Quantum Inventions. (28 February 2018). *Solutions & Services for Intelligent Transport, Dynamic Navigation and Real-Time Traffic Data*. Available: <http://www.quantuminventions.com/>
- [116] H. Li, L. Kulik, and K. Ramamohanarao, "Robust inferences of travel paths from GPS trajectories," *International Journal of Geographical Information Science*, vol. 29, pp. 2194-2222, 2015.
- [117] M. Ben-Akiva and M. Bierlaire, "Discrete choice methods and their applications to short term travel decisions," in *Handbook of transportation science*, ed: Springer, 1999, pp. 5-33.
- [118] D. Eppstein, "Finding the k shortest paths," *SIAM Journal on computing*, vol. 28, pp. 652-673, 1998.
- [119] D. Park and L. Rilett, "Identifying multiple and reasonable paths in transportation networks: A heuristic approach," *Transportation Research Record: Journal of the Transportation Research Board*, pp. 31-37, 1997.
- [120] M. Ben-Akiva, M. Bergman, A. J. Daly, and R. Ramaswamy, "Modeling inter-urban route choice behaviour," in *Proceedings of the 9th international symposium on transportation and traffic theory*, 1984, pp. 299-330.
- [121] K. E. Train, "Properties," in *Discrete choice methods with simulation*, ed: Cambridge university press, 2009.
- [122] M. E. Ben-Akiva and S. R. Lerman, "Multinomial Choice," in *Discrete choice analysis: theory and application to travel demand*. vol. 9, ed: MIT press, 1985.
- [123] M. Bierlaire, "BIOGEME: a free package for the estimation of discrete choice models," in *3rd Swiss Transport Research Conference*, Ascona, Switzerland, 2003.
- [124] J. Bobbin, "An incremental Viterbi algorithm," in *Sixth International Conference on Machine Learning and Applications*,, 2007, pp. 104-111.
- [125] R. Šrámek, "The on-line Viterbi algorithm," Master's thesis, Comenius University, Bratislava, 2007.
- [126] T. Imamichi, T. Osogami, and R. Raymond, "Truncating Shortest Path Search for Efficient Map-Matching," in *International Joint Conference on Artificial Intelligence*, 2016, pp. 589-595.
- [127] S. Knopp, P. Sanders, D. Schultes, F. Schulz, and D. Wagner, "Computing many-to-many shortest paths using highway hierarchies," in *Proceedings of the Meeting on Algorithm Engineering & Experiments*, 2007, pp. 36-45.

- [128] H. Bast, D. Delling, A. Goldberg, M. Müller-Hannemann, T. Pajor, P. Sanders, *et al.*, "Route planning in transportation networks," *Algorithm Engineering*, pp. 19-80, 2016.
- [129] 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*, 2011, pp. 221-230.
- [130] Z. Zeng, T. Zhang, H. Zou, and Z. Wu, "Acceleration of Map Matching for Floating Car Data by Exploiting Travelling Velocity," in *Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on*, 2015, pp. 2895-2899.
- [131] C. Yang and G. z. Gidófalvi, "Fast map matching, an algorithm integrating hidden Markov model with precomputation," *International Journal of Geographical Information Science*, vol. 32, pp. 547-570, 2018.
- [132] L. Yang, X. Zhao, W. Zhang, F. Meng, X. Cheng, and Y. An, "An Improved Map-Matching Method Based on Hidden Markov Model," in *Proceedings of the 2nd International Conference on Information Technology and Intelligent Transportation Systems (ITITS 2017), Xi'an, China, June 10, 2017*, 2017, p. 266.
- [133] T. Shibuya, "Computing the nxm shortest path efficiently," *Journal of Experimental Algorithmics (JEA)*, vol. 5, p. 9, 2000.
- [134] S. Knopp, "Efficient computation of many-to-many shortest paths," *Diplomarbeit, Universität Karlsruhe*, 2006.
- [135] A. V. Goldberg and C. Harrelson, "Computing the shortest path: A search meets graph theory," in *Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, 2005, pp. 156-165.
- [136] R. Ahuja, T. Magnanti, and J. Orlin, "Network flows: theory, algorithms, and applications," *Prentice-Hall Inc.*, 1993.
- [137] D. Schrank, B. Eisele, T. Lomax, and J. Bak, "2015 urban mobility scorecard," *Texas A&M Transportation Institute*, 2015.
- [138] Google. (28 February 2018). *Google Maps*. Available: <https://maps.google.com/>
- [139] Microsoft. (28 February 2018). *Bing Maps - Directions, trip planning, traffic cameras & more*. Available: <https://www.bing.com/maps>
- [140] C. Antoniou, R. Balakrishna, and H. N. Koutsopoulos, "A synthesis of emerging data collection technologies and their impact on traffic

- management applications," *European Transport Research Review*, vol. 3, pp. 139-148, 2011.
- [141] C. de Fabritiis, R. Ragona, and G. Valenti, "Traffic Estimation And Prediction Based On Real Time Floating Car Data," in *Intelligent Transportation Systems, 2008. ITSC 2008. 11th International IEEE Conference on*, 2008, pp. 197-203.
 - [142] J. Steenbruggen, M. T. Borzacchiello, P. Nijkamp, and H. Scholten, "Mobile phone data from GSM networks for traffic parameter and urban spatial pattern assessment: a review of applications and opportunities," *GeoJournal*, vol. 78, pp. 223-243, 2013.
 - [143] B. Hellinga, P. Izadpanah, H. Takada, and L. Fu, "Decomposing travel times measured by probe-based traffic monitoring systems to individual road segments," *Transportation Research Part C: Emerging Technologies*, vol. 16, pp. 768-782, 12// 2008.
 - [144] C. Liberto, R. Ragona, and G. Valenti, "Traffic prediction in metropolitan freeways," in *Traffic and Transportation Studies 2010*, ed, 2010, pp. 846-860.
 - [145] M. Rahmani, "Path Inference of Sparse GPS Probes for Urban Networks: Methods and Applications," KTH Royal Institute of Technology, 2012.
 - [146] T. Hunter, R. Herring, P. Abbeel, and A. Bayen, "Path and travel time inference from GPS probe vehicle data," *NIPS Analyzing Networks and Learning with Graphs*, 2009.
 - [147] F. Zheng and H. Van Zuylen, "Urban link travel time estimation based on sparse probe vehicle data," *Transportation Research Part C: Emerging Technologies*, vol. 31, pp. 145-157, 2013.
 - [148] F. Kunde, A. Hartenstein, and P. Sauer, "Spatio-Temporal Traffic Flow Forecasting on a City-Wide Sensor Network," in *Proceedings of GIS Ostrava*, 2017, pp. 253-265.
 - [149] R. Margiotta, "State of the practice for traffic data quality," presented at the Traffic Data Quality Workshop, Washington D. C., 2003.
 - [150] Land Transport Authority. (23 March 2018). *LTA Datamall API User Guide and Documentation* Available: https://www.mytransport.sg/content/dam/datamall/datasets/LTA_DataMall_API_User_Guide_v4.6.pdf
 - [151] J. Poushter, "Smartphone ownership and internet usage continues to climb in emerging economies," *Pew Research Center*, vol. 22, 2016.

- [152] K. Zheng, Y. Zheng, X. Xie, and X. Zhou, "Reducing uncertainty of low-sampling-rate trajectories," in *Data Engineering (ICDE), 2012 IEEE 28th International Conference on*, 2012, pp. 1144-1155.
- [153] J. Chen and M. Bierlaire, "Probabilistic multimodal map matching with rich smartphone data," *Journal of Intelligent Transportation Systems*, vol. 19, pp. 134-148, 2015.
- [154] M. T. Alrefaie, I. Carreras, F. Cartolano, R. Di Cello, and F. De Rango, "Map matching accuracy: Energy efficient location sampling using smartphones," in *Intelligent Transportation Systems-(ITSC), 2013 16th International IEEE Conference on*, 2013, pp. 2243-2248.

Appendix A

Dijkstra's Shortest Path Algorithm

Let $G = (V, E)$ be a directed and weighted graph with a set of nodes V and a set of edges $E \subset V \times V$. Let $l(u, v)$ be a non-negative length associated with each edge $(u, v) \in E$. Dijkstra's algorithm finds the shortest-path distances from a given source node s to all other nodes in the graph. The pseudo code in Algorithm A.1 provides a description of the algorithm. For each node i , Dijkstra's algorithm maintains and updates a tentative distance $dist[i]$, which is an upper bound on $d(s, i)$, the shortest-path distance from s to node i . At the start of the algorithm, $dist[i]$ is set to infinity for each node i . A node i is considered to be *settled* when the algorithm determines that $dist[i]$ cannot be improved further (i.e., $dist[i] = d(s, i)$). The algorithm maintains a priority queue Q , which is initialized to be empty. The source node s is added to Q after setting $dist[s]$ to be 0. At each iteration step, a node u with the lowest value of $dist[u]$ is removed from Q . For each neighbour node v of node u , $dist[v]$ is improved (i.e., reduced) if the path from s to v through u is shorter than the existing value of the tentative distance $dist[v]$. In such a case, node v is added to Q . It is worth noting that unlike in most versions of Dijkstra's algorithm, in the version used in this work, it is possible for a node to be added more than once to the priority queue. However, this does not affect the correctness of the algorithm.

Dijkstra's algorithm, in its standard form, terminates only when the shortest-path distances from the source node to all other nodes are found. However, the algorithm can be easily modified to terminate as soon as a set of

target nodes are settled. Also, the version given in Algorithm A.1 computes only the shortest-path distances and does not store the details of the shortest paths. However, if needed, the tracing of the shortest paths can be easily enabled by storing the predecessor node of each node, when the latter's tentative distance is updated.

Algorithm A.1: Dijkstra's algorithm

```

1 Dijkstra ( $V, E, s$ )
2    $Q = \emptyset$ 
3   for each  $i \in V$ :
4        $dist[i] = \infty$ 
5        $settled[i] = false$ 
6    $dist[s] = 0$ 
7   add  $s$  to  $Q$  with key equal to  $dist[s]$ 
8   while  $Q$  is not empty:
9       remove  $u$  with minimum key from  $Q$ 
10      if  $settled[u] == false$ :
11          for each  $(u, v) \in E$ :
12              if  $(dist[u] + l(u, v)) < dist[v]$ :
13                   $dist[v] = dist[u] + l(u, v)$ 
14                  add  $v$  to  $Q$  with key equal to  $dist[v]$ 
15       $settled[u] = true$ 

```