8-6-2015

Crowdsourcing Real-Time Traveler Information Systems

Asif Rehan
University of Connecticut - Storrs, asif.rehan@engineer.uconn.edu

Crowdsourcing Real-Time Traveler Information Systems

Asif Rehan

B.Sc., Bangladesh University of Engineering and Technology, 2013

A Thesis

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

At the

University of Connecticut

2015

APPROVAL PAGE

Master of Science Thesis

Crowdsourcing Real-Time Traveler Information Systems

Presented by

Asif Rehan, B.Sc.

Major Advisor _____

Karthik C. Konduri

Associate Advisor _____

Nicholas E. Lownes

Associate Advisor _____

John N. Ivan

University of Connecticut

2015

ACKNOWLEDGMENTS

This thesis would not be possible without the guidance, support, freedom and continuous encouragement from Dr. Karthik Konduri, my major supervisor. I sincerely acknowledge his crucial role at every step and express my sincere gratitude. I would like to thank Dr. John Ivan and Dr. Nicholas Lownes, who co-advised my thesis and supported with their deep insight, expertise and appreciation.

I would also take the opportunity to thank Dr. Krishna Pattipati with whom I built up the foundation that helped my research in multiple ways. I am equally grateful to Mr. Subrata Saha, whose invaluable feedbacks enlightened the path through the darkness.

I am thankful to Mr. Fahim Rahman and Dr. Saidul Islam, who offered a home in the absence of the home. I always remember the undying spirit of my parents, Mr. Rezaul and Mrs. Ferdousi Islam, and draw inspiration from my siblings, Amir Shahan and Fatiha Subah, who are always there for me.

I also acknowledge the financial support for the research project from New England University Transportation Center (NEUTC). I would like to thank all my friends and colleagues, for their help and support throughout the journey.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF EQUATIONS

ABSTRACT

In the last decade, the concept of collecting traffic data using location aware and data enabled smartphones in place of traditional sensor networks has received much attention. With a steady market growth for smartphones enabled with GPS chipsets, the potential of this technology is enormous. This combined with the pervasion of participatory paradigms such as crowdsourcing wherein individuals with portable sensors instead of physical networks serve as sensors providing information. Crowd sensed data overcome a number of issues with traditional physical sensor networks by providing wider coverage, real-time data, data redundancy and cost effectiveness to name a few. While there has been a lot of work on actual implementations of crowd sensed traffic monitoring programs, there is limited work on assessing the quality, and validity of crowd sensed data. A systematic analysis of quality and validity is needed before this paradigm can be more commonly adopted for traffic monitoring applications. To this end, research is underway to deploy a crowdsourced platform for monitoring and providing real-time transit information for shuttles that serve the University of Connecticut. The thesis develops a framework and an open-source prototype system that is able to produce real-time traveler information based on crowdsourced data. In order to build the prototype, first it implements a robust Hidden Markov Model based map-matching algorithm to position the crowdsourced data on the underlying road network and retrieve the likely path. The accuracy of the map-matching algorithm has been found satisfactory for the current usage even when the GPS points are sampled at low frequency. Next, to predict the travel condition across the network from the crowdsourced data, a travel time prediction algorithm, based on Regularized Least Square

Regression, has been implemented as well. This travel time prediction algorithm, together with the map-matching algorithm, has been applied in a simulated crowdsourcing environment. The travel time prediction results of the simulation show that the prototype system is quite capable of predicting travel time even when the crowdsourced real-time data is sparse. The simulation tests the performance of the travel time prediction algorithm in different scenarios. From the demonstrated predictive performance of the implemented prototype system, this approach to providing real-time traveler information is found promising. It is also possible to apply the prototype to all regions and all modes of transportation, exploiting its generalized approach of providing real-time traveler information from crowdsourced data.

CHAPTER 1 INTRODUCTION

In the recent past, there has been a shift from capacity oriented policies to demand management (TDM) oriented policies for addressing the congestion issues faced by transportation infrastructures in the US and elsewhere. As the name suggests, the primary inspiration behind TDM strategies is to manage the demand for travel either by reducing the demand or by distributing the demand in time and space as mentions FHWA *(1)*. The shift to TDM oriented policies has grown mainly due to issues associated with capacity oriented policies namely funding limitations to add new infrastructure, implications for air quality, impact on energy consumption, and sustainability considerations of urban regions. Further, the adoption of TDM practices for managing congestion has grown due to advances in hardware and software technologies, accelerated by innovative solutions and furthered by a better understanding of the factors underlying travel behaviors and traffic movement.

Travel Demand Management includes a whole host of strategies aimed at improving accessibility, providing information about alternatives, promoting alternative modes, improving transportation predictability, reducing unreliability and enhancing system performance *(1)*. Intelligent Transportation Systems (ITS) are a popular TDM strategy that utilizes advances in information and communication technologies (e.g., computers, telecommunications, location services, and data). Real-time traveler information services (RTIS) is a subset of ITS wherein travelers are provided with up-to-date information about transportation networks so that individuals can make efficient pre-trip and en-route transportation choices. Additionally, the information is used by transportation agencies to manage the transportation infrastructures in real-time to ensure quality service and maximum capacity utilization.

Traditional implementations of RTIS solutions depend on physical traffic sensor networks (including loop detectors, and traffic cameras) to collect information about transportation conditions in real-time and then disseminate such information to travelers and transportation agencies. However physical sensor networks suffer from limitations. They are unable to provide a wide and reliable coverage because they are often limited to major roads due to high installation and maintenance cost. In the last decade, an alternate approach to providing RTIS has been growing in popularity mostly in the private industry circles – data-enabled and location aware smartphone devices combined with crowdsourcing – a participatory paradigm of information sharing are being used to replace traditional traffic sensor networks. In this new approach to providing RTIS, smartphones of the end-users serve as a network of mobile sensors providing information about prevailing network conditions. The resulting RTIS solution has potentially wider coverage, almost real-time currency, and redundancy in data collection when compared with RTIS based on physical sensor networks. Additionally, since the end-users serve as sensors and share information, there is little investment involved in deploying the sensor network and minimal cost is incurred in maintaining and operating the information services.

Crowdsourcing has gained in popularity for providing real-time traveler information because of the growing penetration of data-enabled and location-aware handheld devices *(2)(3)*. More recently, the participatory paradigm has been implemented by private sector companies, such as Waze *(4)*, INRIX *(5)*, and NAVTEQ *(6)*, to mine travel behaviors and provide traveler information services. However, the emphasis by these providers is on traveler information services in major metropolitan areas; coverage is mostly limited to higher roadway functional classes (mostly interstates) because of the demand for such data by consumers such as media outlets, and navigation service providers. RTIS solutions based on the paradigm of

crowdsourcing hold promise for providing high-quality, real-time traveler information services in small and big regions not only for interstates, but also across all functional classes of roadways. Further, the paradigm offers the ability to provide real-time information across non-automobile modes including transit. The primary objective of the research presented in this thesis was to develop a RTIS solution that can be used to provide real-time travel information for transit buses for a small region.

1.1 Issues with Crowdsourcing

Today, crowdsourcing solutions mostly by private sector companies – the companies offer these solutions as commercial services. The solutions are implemented through patented methods and algorithms which are mostly inaccessible for further exploration and evaluation of the real-time traveler information that is generated and disseminated. While the potential of the crowdsourcing approach to RTIS is undisputed a number of concerns nevertheless remain and issues abound along three lines of inquiry about their applicability as a complete real-time traveler information services solution.

- First, quality of the data that is generated in crowdsourced platforms needs to be explored. Crowdsourced data suffers from data quality issues including sparsity, expanse, and validity. There is a need to qualify the data obtained from crowdsourced projects by comparing them against data obtained from traditional traveler information services based on fixed physical sensor networks.

- Second, data collected from crowdsourcing platforms only provides a sparse snapshot of the entire network and does not always have complete temporal coverage across all time periods and spatial coverage across the entire transportation network. At any given time,

only information about the links traversed by the users is available. This leads to a problem of inferring state of the entire network only from sparse real-time information. Therefore, there is need for methodologies and algorithms that can process sparse real-time information and possibly combine them with historical information to provide accurate real-time travel information for the entire network.

- Third, understanding what motivates users to contribute to crowdsourcing solutions and what factors will contribute to their continued engagement in such solutions is important to ensure longevity of the crowdsourcing solution.

In this thesis, the focus is on addressing the second line of inquiry described above. The thesis develops a framework and an open-source prototype system that is able to produce real-time traveler information based on crowdsourced data. The framework and prototype developed in the thesis can be generalized to all regions and all modes of transportation. However, in the thesis, the framework and prototype are demonstrated to provide real-time traveler information for the shuttle buses that service the University of Connecticut.

1.2 Research Objectives

The primary objective of this thesis is to develop a framework and build an open-source software prototype of a crowdsourced RTIS solution that is able to provide arrival time for transit vehicles in real-time.

1.3 Thesis Outline

The remaining thesis is organized as follows. **In the second chapter**, a general framework for crowdsourced RTIS system is described. **In the third chapter**, the focus shifts to the open-

source prototype of the crowdsourced system that was developed in this thesis. In particular, the map-matching (i.e. inferring path from the raw traces provided by users) component of the prototype is described in this chapter. **In the fourth chapter**, the travel time prediction algorithm that was implemented in the prototype is presented. **In the fifth and the last chapter**, some concluding thoughts as well as some future research directions are discussed.

CHAPTER 2 FRAMEWORK FOR CROWDSOURCED REAL-TIME TRAVELER INFORMATION SYSTEMS

2.1 Framework for Crowdsourced Real-Time Traveler Information Systems

A schematic of the framework for the Real-Time Traveler Information Systems based on the concept of crowdsourcing is shown in the diagram below. The different components of the framework are described in detail in the following sub-sections.

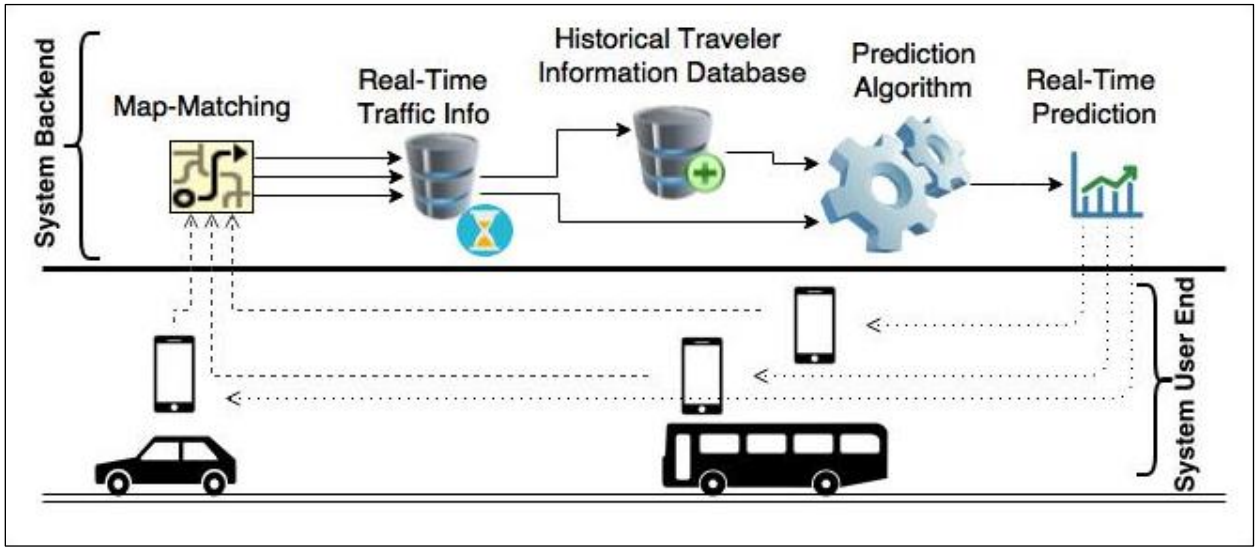

FIGURE 2.1 Conceptual Framework of the Proposed Crowdsourced Real-Time Traveler Information Systems

The system is part of a larger project titled as RETTINA: **Re**al-time **T**ransit **T**raveler **In**forma**a**tion, funded by New England University Transportation Center (NEUTC). The focus of this thesis is on the backend of the RETTINA system.

2.1.1 User End

The bottom portion of the diagram in FIGURE 2.2 below the thick black line shows the user end of the system. Users collect and contribute information typically using a data-enabled and

location-aware portable device. These days the portable device is the smartphone and users participate in crowdsourced efforts using an application that is installed on the smartphone. The smartphone application will allow users to interact in two ways. First, users will be able to access information about both expected traveler information (based on historical information typically used for planning a trip) and real-time traveler information (based on prevailing conditions that is used to make changes to the trip). Second, users will be able to share real-time traveler information both passively (about their current location as they pursue a trip) and actively (about prevailing network conditions such as disruptions, accidents, weather conditions, and ride experience). Information shared by the users is sent to the backend infrastructure shown above the thick black line in the diagram. The data shared by all the users is consumed in real-time by the backend infrastructure to provide real-time traveler information. In the bottom half of the figure, the data that is accessed by the users is shown with incoming arrows and data that is shared by the users is shown with outgoing arrows.

2.1.2 Backend

The top portion of FIGURE 2.2 represents the backend infrastructure. Backend infrastructure features both hardware and software to process, synthesize, and disseminate real-time traveler information based on the data that is shared by the users. The connections and workflow between different components of the backend infrastructure are shown in solid arrows. In the remaining thesis, the location data shared passively by users is of interest because of their direct relevance to providing traveler information. While other information shared actively about the surrounding environment may potentially be useful for predicting traveler information in real-time, it is not the focus in this thesis.

Map-Matching

Raw location data shared by users consists of a series of points – each point comprises three main information items namely latitude, longitude, and time. Series of points is also referred to as a trace providing the location of the user in space and time. The traces shared by users are typically anonymous. Further, they are devoid of any network information i.e. the points are not mapped to the links on which the user was traveling. Therefore, the first step in utilizing the user provided location information to provide traveler information is to map-match the data i.e. map the points to the corresponding links on the transportation network. Once map-matching is done then the data can then be used to infer traffic conditions on the transportation network. As can be seen, map-matching the raw location data is one of the fundamental steps and the quality of the map-matching algorithm has implications for the subsequent steps. The map-matching problem is described in greater detail in Chapter 3. Also, the specific solution to the map-matching problem that was implemented in the prototype is presented in Chapter 3.

Real-Time Traffic Information

With the map-matched location data, the real-time traveler information such as speed, travel time, and delay attributes can be obtained for those links that were traversed by the users.

Historical Traveler Information Database

The information that is collected in real-time is also typically archived into a database because such information is useful to predict conditions along roadway links for which data is not available. For example, when users have not traversed certain links in the network for given time period, then historical repositories can be mined to provide traveler information for those links.

Travel Time Prediction Algorithm

If the users provide information for all links in the transportation network then providing traveler information for a path on the network is trivial – just add the travel times on links along the path. However, in a RTIS solution based on crowdsourcing, location information shared by users does not cover the entire network and information for links is missing. Thus, in the absence of information for all links, a travel time prediction algorithm must be implemented that can provide information for all links on the network. Travel time prediction algorithm would typically combine real-time information for some links with historical information for remaining links to predict the real-time conditions for all links on the network. To predict the travel time for all the roads in the network using only the partial real-time information of the network remains a challenging task. The travel time prediction problem is presented in greater detail in Chapter 4. Also, the specific solution to the problem that was implemented in the prototype is presented in detail in Chapter 4.

2.2 Location Tracking Technologies in Smartphones

As described earlier, data enabled and location aware smartphone devices serve as the sensors for providing information about the prevailing network conditions. Location awareness allows smartphones to collect data and data connectivity allows users to transmit information in real-time using smartphones. Smartphone devices today feature a number of technologies for providing the location information. TABLE 2.1 provides a summary of different technologies including the approach to calculating the location, accuracy, and potential applications; more details regarding these technologies can be found in Zeimpekis et al *(7).* From among the different technologies, a short discussion of three technologies and their usefulness for a crowdsourced traveler information system are described in the following subsections.

TABLE 2.1 Technology Suitability for Location Tracking

Example Applications	Application Environment	Accuracy Requirement	Proposed Location Method	Position Calculation	Technology
Emergency Calls	Outdoor	Medium to High	TDOA	Terrestrial network or Device	Triangulation
Automotive Assistance	Outdoor	Medium	AOA/TOA	Terrestrial network or Device	Triangulation
Travel Services	Outdoor	Medium to High	Cell-ID	Terrestrial network	Cell proximity
m-yellow pages	Outdoor	Medium	Cell-ID	Terrestrial network	Cell proximity
Banners, Alerts, Marketing	Outdoor	Medium to High	TOA	Terrestrial network or Device	Triangulation
People tracking	Indoor/ Outdoor	High	GPS/Indoor GPS	Device from satellite data/Pseudo satellite	Triangulation
Indoor routing	indoor	High	Indoor GPS	Pseudo satellite	Triangulation
Vehicle tracking	Outdoor	Medium	GPS/ A-GPS/MT over S-UMTS	Device from satellite data	Triangulation
Product tracking	Indoor/ Outdoor	Medium to High	GPS/Indoor GPS	Device from satellite data/Pseudo satellite	Triangulation
Traffic management	Outdoor	Medium	GPS/A-GPS/MT over S-UMTS	Device from satellite data	Triangulation
Product replenishment	Indoor/	High	Indoor GPS	Pseudo satellite	Triangulation

2.2.1 Cell ID

The technology primarily influences the detail of the spatial information – this subsequently influences the ability and accuracy of traveler information on links in real-time. In this technology, the spatial information contained in each point of a trace is the location of the closest cell phone tower. It can be seen that there are a number of links in the vicinity of a cell phone tower so it is often hard to map the point to the link that was traversed by the user. The spatial information contained in very aggregate and is not very usable for providing real-time traveler information. For example, Watzdorf and Michahelles *(8)* say that this technology is not capable of being within 300 meters of the road actually traversed. While this low resolution location information is sufficient for many location based services, it does not offer enough data for providing traveler information along individual link sin a network. The accuracy of the locations that were provided was in the order of hundreds of meters. Similar observations were also reported in another study by Zeimpekis et al. *(7).* In their study, the authors found accuracy of this technology in the range of 200 meters. They also state that the accuracy could be improved in denser urban areas where there is a higher where cell networks are closely located. However, they are still not accurate enough to provide traveler information along individual links.

2.2.2 Wireless Local Area Network Technologies

Commonly known as Wi-Fi, this technology offers relatively better accuracy than Cell-ID. In this technology, the spatial information contained in each point of a trace is the location of the closest Wi-Fi hot spot with a known address. This technology also suffers from similar accuracy concerns as the Cell-ID technology. For this technology to be usable for providing traveler information, it requires dense coverage of Wi-Fi hotspots along roadways. Even with a dense

coverage, the accuracy of the technology is large and thus not usable for providing traveler information.

2.2.3 Geographic Positioning System (GPS)

GPS technology uses a system of satellites to estimate the location of a cell phone at any given time. Unlike the earlier two technologies where the spatial information contained in the trace is either the location of the cell tower or the location of the Wi-Fi hotspot, the spatial information contained in a GPS trace is based on the actual location of the smartphone. As a result, the accuracy of location information based on the GPS technology is very high and generally less than 10 meters Zeimpekis et al. *(7)*. GPS technology is not usable when satellites are not visible to the smartphone – this occurs when the smartphone is inside a covered area (e.g. buildings). Also, GPS technology requires significant warm up time called Time-To-First-Fix (TTFF). To overcome this limitation, a variant of the GPS called Assisted-GPS (A-GPS) is now featured in most smartphones. A-GPS combines cellular data technology and GPS technology to provide location information when there is a poor signal and also provides a better TTFF, according to Zandbergen and Barbeau *(9)*.

It can be seen from the above discussion that GPS technology and its variants are the most appropriate for a crowdsourceing based real-time traveler information. The observations of superior accuracy of the GPS technology compared to Cell-ID and Wi-Fi were further confirmed with an exploratory analysis of the different technologies using an Android application called GPS Logger *(10)*. Therefore, in the remaining thesis it is assumed that the location information shared by users is based on GPS technology.

2.3 Prototype of a Crowdsourced Real-Time Traveler Information

A prototype of a crowdsourced Real-Time Traveler Information Systems is being developed as a larger research effort sponsored by the New England University Transportation Center (NEUTC). The prototype will feature a smartphone application on the user end and the hardware and software infrastructure on the backend as described in the framework earlier. Further, the prototype will be demonstrated by providing real-time traveler information for shuttle buses that service the Storrs campus of the University of Connecticut (UConn). In this thesis, the focus is on the backend infrastructure of the prototype. In particular, the thesis presents methods and algorithms employed to implement the two key components of the Real-Time Traveler Information namely map-matching and travel time prediction.

In the demonstration, the shuttle bus service for the Storrs campus was chosen because they are already fitted with on-board GPS units. The data from the on-board GPS units served two purposes. First, it provided a baseline (known) against which the crowdsourced data could be compared and characterized. Second, it allowed the creation of synthetic datasets that mimic varying levels of user participation in the actual crowdsourced project.

The UConn shuttle service area map is shown below in FIGURE 2.2. The roads served by the UConn shuttle are marked with red lines while all other roads are marked with blue lines. UConn shuttle serves both the fairly built up area of the university core beside CT 195 and the rural campus area closer to the US 44 as shown on the map. Thus offering the ability to study the performance of this system under varying configurations of the roadway network and built environment.

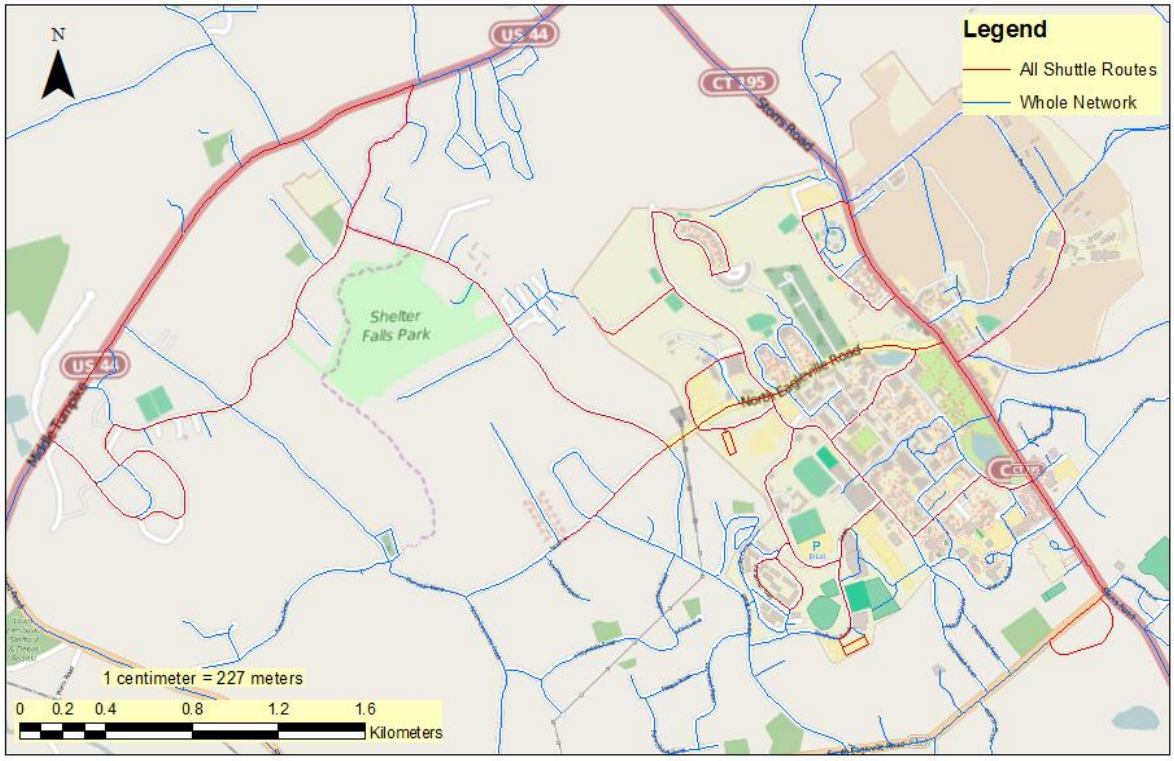

FIGURE 2.2 UConn Shuttle Service Area Map

2.4 Summary

This chapter outlined the conceptual framework of the crowdsourced based solution for real-time traveler information. The following chapter presents the particular map-matching approach that was implemented in the prototype. Additionally, a detailed analysis if the performance of the map-matching algorithm is presented using data from the on-board shuttle GPS.

CHAPTER 3 MAP MATCHING ALGORITHM

3.1 Introduction

As noted in the previous chapter, it is assumed that the location information shared by users will be based on GPS technology. The first step in providing traveler information in the crowdsourced traveler information system is map-matching. In map-matching, the path (i.e. series of links) traversed by the user on the transportation network is inferred from the raw GPS data points. A wide range of algorithms are available for map-matching. The next section explains the motivation behind the map-matching problem. The approaches for developing map-matching algorithm are outline in the following literature review section. The implementation of the selected algorithm and the results appear in the last part of this chapter, right before the concluding remarks.

3.2 Motivation

There are a number of considerations that go into selecting an appropriate map-matching algorithm. Three key considerations that dictate the choice of map-matching algorithm include the following:

3.2.1 GPS Errors

Even though GPS is more accurate when compared with other location technologies featured on smartphones, they are still not exact and suffer from errors. Primary causes of error in GPS data include signal noise and clock issues. Also, GPS data from smartphones suffer from inaccuracies based on the quality of the chipset embedded in the smartphone. These errors in turn affect the ability to position the points directly on the links traversed by the user.

3.2.2 Density of Roadway Network

The implications of GPS errors for map-matching are magnified further when the density of roadway networks is high. As described in the previous chapter, the prototype will provide real-time traveler information for the shuttle services serving the UConn campus area. In the vicinity of the UConn, the density of the streets is generally high thus magnifying the difficulty of map-matching in the presence of any GPS errors.

3.2.3 Frequency of GPS Data

The third consideration in the choice of map-matching algorithm is the frequency at which location data is collected. If the frequency of the points is high then the problem of map-matching is more straightforward then when the frequency is low. At low frequency it is difficult to infer the path between two subsequent points especially when many paths connecting the links adjacent to the two consecutive GPS points. This is even larger consideration in a crowdsourcing solution wherein smartphones are utilized because utilizing GPS on a smartphone comes at the expense of very high battery power consumption. For example, it was observed by Thiagarajan *(11)* that, sampling GPS points at 2 min interval in a smartphones reduced its battery life by nearly one third. To avoid draining smartphones battery GPS points are sampled at lower frequencies. This in turn makes the problem of map-matching difficult because it is hard to infer the path when no information exists.

3.3 Objectives

The objective of this chapter is to implement a robust map-matching algorithm to retrieve the path from the crowdsourced data, as shown in the conceptual framework in FIGURE 2.1, without assuming any specific mode or any fixed routes, such as transit routes. This objective

leads to a generalizable approach for the crowdsourced RTIS solution. Moreover, it intends to apply the algorithm in a case study and validate the applicability of the algorithm in the current scenario.

3.4 Literature Review

Until recent past, GPS data was only collected using dedicated GPS units. This approach didn't suffer from some of the considerations noted above. For example, frequency of sampling the points was never a consideration because the dedicated devices were fitted with continuous power or large battery packs. Further, the dedicated units featured quality hardware thus avoiding potential GPS errors. Therefore, map-matching GPS data from dedicated units was possible using simplistic approaches. However, the above considerations became more important with the introduction of GPS chips in smartphones requiring sophisticated algorithms for map-matching. The development of algorithms for the map-matching problem is an active area of research. In this section, a review of the literature on map-matching algorithms is presented.

3.4.1 Terminologies: Map Matching and Path Inference

In the literature, map matching has traditionally been referred to the problem of matching raw GPS points to corresponding points on the network links. On the other hand, the problem of predicting the likely path between two consecutive points on the network corresponding to two GPS points is referred to as path inference. A detailed review of the differences between the two can be found in *(12).* In this particular research, map-matching not only includes matching raw GPS points onto the network but also inferring path between the points.

3.4.2 Existing Approaches

This subsection reviews existing literature on map-matching. In an effort to qualify the effectiveness of the algorithms, accuracy values are presented. However, it must be noted that the formulation of the accuracy metric varies across the different articles and are not directly comparable. Therefore the values should be just used only as guidance and not to compare the algorithms. The authors in *(13)* provide a detailed review of earlier papers on map-matching. The algorithms for map-matching can broadly be categorized into two groups namely 1) approaches based on geometry and topology matching and 2) approaches based on statistical methods. Below methodologies within these two groups of map-matching algorithms are reviewed.

Geometry and Topology Based Approaches

White et al. *(14)* proposes four algorithms. The simplest one projects GPS points onto nearest link. The second one considers the heading of the GPS points. The relatively sophisticated one uses the topological and network connectivity information to determine which links, associated with the next GPS point, are accessible from the current one. It helps to preclude unlikely road links using the empirical restriction. The last algorithm uses curve-to-curve matching in addition to the link-identification technique used in the third algorithm. The identified piece-wise linear curves emanating from a node on the network are then matched with the piece-wise linear curve formed by the GPS point sequence. These algorithms are useful for high frequency GPS points but are quite empirical in nature.

Statistical Approaches

Particle Filter The use of particle filter or sequential Monte Carlo method is used for map-matching by Davidson et al. *(15)*. The algorithm can use GPS or Dead Reckoning (DR) sensor.

The heading information taken into consideration makes it vulnerable to low-sampling and noisy cellphone GPS data.

Fuzzy Logic is used by Quddus et al. *(16)* wherein a Fuzzy Inference System was implemented to obtain the path. The algorithm takes both dead reckoning sensors information and high frequency GPS data as inputs. It produces 99.2% accuracy at 1 second interval and 5.5m horizontal errors. Velaga et al. *(17)* also use Fuzzy Logic for map-matching in a crowdsourced project focusing on transit traveler information systems. The users on-board contribute to the GPS location with their cellphones. The collective data are cross-validated in the map-matching process. The outliers from one user can be balanced by the weightage on the GPS data from the other users. Input data into the process includes speed, heading and estimated positioning accuracy. The estimated accuracy is used to create an error region for each GPS observation from a user. Roadway links within this error region are considered part of the actual path. These data are used to assign weights to each potential link. Most importantly, since, the scheduled route is known, the weight to the links that have a similar heading as reported by GPS data, gets the higher weightage. Multiple potential links may need to be considered for observations for each GPS observation from a given user. The Fuzzy Inference System is then used to choose a series of link that represents the likely path taken by the user.

Kalman Filtering is used Najjar and Bonnifait *(18).* In this paper Belief Theory and Extended Kalman Filter are used to fuse data from the GPS sensor and the Anti-lock Braking System (ABS) to compare the GPS observation to the estimated position. The algorithm is designed towards situations when GPS is unavailable for a substantial period. In this situation the location can be approximated from the ABS sensor data.

Multiple Hypothesis Technique (MHT) was applied by Chen et al. *(19).* They propose a Multi-criteria Dynamic Programming technique to extend the MHT approach used in Marchal et al. *(20)*. The algorithm is also compared against other algorithms to assess performance of their proposed approach. The algorithm was tested on different GPS datasets with frequencies ranging from 10 to 120 seconds. The results show that the algorithm performed slightly better than the Hidden Markov Model based algorithm proposed by Newson and Krumm *(21)*. But when the at a GPS data frequency of 120 seconds, the performance of the algorithm was comparable to that of Newson and Krumm *(21).*

Bayesian Belief Networks was used in Feng and Timmermans *(22).* In this study, data was collected using a portable GPS logger at a 3 second frequency. The accuracy of this algorithm was close to 87%. This approach also depends on difference between heading and the road azimuth which is likely to suffer from inaccuracy when dealing with cellphone GPS data.

Frechet Distance approach was used by Brakatsoulas et al. *(23).* The study presents one greedy curve-to-curve technique for quick map-matching results and the authors also propose two other map-matching based on Frechet Distance techniques for slower but more accurate results. Their global algorithm uses the Frechet Distance and Weak Frechet Distance to find similarity between the curves in the road network and the observed trajectory. Given the runtimes associated with the two later techniques, they are applicable for offline usage.

Hidden Markov Model was originally adapted for Map-Matching by Hummel*(24).* The transition probability accounts for the connected roads. Taking driving direction as input weakens its applicability for cellphone GPS. The transition probability assigns similar probability to potential paths, i.e. no state transition is preferred. This is overcome by the proposed function for state transition in Newson and Krumm *(21)*. In this paper emission

probability takes the form of the Gaussian Normal distribution. Transition probability depends on the difference between the GCD and shortest path through an exponential function. Intuition behind this formulation of the function is that individuals choose the more direct path between two points. The algorithm gives best result when the frequency of GPS data is below 30 seconds (almost 95% accuracy) and gradually drops with higher noise in GPS data and lower frequency.

Concurrently, Lou et al. *(25)* offers a similar approach as in Newson and Krumm *(21)*. Here emission and transition probability are combined into the spatial function and observed speed of the vehicle is matched with the road speed profile through temporal functions. Thus this algorithm captures geometric, topological and speed information. The combination of the two functions is termed as ST-matching algorithm and is used in a manner that is similar to the Viterbi algorithm used in HMM to obtain the sequence of states that produces the maximum probability. The ST-matching algorithm considers the maximization of the probability of a path as a whole and is thus termed as global algorithm. On the other hand, the algorithms that decide the best path at each GPS point are termed incremental algorithms. The ST-matching algorithm is developed for frequency of 2 minutes or higher. Reported accuracy of matched nodes drops from 87% at 30 seconds GPS frequency to 68% at 5 minutes. But compared to incremental and Average Frechet Distance based algorithms, ST-matching results are more robust.

Yuan et al. *(26)* report that for multi-lane road and long trajectories, ST-matching is not reliable. Thus, in their paper they improved the ST-matching algorithm by introducing weights in their Interactive-Voting based Map Matching (IVMM) algorithm, both for influences between neighboring GPS points and points that are farther apart. This weighting scheme, termed as *interactive-voting*, is designed to handle outliers in low frequency data (2 minutes or higher interval). This algorithm takes input from the road network nodes, road links and travel speed of

the road links. It also requires the whole path to calculate the mutual influence of a point based on its distance from all other points. For the observation probability estimation, empirical mean and variance values for GPS accuracy are used. Parameter estimation remains a trial-and-error process. This global algorithm performs quite well even when the GPS frequency is as low as 10 minutes (65%). For frequency of 2-6 minutes, it is able to identify almost 83%.

Another improvement was proposed using the Viterbi decoder for online map-matching by Goh et al. *(27)* based on the HMM framework proposed in Newson and Krumm *(21)*. This approach uses Dead Reckoning (DR) sensor data for calculating the transition probability. This online map-matching algorithm uses an online Viterbi decoder with a Variable Sliding Window (VSW) to look back in the previous positions to estimates the path up to the current observation. But this backtracking of the parent nodes may not always yield satisfactory results within a reasonable time. It can produce suboptimal results as the length of the time-window needs to be constrained for practical reasons. The accuracy for urban roads is lower than for rural ones. Though, accuracy at very high frequency is nearly perfect, at lower frequency level, the accuracy drops in the range from 75% (5 minutes) to 90% (2 min).

Osogami and Raymond *(28)* extend the HMM framework proposed by Newson and Krumm *(21)* by introducing turn penalty. The model learns the weight of turns at intersections from the training data set of trajectories for one or more drivers on one or more routes. If more hard-turns along a path exist, the path is assumed to be less likely to be taken by the driver. The cost of turns along the path is incorporated in the cost function defined by Newson and Krumm *(21)*. Using the Maximum Entropy classifier in Inverse Learning framework, this algorithm learns the weightage for the given training dataset and uses it to estimate the path. Introduction of such

artificial intelligence increases the accuracy of HMM based algorithm proposed by Newson and Krumm *(21)* by over 40%.

Bierlaire et al. *(29)* is another example of HMM-inspired algorithm for map-matching GPS data from smartphones. This model follows the HMM framework and finds emission probability with Gaussian distribution of GPS points. The transition probability is based on the network performance model.. This network performance model requires estimation of parameters beforehand. Based on the given network performance model, it assigns a likelihood value to each probable path that connects the points. The path with the highest likelihood, given the observed GPS data sequence, is the most likely path.

While earlier papers used Gaussian probability distribution function to assign emission probability to the candidate road segments, Oran and Jaillet *(30)* argue that proximity-weight or emission probability should not be based on projection distance between GPS point and the candidate road segments. Emission probability or proximity weight is redefined as to assign probability to the road links proportional to their length falling within the circular error region of the GPS points. Thus it employs cumulative Gaussian distribution function for emission probability calculation. The paper shows when accuracy is too low, the radius of error region can be assumed infinity. Thus the nearest distance assigns equal weight for the roads equidistant from the GPS point. But when the length of the road within the error regions is significantly different, the longer one obtains maximum weight. In other words, short road segments are less probable of being the true location for that given GPS observation point. As a result their formulation always penalizes the road segments that have shorter length within the error region.

Eventually Oran and Jaillet *(31)* used the HMM-model from Newson and Krumm *(21)* as is except that they used the cumulative weightage function defined as emission probability. Next, the method uses Hidden Markov Model to calculate the probability of transmission from one candidate point for the first GPS point to the next point. Finally the proposed method uses Viterbi algorithm to calculate the likelihood of the most likely state transitions to construct the path. Similar to Newson and Krumm *(21)*, the HMM model is dependent on a function of the difference of Greater Circular Distance and the Shortest Distance between the two consecutive GPS points. It was also mentioned that for some Map-Matching Algorithms, the error region definition for GPS point is either obtained by experimentation while some others simply assume the parameters a priori based on process knowledge. Such as in this paper, the GPS accuracy parameters are taken from established guidelines *(32)*. Compared to Newson and Krumm *(21)*, this algorithm performs slightly better for higher noise and longer interval data. For example, at 5 minutes interval and standard deviation of 17.25 meter, the accuracy improves by about 4.4% compared to Newson and Krumm. But for high frequency data (1 second to 10 seconds), this algorithm offers slightly lower accuracy.

Repeated Regular Route is a more recent approach. Javanmard *(33)* propose a multiple sparse tracking data on the same route for map-matching. It is useful for trips that take place along the same route but with only sparse GPS observations. The authors propose two algorithms namely iterative projection scheme and graph Laplacian scheme for multi-track map-matching problem. A boosting method is also proposed for augmenting the path estimation process. In general this method is suitable to track commuter trips and transit routes.

Conditional Random Field is another technique proposed recently for map-matching by Hunter et al. *(34)*. The algorithm uses GPS data from a smartphone application that is pre-

programmed to send the location data to the server only when it crosses predefined Virtual Trip Lines (VTL). This approach to data collection was proposed in response to growing privacy issues with individual vehicle tracking. Path is estimated in the CRF approach using forward algorithm and backward algorithm. This approach corrects the Selection Bias in HMM, a limitation arising from the scaling process in calculating the state-transition probabilities in HMM.

3.5 Map-Matching Algorithm: Overview

3.5.1 Rationale for Selected Approach

After considering all available methodologies, Hidden Markov Model based map-matching algorithm was chosen because of its simplicity and robustness. The original HMM model formulation, along with modifications that were made to the original HMM methodology, is described in the following subsections.

3.5.2 Hidden Markov Model

Hidden Markov Model (HMM) is a general approach that is commonly used to draw inferences from sequential data. Location data obtained shared by users in crowdsourced traveler information systems also fits within this description – series of GPS points represents a sequential dataset providing information about the path traversed by the user. The HMM approach has been used to answer three types of problems in the literature as described below from Rabiner *(35)*:

- **First,** the Evaluation Problem: Finding the probability of observing a sequence, given the initial probabilities, state probabilities and state-transition probabilities.

- **Second,** the Decoding Problem: Finding the optimal sequence of states given the sequence of observations given the initial probabilities, state probabilities and state-transition probabilities.

- **Third,** the Learning Problem: Finding the initial probabilities, state probabilities and state-transition probabilities to maximize the observation probability given the sequence of the observations.

Map-Matching problem is similar to the decoding problem described above. Just like the decoding problem, map-matching involves solving for series of links (i.e. the optimal state sequence) for the given sequence of GPS points.

Hidden Markov Model for Map Matching

HMM approach has five elements which are also adapted for the map-matching problem implementation. These elements are illustrated in FIGURE 3.1and FIGURE 3.2. HMM attempts to temporally connect hidden or unknown states based on given observations. The GPS points that are shared by users in a crowdsourced application can be likened to observations. Their actual locations are along the links on a transportation network which are generally in close proximity of the GPS point. The actual location on the roads for the observed GPS point are unknown, thus, they can be likened to hidden states or unknown states in the HMM.

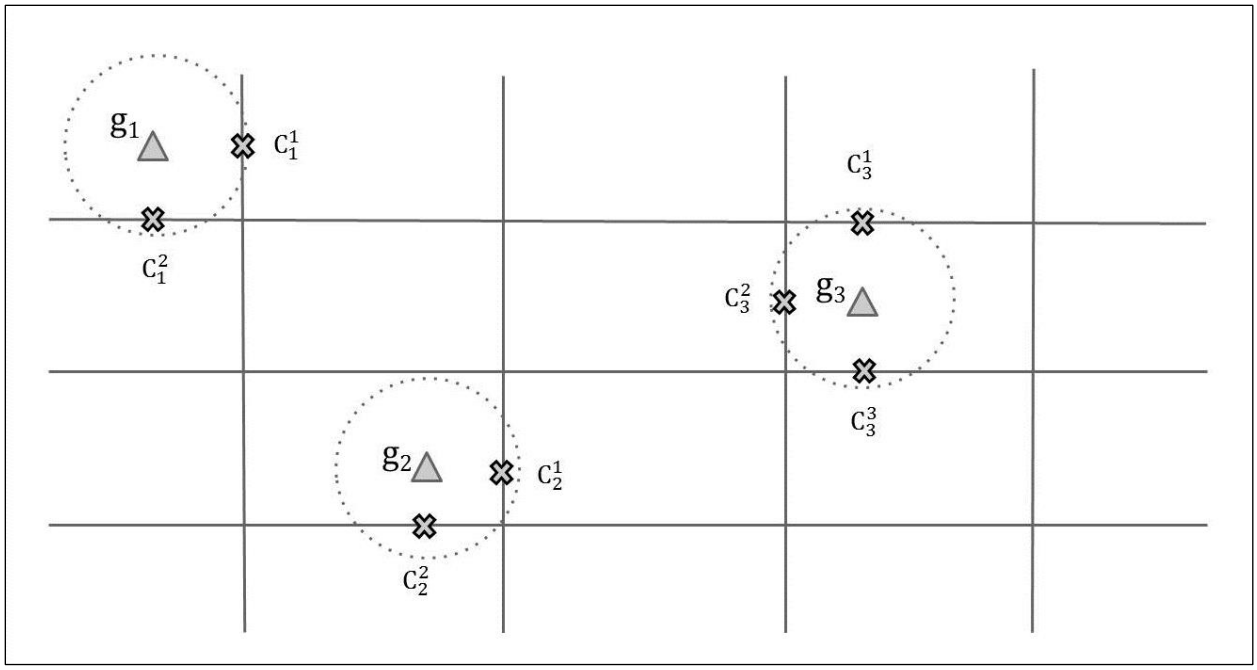

FIGURE 3.1 GPS Points and Map-Matching Context

In FIGURE 3.1, observed GPS points sequences is $g_{1:T} = \{g_t : 1 \leq t \leq T\}$. The most probable path has to be reconstructed from these temporally sequenced GPS points. Each GPS point is observed to have a Gaussian noise. So the dotted isocentric circles centering at each GPS point indicates a probable area within which the true location generating the GPS point exists. In usual case, this area and the radius can be found in *(32)*. A discussion on reasonable values for such 'Error Region' is found later.

For each of the observed GPS points, potential true locations or hidden states are marked with crosses on the figure. The potential true locations are obtained by drawing a perpendicular projection of the GPS point, g_t, on to the adjacent road links within the *error region*. These potential true locations are denoted by $c_t = \{c_t^j : 1 \leq j \leq J(g_t)\}$ which are also commonly known as 'candidate points' for that observed GPS point. Each GPS point g_t, has candidate point

c_t, and each candidate point c_t is denoted with j as a superscript. The number of candidate points can vary for each GPS point. So the number of candidate points for the GPS point at time t is noted as $J(g_t)$.

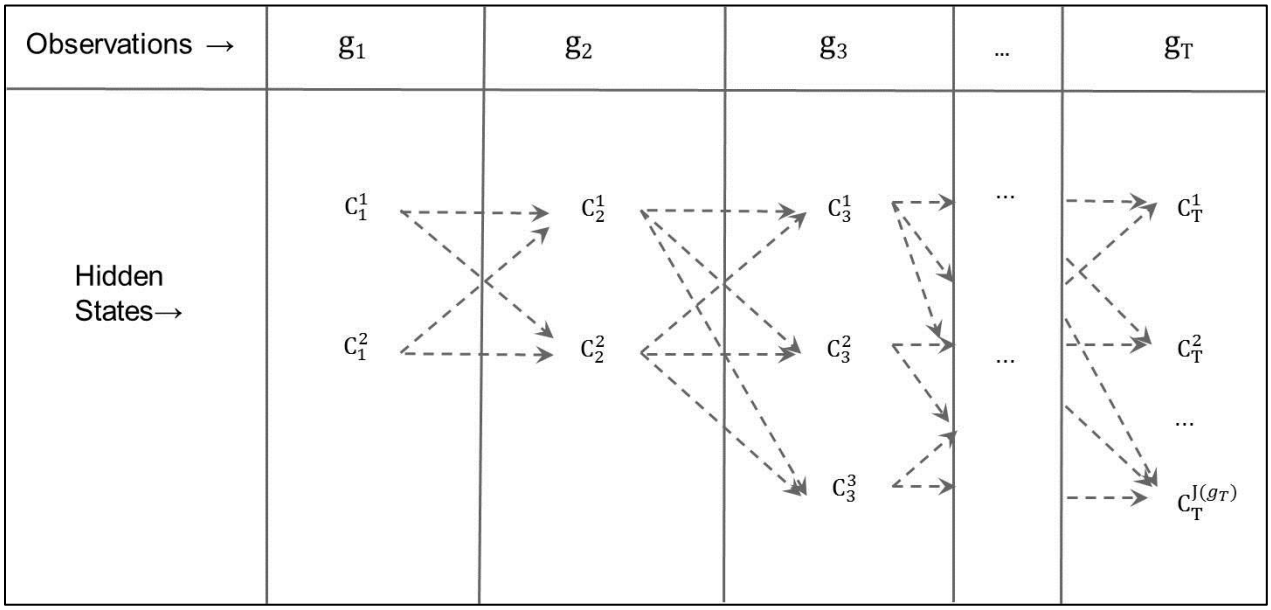

FIGURE 3.2 Trellis Diagram for Map-Matching Problem

The likely actual path from the given sequence of GPS points is obtained by connecting a *candidate point for each of the GPS points*. In order to obtain the likely actual path, the candidate points for consecutive points are all connected by potential paths – these potential paths between candidate points are referred to as transitions. Each possible transition from a state at time, *t*, to another at time *t+1* is also assigned a *transition probability*. In the FIGURE 3.2 above, the transition between candidate points as shown in FIGURE 3.1 are outlined as a Trellis Diagram. The transitions are shown in dashed-arrows. The search for optimal path that represents the likely actual path traversed by the user is actually obtained by applying the Viterbi Algorithm as in Rabiner *(35)* and Murphy *(36)*.

The likely actual path is denoted by the set $c^*_{1:T}$ below.

$$c^*_{1:T} = \{c^*_1, c^*_2, \ldots, c^*_T | \, g_{1:T}\} \qquad \text{EQUATION 3.1}$$

3.5.3 Definitions of HMM Elements

As noted earlier the map-matching algorithm implemented in the prototype is based on the work by Newson and Krumm *(21)*. Therefore, expressions in the formulation presented below follow notations in the original paper with small changes where appropriate.

Initial Probability

At the beginning of the map-matching process, the actual start location could be any of the candidate points for the first GPS point. As a result, initial probabilities are assumed 1 for each of the candidate points indexed by j for the observed GPS point g_t at time $t = 1$ which has $J(g_1)$ candidate points.

$$\pi_j = 1, \qquad 1 \leq j \leq J(g_1) \qquad \text{EQUATION 3.2}$$

Emission Probability on Candidate Point

The emission probability in a HMM refers to the probability that the candidate point is the unknown true location for the observed GPS point. Since the Gaussian error model for GPS noise is assumed based on the results by van Diggelen *(37)*, the emission probability is thus given by this expression

$$p\,(c^j_t | \, g_t) = \frac{1}{\sqrt{2\pi}\sigma} e^{-0.5\left(\frac{dist(c^j_t, g_t)}{\sigma}\right)^2} \qquad \text{EQUATION 3.3}$$

In Newson and Krumm *(21)* $dist(c_t^j, g_t)$ is calculated as the Great Circular Distance between the candidate point c_t^j from the observed GPS point at g_t. However, in this implementation, simple Euclidean distance was employed since the network information was already projected into UTM coordinate system. In the equation above σ indicates the standard deviation associated with the GPS noise – the value assumed for this parameter is discussed later.

Transition Probability between Candidate Points

As shown in the Trellis diagram in FIGURE 3.2, the state transition probability indicates the probability that a candidate point at time *t* is connected to candidate point at time *t+1* given observations of corresponding GPS points. This transition in real-world means the likelihood that an individual would have chosen to go to a candidate point at time t+1 given the person currently is at a candidate point at time t. Intuitively the transition probability depends on the characteristics of the path connecting the candidate points. Newson and Krumm *(21)* assign higher importance to a straight line path connecting two candidate points over a circuitous path. This is intuitively appealing and likely reflects reality. More precisely, the unscaled transition probability is defined as follows:

$$\omega(c_t^j, c_{t+1}^{j'} | \, g_t \, , g_{t+1}) \; = \frac{1}{\beta} e^{-\left(\frac{\delta(c_t^j, c_{t+1}^{j'}, g_t \, , g_{t+1})}{\beta}\right)} \qquad \text{EQUATION 3.4}$$

Where,

$$\delta(c_t^j, c_{t+1}^{j'}, g_t \, , g_{t+1}) = \; | \, GCD(g_t \, , g_{t+1}) \; - \; SPD(c_t^j, c_{t+1}^{j'}) \, | \qquad \text{EQUATION 3.5}$$

For any candidate point c_t^j, the total probability of all possible transitions to the candidate points $c_{t+1}^{j'}$associated with downstream GPS point should be mutually exclusive and thus, sum up to one. So the instead of raw transition probabilities, scaled transition probabilities are used as shown in the expression below:

$$p\left(c_t^j, c_{t+1}^{j'} \mid g_t, g_{t+1}\right) = \frac{\omega(c_t^j, c_{t+1}^{j'} \mid g_t, g_{t+1})}{Z_j} \qquad \text{EQUATION 3.6}$$

$$Z_j = \sum_{j'=1}^{J(g_t)} \omega(c_t^j, c_{t+1}^{j'} \mid g_t, g_{t+1}) \qquad \text{EQUATION 3.7}$$

In the equation above, $GCD(g_t, g_{t+1})$ indicates the Great Circle Distance between the two consecutive GPS points and the $SPD(c_t^j, c_{t+1}^{j'})$ is the shortest path distance between one candidate point c_t^jat time *t*, and another$c_{t+1}^{j'}$at time *t+1*. The δ function indicates that if two candidate points have a higher difference between the corresponding GCD and SPD, then the shortest path is more circuitous so the transition should be assigned a smaller weight. The β value is simply a tuning parameter, which is explained further in a later section.

3.5.4 Solution for Map-Matching

The solution to HMM based map-matching problem is obtained by applying the Viterbi algorithm. The optimal sequence of candidate point is found by a particular sequence of candidate points as in EQUATION 3.1 that maximizes the probability of $p(c_{1:T}|g_{1:T})$, i.e.,

$$c^*_{1:T} = \ argmax_{c_{1;T}}\, p(c_{1;T}|g_{1:T}) \qquad \text{EQUATION 3.8}$$

Where, the $p(c_{1:T}|g_{1:T})$ is the joint probability of the optimal sequence of the candidate points given the observation sequence of the GPS points.

$$p\ (c^*_{1:T}|g_{1:T}) = \ \pi^*_1 p(c^*_1|g_1) \sum_{t=2}^{T} p(c^*_{t-1}, c^*_t|\ g_{t-1}\ , g_t)\ p(c^*_t|\ g_t) \qquad \text{EQUATION 3.9}$$

Since, typical HMM follows the Markov assumption that the current state depends only on the previous state, this expression can take a form of recursive function which is evaluated by dynamic programming technique at each time step based on the values from the previous step and maximizing the joint probability. Additional detail regarding the algorithms can be obtained from Rabiner *(35)* and Murphy *(36)*.

3.5.5 Viterbi Algorithm for Map-Matching Context

Step-by-step detail of the Viterbi algorithm for the HMM-based map-matching is presented below. Here, the *path* can either be simply a series of candidate points or a series of links joining the candidate points ending at the *j*-th candidate point at time *t*. For crowdsourced traveler information application, the series of links defining the path between two points is more relevant. The path connecting the two points is obtained by taking the shortest path between the candidate points and is denoted as SP in Step#2 below. In simplest form, if only the optimal sequence of candidate points is desired instead of real-path, $path_1^j$ can be set to c_1^j and it can be

appended at each step by setting $SP(c_{t-1}^{i}, c_t^{j}) = c_t^{j}$. Otherwise, since in Map-Matching, transition probability function requires the calculation of shortest path distance, SP is usually a by-product from the computation.

Step#1: Initiate. t=1

For $1 \leq j \leq J(g_1)$

$$\psi_1^j = \pi_1^j p\,(c_t^j \mid g_1)$$ EQUATION 3.10

Set $path_1^j$ as empty

Step#2: Recurse for $2 \leq t \leq T$

For $1 \leq j \leq J(g_t)$

$$\psi_t^j = max_i\big(\psi_{t-1}^i\, p(c_{t-1}^i, c_t^j \mid g_{t-1}, g_t)\, p(c_t^j \mid g_t)\big), \quad 1 \leq i \leq J(g_{t-1})$$ EQUATION 3.11

$$i = argmax_i\big(\psi_{t-1}^i\, p(c_{t-1}^i, c_t^j \mid g_{t-1}, g_t)\big), 1 \leq i \leq J(g_{t-1})$$ EQUATION 3.12

$$path_t^j = path_{t-1}^i + SP(c_{t-1}^i, c_t^j)$$ EQUATION 3.13

Step#3: Terminate at $t = T$

$$j^* = argmax_j\, \psi_T^j, 1 \leq j \leq J(g_T)$$ EQUATION 3.14

$$path^* = path_T^{j^*}$$ EQUATION 3.15

So unlike regular Viterbi algorithm, this algorithm below does not recurse backwards for backtracking to construct path upon termination, which would require massive memory to store all possible shortest paths. Instead, at each step the optimal shortest path is appended to the previous for efficiency.

3.5.6 Practical Issues with Viterbi Algorithm

Since the algorithm multiplies fractional probability values, the ψ_t^j value quickly drops to a very small value within a few recursions across time interval t in the algorithm above. This is problematic because when implementing the algorithm in a software, this will lead to arithmetic underflow error i.e. the value of the variable is too small to be stored in a computer's memory. In an effort to avoid this issue, in the algorithm's implementation, instead of handling ψ_t^j that requires multiplication of probabilities, $\ln \psi_t^j$ is handled which requires summation of probabilities

3.6 Implementation in Python

This Map-Matching Algorithm was implemented in Python programming language. The road network was built from the roadway network shapefile obtained from UConn's MAGIC Library *(38)*. After trimming the shapefile to only include the UConn Storrs campus, pre-processing and breaking multi-line features into line-features, a standard multigraph object was created using NetworkX (a Python graphing library) *(39)*. In addition to underflow, inaccessible shortest path or invalid GPS points may also cause breakage in HMM. To avoid the breakage, the implementation ignores GPS points with no valid candidate point and moves onto the next. It also carefully stops the tree of path growing into an inaccessible or invalid path.

3.6.1 Parameter Values

For the emission probabilities, the circular error region is assumed to have a radius of 30 meters *(31)*. As per the guideline in *(32)*, this is larger than the 3 Deviation Root Mean Square (3DRMS) zone (27m) and captures the true location with more than 99.9% probability. Even though the map-matching algorithm implemented is based on Newson and Krumm *(21)*, unlike the original article where the standard deviation value is estimated from Mean Absolute Deviation, the value of the parameter is assumed following Oran and Jaillet *(31)*. Similarly, with the relationship 4.24σ = 3DRMS ≈ 30m, σ value of 7m is used. So the candidate points are found by perpendicular projections of the observed GPS points on the roads that are within the 30m radius of the GPS point. In the implemented Viterbi algorithm, if a GPS point cannot find any candidate point within that limit, then the GPS point observation is assumed invalid. In the transition probability expression, the tuning parameter β is assumed to have a unit value for simplicity.

3.6.2 Shortest path

The implementation actually finds the shortest path between projections of the GPS points on to the network i.e. between candidate points. It combines the Dijkstra shortest path between pairs of intersections at the two ends of the roads the points are projected on to and then adds the fractional distance from that pair of intersections and actual projection location.

3.7 Case Study

This section presents a case study wherein the map-matching algorithm was applied to assess the applicability and performance of algorithm using data from the UConn shuttle buses.

3.7.1 Dataset

In this case study, GPS dataset was not collected directly from a smartphone but it was collected from the GPS units on-board the UConn Shuttle buses. It must be noted that in a real application users will be asked to share information using Assisted-GPS (A-GPS) whose accuracy is comparable to the on-board GPS units. Thus the insights from the validation analysis should be transferable to the actual crowdsourced traveler information systems prototype wherein data from actual smartphones will be used.

TABLE 3.1 Count of GPS Traces and Corresponding GPS Sampling Frequencies

	GPS Freq	**Route**						
		bl	**gr**	**or**	**pl**	**rd**	**yl**	**Total**
Raw GPS Traces	5	0	0	0	0	6	0	6
	8	9	8	4	9	3	8	41
	10	0	1	5	0	0	1	7
	Total	**9**	**9**	**9**	**9**	**9**	**9**	**54**
Resampled GPS Traces	10	9	9	9	9	9	9	54
	15	9	9	9	9	9	9	54
	30	9	9	9	9	9	9	54
	45	9	9	9	9	9	9	54
	60	9	9	9	9	9	9	54
	Total	**45**	**45**	**45**	**45**	**45**	**45**	**270**

Each GPS trace covered a complete round-trip for a different shuttle bus route serving the UConn campus area. For each of the shuttle bus routes, separate GPS traces were collected from 3 weekdays namely Tuesday, Wednesday and Thursday. Also, different traces were collected from three time periods including morning (7-11 AM), afternoon (12-4 PM) and evening (4-8

PM). This process resulted in a total of 54 raw GPS traces (=6 routes X 3 time periods X 3 days). The detailed count of traces is shown in TABLE 3.1.

The sampling frequencies in these GPS traces are variable and are generally at 5, 8 or 10 second frequency. These GPS traces were resampled at 10, 15, 30, 45 and 60 second interval to create additional GPS traces that can be used to assess the performance of the algorithm under varying levels of frequencies. These GPS traces created by resampling the original GPS traces will be referred to as 'synthetic datasets'. The original GPS traces will be referred to as raw GPS traces. The raw GPS traces were not directly comparable to each other due to the inconsistent frequencies. As a result, in some of the analyses, only the synthetic datasets were used.

TABLE 3.2 below displays the information regarding the traces. Each trace was collected from one of the shuttle routes, so the true travel lengths are equal to the corresponding route length as shown in the table.

TABLE 3.2 UConn Shuttle Routes

Index	Route	Route ID	Route Length (m)	Round Trip Time along Route (Rounded to nearest 5 minutes)
1	Blue	bl	9176.74	35
2	Green	gr	10466.78	40
3	Orange	or	5119.45	25
4	Purple	pl	17018.93	60
5	Red	rd	7294.30	30
6	Yellow	yl	9992.98	35

The route lengths in TABLE 3.2 are calculated from the shapefile used for creating the road network. In addition to the route length, information about links along the route are used to assess accuracy of the map-matching algorithm.

3.7.2 Performance Metrics

To assess accuracy of the map-matching algorithm quantitatively, two types of performance metrics were utilized as described below.

Length-based Metrics

In this approach, the map-matched length is compared with the true length for the trip.

$$Error = \frac{Calculated\ Path\ Length - Actual\ Route\ Length}{Actual\ Length}$$

$$Score = 1 - |Error|$$

Error indicates the percent deviation from the true length of the traces. On the other hand, Score shows the percentage length accurately retrieved by the Map-Matching Algorithm. Unlike error, score value equally penalizes overestimation and underestimation of the travelled distance. The values are presented as percentages in the results.

Identification-based Metrics

In this approach the map-matching results are compared at a disaggregate level by comparing the links identified in the map-matched path versus the links in the original shuttle route. In comparing the set of map-matched path an original shuttle route, a confusion matrix *(40)* can be constructed as shown in TABLE 3.3. As can be seen, the confusion matrix allows one to not only check what was retrieved correctly but also assess what was not retrieved correctly, what was wrongly retrieved, and what was correctly not retrieved.

TABLE 3.3 Confusion Matrix for Link Identification

		Ground Truth Route		
		On-Route	**Not On-Route**	**Total**
Map-matched Links	**Retrieved**	Correctly Retrieved	Incorrectly Retrieved	# of Links Retrieved
	Not-Retrieved	Incorrectly Not Retrieved	Correctly Not Retrieved	# of Links Not Retrieved
	Total	# of Links On-Route	# of Links NOT On-Route	# of Links in the Network

The definitions of the particular metrics derived from the confusion matrix to assess performance are given below. Again these metrics are presorted as percentages in the results.

1. **Recall** : Percentage of on-route links that are correctly retrieved in map-matching process

$$Recall = \frac{\#\ of\ Links\ Correctly\ Retrieved\ ones}{\#\ of\ Links\ Correctly\ Retrieved + \#\ of\ Links\ Incorrectly\ Not\ Retrieved} = \frac{\#\ of\ Links\ Correctly\ Retrieved}{\#\ of\ Links\ on\ Route}$$

2. **Precision** : Percentage of link actually on-route out of all links retrieved

$$Precision = \frac{\#\ of\ Links\ Correctly\ Retrieved}{\#\ of\ Links\ Correctly\ Retrieved + \#\ of\ Links\ Incorrectly\ Retrieved} = \frac{\#\ of\ Links\ Correctly\ Retrieved}{\#\ of\ Links\ Retrieved}$$

3. **Accuracy**: Percentage of links either retrieved or omitted correctly out of all links in the network. It indicates an overall balance between conservative exclusion to reduce false alarm and liberal inclusion of links that are not actually on the route.

$$Accuracy = \frac{\#\ of\ Links\ Correctly\ Retrieved + \#\ of\ Link\ Correctly\ Not\ Retrieved}{\#\ of\ Links\ in\ the\ Network}$$

4. **Specificity** : Percentage of links correctly not-retrieved out of all links actually not on that route

$$Specificity = \frac{\#\ of\ Link\ Correctly\ Not\ Retrieved}{\#\ of\ Link\ Correctly\ Not\ Retrieved + \#\ of\ Links\ Incorrectly\ Not\ Retrieved} = \frac{\#\ of\ Link\ Correctly\ Not\ Retrieved}{\#\ of\ Links\ Not\ on\ Route}$$

5. **False Negative Rate (FNR)** : Percentage of links failed to retrieved out of the ones on-route

$$FNR = \frac{\#\ of\ Links\ Incorrectly\ Not\ Retrieved}{\#\ of\ Links\ Correctly\ Retrieved + \#\ of\ Links\ Incorrectly\ Not\ Retrieved} = \frac{\#\ of\ Links\ Incorrectly\ Not\ Retrieved}{\#\ of\ Links\ on\ Route}$$

3.8 Results

In the crowdsourced traveler information systems application, smartphones are the source of GPS points. So the level of noise in GPS is largely beyond control. However, what can be controlled or at least requested is the frequency at which GPS points should be sampled and shared with the crowdsourced project. As a result, the focus in the analysis is on assessing the accuracy of the map-matching algorithm as a function of the GPS sampling frequency.

Disaggregate Summary of Performance Metrics

A disaggregate quantitative summary of all the performance metrics for the 270 resampled GPS traces is shown in the TABLE 3.4. The performance metrics represent average value for a given route and sampling frequency – averages are calculated across 9 sets of resampled GPS datasets

representing different days of week (i.e. Tuesday, Wednesday and Thursday) and different times of day (i.e. 7-11 AM, 12-4 PM, 4-8 PM). Intuitively, the most important factor influencing map-matching performance should be just the GPS sampling frequency. So the different metrics are aligned with the frequency in the table. Also, different routes with their individual topological characteristics should have some influence on the values. So the analysis results are shown accordingly in the table below. The first two columns in the table present the length-based performance metrics and the remaining columns presents disaggregate identification-based metrics. It must be noted that all metrics are reported as percentages. Except Error and False Negative Ratio (FNR), for all the other metrics a value of close 100% represents good performance of the algorithm. For the FNR and Error metrics, a value of 0% represents good performance of the map-matching algorithm. It appears that the map-matching algorithm performs very well for all combinations of route and sample frequency.

TABLE 3.4 Disaggregate Route-wise Summary of Performance Metrics

		Based on Length (%)		Based on Link Identification(%)				
Route	**Freq**	**Error**	**Score**	**Recall**	**Precision**	**Accuracy**	**Specificity**	**FNR**
bl	10	0.77	98.33	99.54	99.38	99.56	99.58	0.46
	15	0.28	98.80	99.85	99.23	99.63	99.48	0.15
	30	-0.71	98.95	100.00	99.38	99.75	99.58	0.00
	45	-1.97	98.03	100.00	98.92	99.56	99.27	0.00
	60	-1.21	98.50	100.00	98.30	99.31	98.85	0.00
gr	10	0.16	98.57	97.90	99.60	99.19	99.82	2.10
	15	-1.19	98.73	98.64	99.60	99.44	99.82	1.36
	30	-3.32	96.68	99.02	99.60	99.56	99.82	0.98
	45	-4.41	95.59	99.81	98.81	99.56	99.46	0.19
	60	-5.45	94.55	100.00	96.83	99.00	98.56	0.00

or	10	-0.51	98.66	99.01	97.78	99.19	99.25	0.99
	15	-1.09	98.29	99.26	97.78	99.25	99.25	0.74
	30	-1.14	98.24	100.00	97.78	99.44	99.25	0.00
	45	-1.96	98.04	99.27	97.78	99.25	99.25	0.73
	60	-2.25	97.13	100.00	97.78	99.44	99.25	0.00
pl	10	-0.22	99.25	97.81	100.00	98.94	100.00	2.19
	15	-0.38	99.29	98.31	100.00	99.19	100.00	1.69
	30	-0.71	99.23	98.56	100.00	99.31	100.00	1.44
	45	-1.74	98.26	98.19	99.87	99.06	99.88	1.81
	60	-1.47	98.50	97.20	99.74	98.50	99.75	2.80
rd	10	-0.54	98.63	98.84	99.33	99.31	99.60	1.16
	15	-0.96	98.57	99.33	99.16	99.44	99.51	0.67
	30	-1.60	98.17	99.83	98.99	99.56	99.41	0.17
	45	-2.00	97.78	100.00	98.14	99.31	98.91	0.00
	60	-2.63	97.37	100.00	98.31	99.37	99.01	0.00
yl	10	0.47	98.80	95.68	100.00	98.75	100.00	4.32
	15	-0.23	98.51	97.20	100.00	99.19	100.00	2.80
	30	-1.47	97.72	99.33	100.00	99.81	100.00	0.67
	45	-3.30	96.19	99.11	99.77	99.69	99.91	0.89
	60	-3.00	97.00	99.78	99.77	99.88	99.91	0.22

Note: Values in each cell is averaged over 9 map-matching tasks

GPS Frequency and Estimation of the Travelled Length

Following the definitions in Section 3.6.2, the Length-based performance metrics, Error and Score are shown in FIGURE 3.3 and FIGURE 3.4for different routes. These two charts include the raw GPS traces along with the resampled GPS traces. From FIGURE 3.3, the influence of GPS frequency on Error values indicate a systematic trend. The Error values are shown in

percentage and the most desirable outcome is an Error of 0.0%. Higher positive percent values indicate overestimation, and similarly, negative values indicate underestimation of the true length.

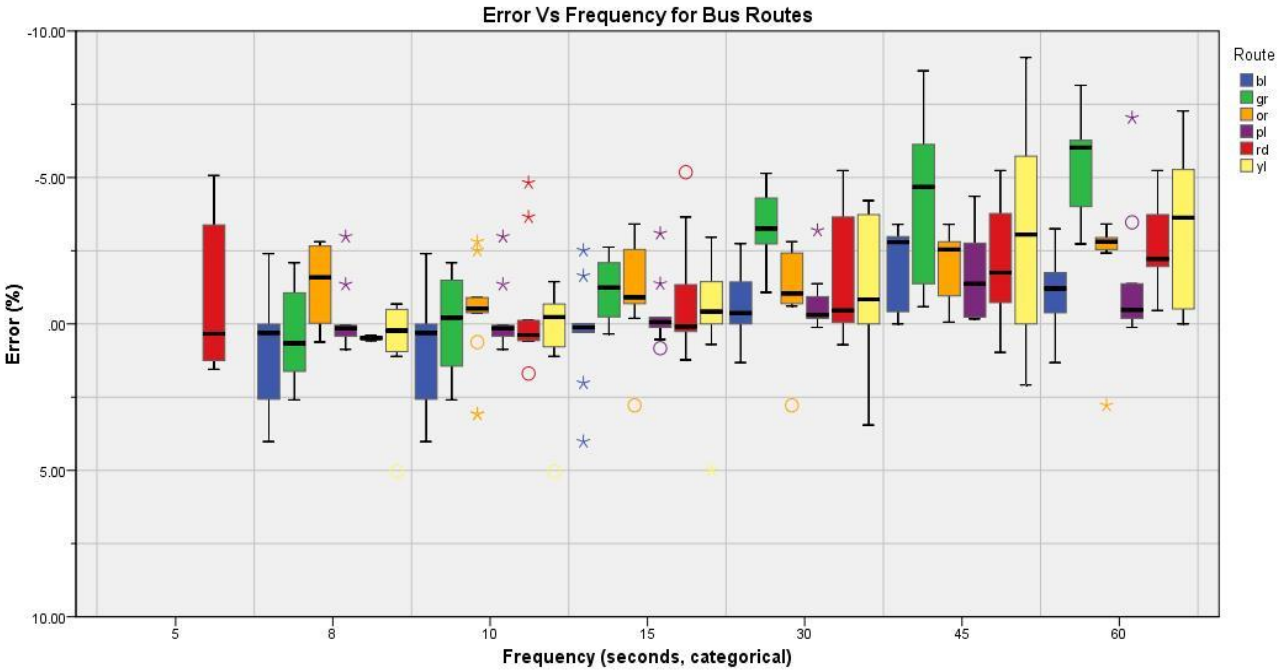

FIGURE 3.3 Error for GPS Frequency and UConn Shuttle Route

In the figure, the values of the mean percent Error in the boxplots at low GPS frequency (60 sec) move towards the negatives showing underestimation of the length of the true path. On the other hand, at high-frequency range (10 sec or even less), the mean values shifts downwards, showing tendency to overestimate the traversed length. The Map-Matching Algorithm tends to predict the traversed length quite well in the 10-15 seconds sampling frequency. In that range, not only the mean Errors are closer to 0%, but also the ranges of Errors are narrow, pointing to consistent results.

This underestimation at high frequency and over-prediction at low frequency can be easily explained by inspecting the definition of the transition probability in the Viterbi Algorithm. Due to emphasis on the straight line segments for transition probability calculations, the low frequency GPS data (or higher time interval) should short-circuit the path between two candidate points if such shortcut exists and deem such path more probable than a circuitous shortest path between another pair of candidate points. The widening interquartile range on the left and right ends are also possible outcome of this fact. Such trend is also visible for each Shuttle route individually, marked with colors and their route ID's as listed in TABLE 3.1. The lone boxplot at 5 second frequency on the left comprises of only 6 raw GPS traces for red route.

A similar trend is visible in FIGURE 3.4 where the Score values are visualized in the boxplots.

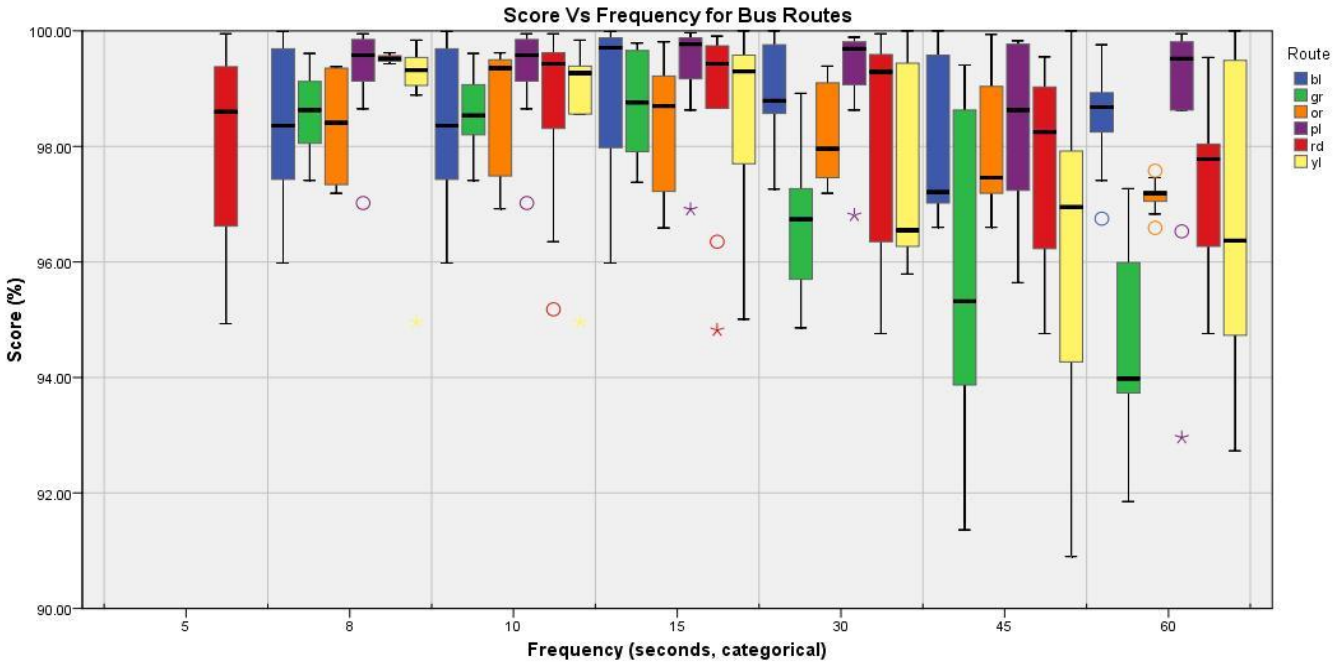

FIGURE 3.4 Score for GPS Frequency and UConn Shuttle Route

Since the Score values mean the percentage of the actual length that was correctly retrieved, this downward trend in FIGURE 3.4 maintains a relationship with that in FIGURE 3.3. In this FIGURE 3.4, irrespective of the Shuttle routes, the Score gradually drops as GPS frequency is lowered from sub-10 second zone towards 60 second block. The perfect Score is 100% when the total traversed length is successfully retrieved.

This is consistent with the expectation, because Score values are supposed to penalize both over and underestimation of the distance travelled from the observed GPS trace. It is also interesting to observe that at lower GPS frequency (closer to 60 sec), the Score values have a wider range. From FIGURE 3.3, it is seen that the magnitudes of over and underestimation for higher frequency (10 to 15 seconds) are within a narrower range, which is more clearly reflected in FIGURE 3.4.

A quantitative summary of the two plots can be found in TABLE 3.5. It gives the range and mean values for both Error and Score aggregated across all routes.

TABLE 3.5 Length-based Performance Metrics

GPS Freq (sec)	Error (%)			Score (%)		
	Minimum	Mean	Maximum	Minimum	Mean	Maximum
10	-4.82	0.02	5.04	94.96	98.71	99.99
15	-5.18	-0.60	4.99	94.82	98.70	100.00
30	-5.24	-1.49	3.45	94.76	98.17	100.00
45	-9.10	-2.57	2.08	90.90	97.31	100.00
60	-8.15	-2.67	2.78	91.85	97.17	100.00
Note: Values are from averages over 54 map-matching tasks averaged over all routes, TOD, DOW						

Each cell value in TABLE 3.5 is averaged over 54 map-matched GPS traces from the resampled GPS traces. It can be seen that, the mean error values decreases from positive 0.02%, showing slight overestimation of travelled length, towards negative -2.67%, indicating underestimation. Similarly the range between minimum and maximum also slides down in the same direction. At the same time, mean Score values indicate, estimation of travelled length is closer to actual value (98.71%) when GPS frequency is low (10 sec) whereas at 60 second sampling frequency, it drops to 97.17%. These values in TABLE 3.5 summarize the key findings from this exploration.

The length-based metrics are visualized along the day and time information in FIGURE 3.5 and FIGURE 3.6. FIGURE 3.5 displays the matrix of boxplots for Error values for different days of week along its rows. The common theme of increasing percentage Error values with higher GPS frequency is visible across every route. This is consistent with the findings in FIGURE 3.3.

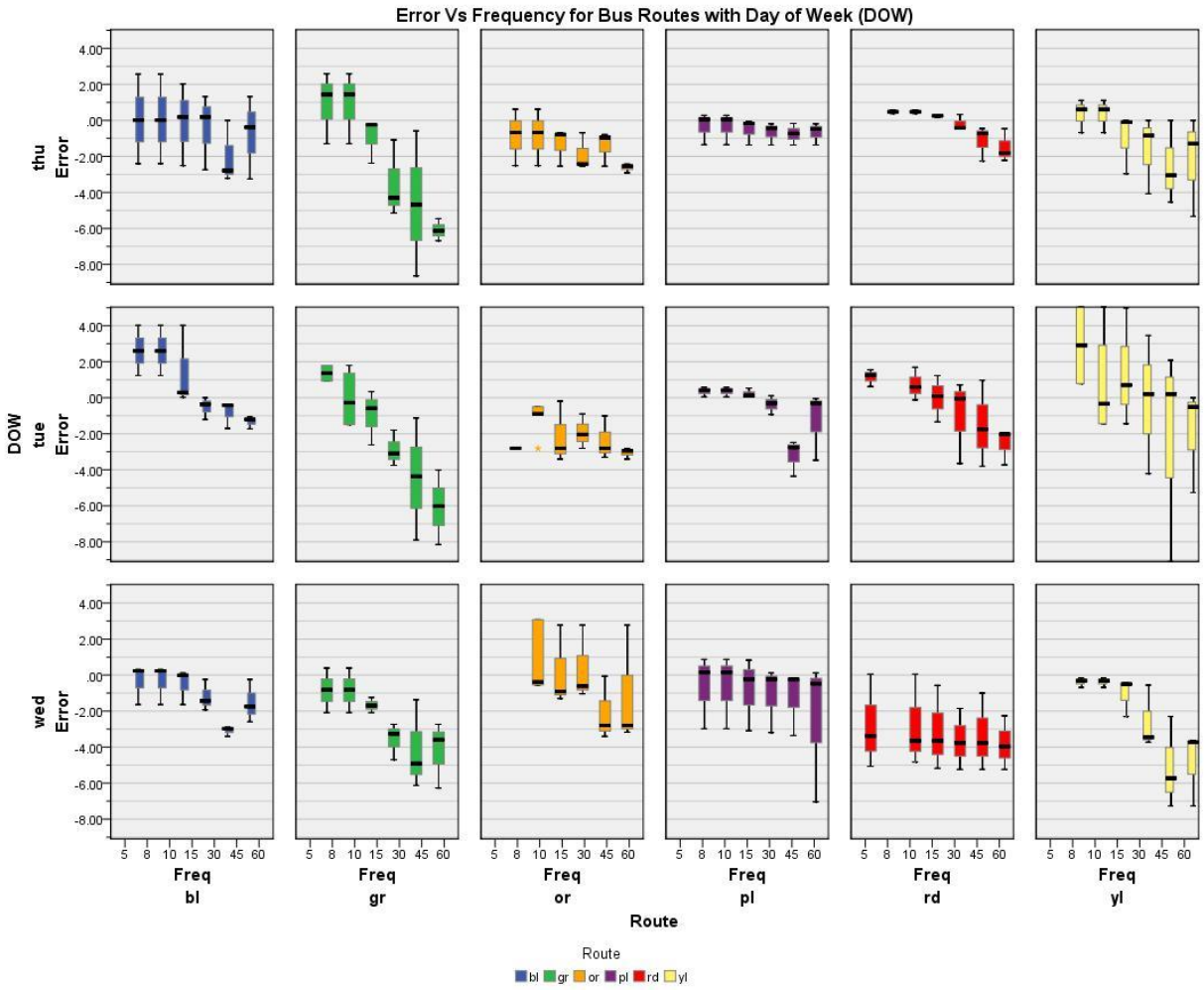

FIGURE 3.5 Day-wise Error for GPS Frequency and UConn Shuttle Route

A similar exploration for different time segments is in FIGURE 3.6. In this FIGURE, the rows are showcasing the Error values for morning, afternoon and evening periods (labeled as ‘mo’, ‘af’, ‘ev’ in the figure) based on the 324 map-matched traces.

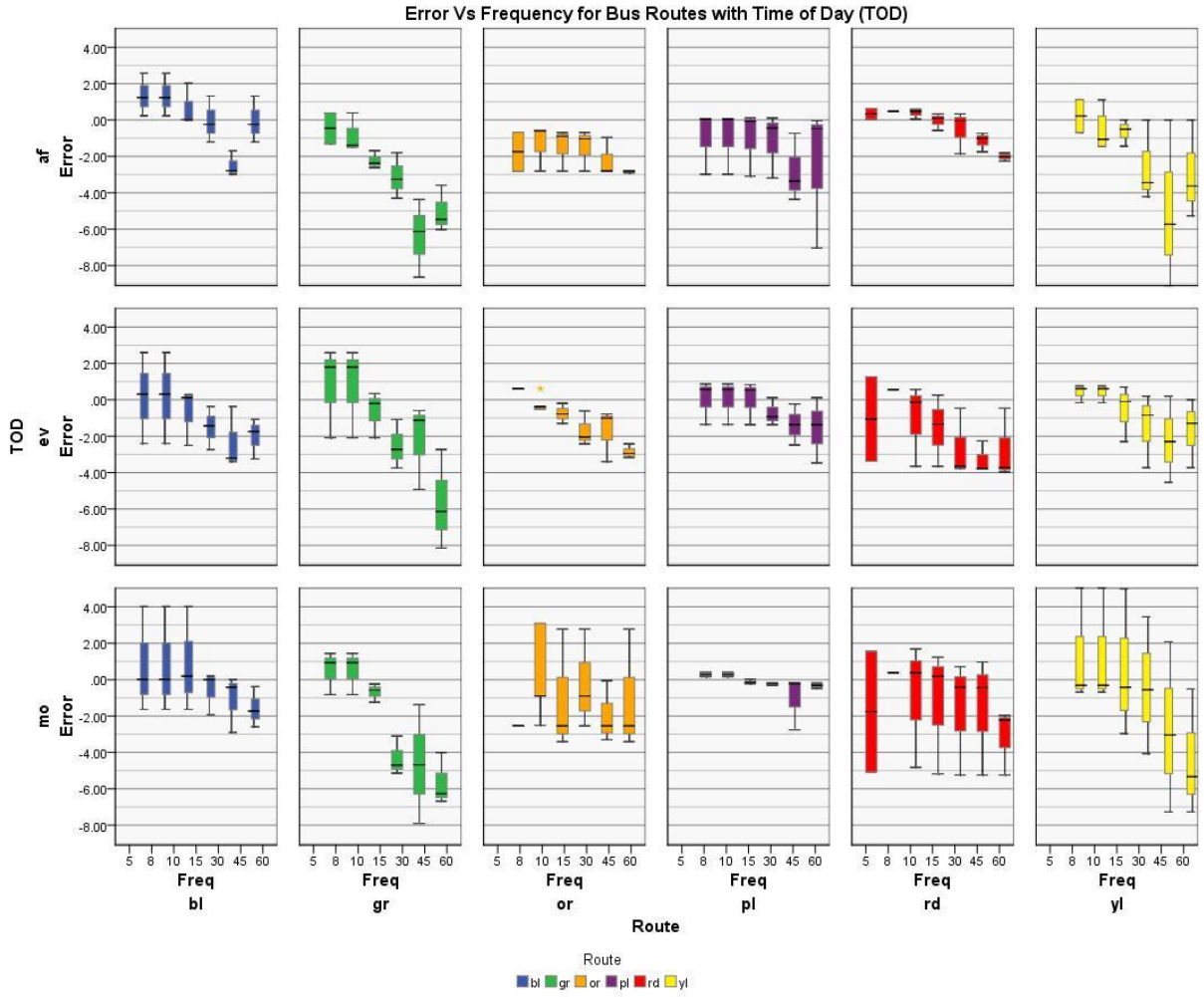

FIGURE 3.6 Time-wise Error for GPS Frequency and UConn Shuttle Route

The observations for FIGURE 3.6 are the same as in FIGURE 3.5 and the variation in interquartile range is only due to rearrangement of the outputs. This is perfectly along the expectation that GPS traces would not be influenced by day or time. It also confirms that the original traces are all consistent and has no unexpected pattern.

Comparing with the results from Newson and Krumm *(21)*, these results are consistent, where lowering the GPS frequencies are observed to deteriorate the accuracy as well. But the reported

values were the ratio of total length either falsely added or falsely omitted and the true length. So in that sense, the direct numerical comparison is not possible.

GPS Frequency and Identification of True Path

Out of all the Identification-based metrics, the Accuracy is the one measuring a balance between false-alarm and insensitive overlooking of a link. As the Accuracy values are all close to 100%, a table with *100-Accuracy(%)* values as below accentuates their incompleteness. This way, the relative difference between Accuracy values for different routes and frequencies are more prominent, as seen in TABLE 3.6. This *100-Accuracy(%)* values are termed as *'inaccuracy'* in this section.

TABLE 3.6 Stability of Accuracy Measure Accross GPS Sampling Frequency

GPS Freq	100 - Accuracy (%)						
	Route						
	bl	gr	or	pl	rd	yl	Average
10	0.44	0.81	0.81	1.06	0.69	1.25	0.84
15	0.37	0.56	0.75	0.81	0.56	0.81	0.65
30	0.25	0.44	0.56	0.69	0.44	0.19	0.43
45	0.44	0.44	0.75	0.94	0.69	0.31	0.59
60	0.69	1.00	0.56	1.50	0.63	0.12	0.75

Note: Value in each cell is averaged over 9 map-matching tasks

From the TABLE 3.6, the average *inaccuracy* values across all routes for a particular GPS sampling frequency do not reveal much interesting information. Rather, for any route, the

inaccuracy value is the minimum at 30 second sampling frequency, which is the middle-ground of the sample frequency range. This is visualized in FIGURE 3.7 below.

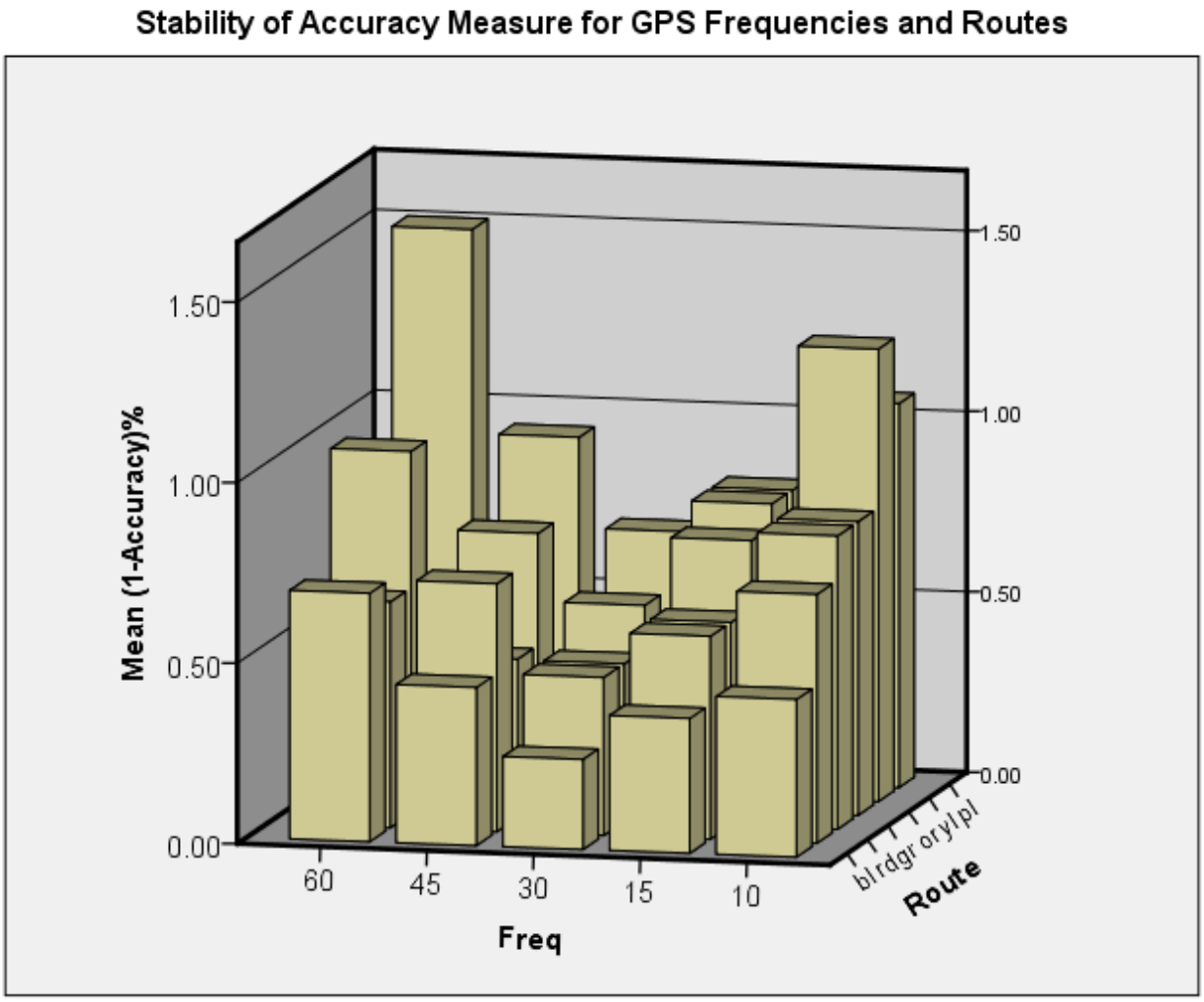

FIGURE 3.7 Stability of Accuracy Measures across GPS Frequencies

In the figure, it is observed that the *100-Accuracy(%)* values are lowest at the middle of the GPS sampling range (30 seconds), where lower value is the better. It indicates that irrespective of the true-path, any extreme scheme with GPS sampling frequency would have lower Accuracy values compared to the moderate ones.

This is consistent as what is seen in length-based metrics. There, with high sampling frequency such as 10 seconds, the lengths tends to over-predict due to GPS noise, causing more links to be picked-up in the map-matching process than the true structure of the path. It also means that the false positives are raised, lowering the overall Accuracy value. On the other hand, low frequency tends to under-predict lengths and misses out some of the links that are on the true path, leading to to higher false negatives. At the same time, inclusion of alternative short-cuts in the retrieved path, increases the false positives as well. Consequently, it affects the Accuracy values harshly.

Thus, while any particular trend for other identification-based metrics is not readily visible, the Accuracy presents interesting insights. It tells us that at medium GPS frequency, the most balanced and potentially optimum results would be expected.

False Negative Ratio (FNR) is shown in FIGURE 3.8 below. Since FNR is the rate at which correct links are mistakenly overlooked, it is exact opposite of Recall.

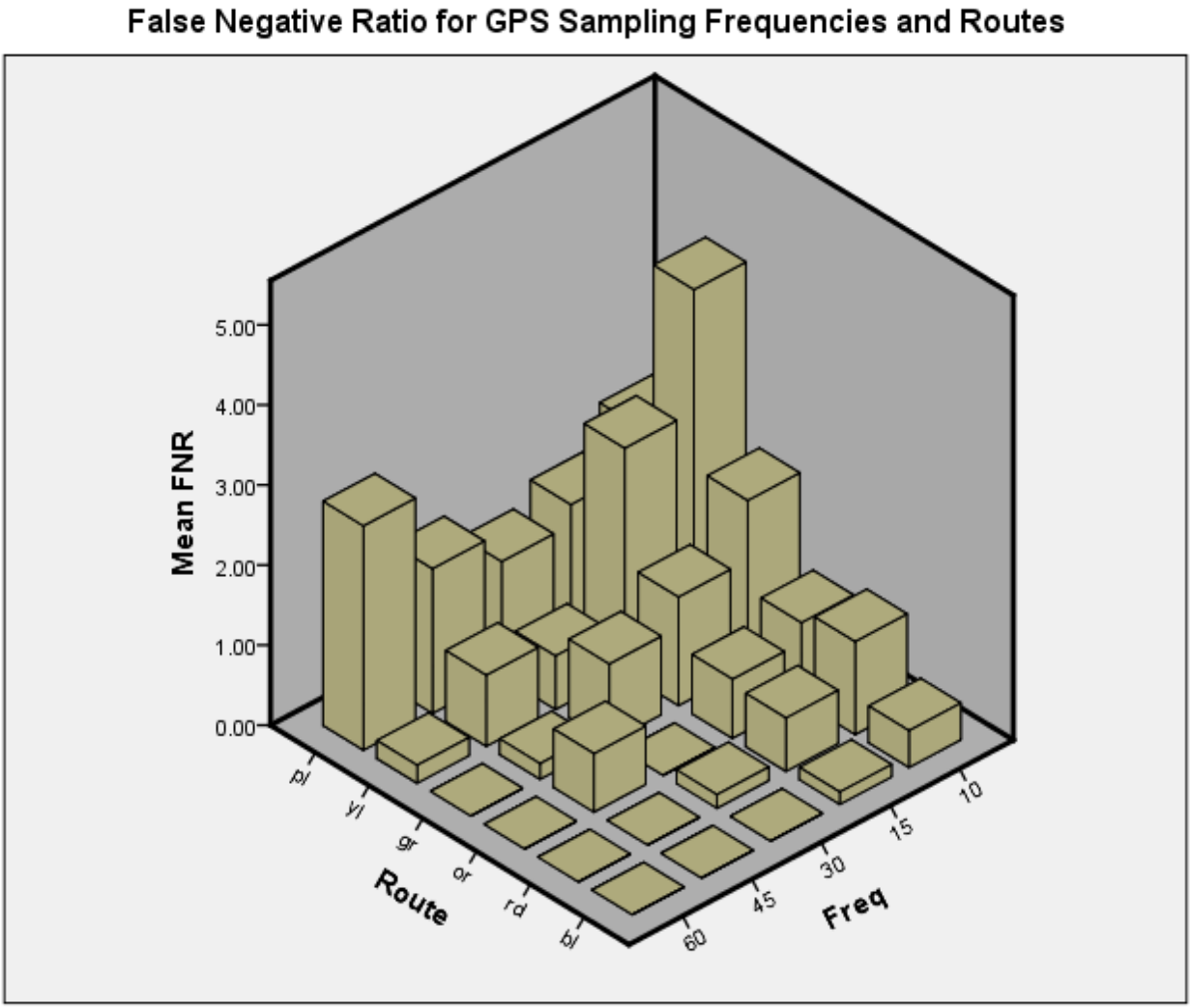

FIGURE 3.8 False Negative Ratios across GPS Frequencies

In the figure, mean FNR values across multiple map-matching tasks show that in most routes, the FNR values are quite low at low frequency. Except purple route, the FNR values are worse at high GPS sampling frequency. Apart from Purple route, this is manifestation that, at high frequency GPS noise would lead to picking up many more links than in the true path. This is why at lower GPS sampling frequency, the false negatives are low. But in purple route, the distance is longer than the other routes and the probable reason could be the presence of

multiple sharp-turns and U-turns, where shortcuts would be picked over the actual path. Thus it indicates topology of the network is crucial in Map-Matching problems.

From the perspective of Recall values, the same explanation from FIGURE 3.8 would be applicable in this case as well due to its mutual relationship with FNR.

3.9 Conclusions, Limitations and Future Directions

From the results of map-matching performance, the following conclusions can be drawn,

1. A trimmed smaller network can be used with transit services for map-matching the GPS data with quite impressive performance. This approach can also offer freedom for obtaining transit, non-transit or non-motorized traffic information for a rural network.
2. For finding travel length from GPS data, moderate to high frequency sampling frequency (10-15 sec) is desirable. Moderate frequency would be most likely to estimate the length within a narrow margin of error. A slight overestimated length can be found from high frequency GPS sampling (45-60 sec), but the possible range of Error is still narrower than low-frequency GPS data (5-8 sec), where the lengths are underestimated and may have Errors with a relatively wider margin compared to the other ones.
3. For the problem where identification of the links traversed is of interest, Accuracy is most useful metric for evaluation of performance. Both the false positives and false negatives are captured well in this metric. Generally, real-time traffic information would require minimization of both kinds of errors. Using Recall, precision or Specificity would offer insights of the bias of the outcome, but Accuracy measures are most useful in this case. A medium range of frequency (30 sec) offered most balanced performance.

Nonetheless, based on the results and the procedure employed for the map-matching process, the following limitations are visible.

1. The map-matching exercise only tests performance based on GPS sampling frequency and different paths. The GPS noise is not inspected in this exercise, which is another major factor determining map-matching performance.
2. Network structure and road density is an important factor in the Map-Matching performance. The results found in the analyses would not be readily applicable for another road network, though the trends and patterns may hold.
3. The performance is extremely high due to a very small rural network. Application of the same algorithm in large, real road network may be of interest.
4. Comparison of the performance of this implementation with other established map-matching packages such as OSRM *(36)* would be useful.

Future research directions regarding the Map-Matching problem are below

1. Traffic speeds can be found from the map-matched traces. This is the basis of the travel-time prediction mechanism and crowdsource simulation process in Chapter 4.
2. Map-matched traces are useful for identification of travel mode from GPS traces. Using GPS traces from smartphones and combining accelerometer and gyroscopes sensors, mode detection can be energy-efficient and can offer very wide coverage.
3. The implemented map-matching algorithm is soon to be released as open-source project. An interesting improvement would be to use the emission probability definition proposed by Oran and Jaillet *(30)* and the transition using Conditional Random Field (CRF) proposed by Hunter *(34).*

CHAPTER 4 TRAVEL TIME PREDICTION FROM CROWDSOURCED GPS TRACES

4.1 Introduction

As mentioned in previously, the crowdsourced Real-Time Traveler Information Systems (RTIS) relies on the users to get the traveler information. Their contribution can be ensured with a reward-system i.e., by informing them the travel condition that enables better travel decisions. Disaggregate traveler information from the users, when mined, can be used to predict the travel time or Estimated Time of Arrival, that are the premier currencies of this reward system.

4.2 Motivation

While real-time information for all the links in the network would make the prediction a trivial task, practically, sparse real-time crowdsourced data poses difficulty in inferring real-time travel conditions for all the roads due to absence of information for some links in the network. So, the connectivity between links can be used to infer the travel condition for the unobserved links by borrowing real-time information from neighboring links and combining it with historical information for those unobserved links. This problem is not much explored from crowdsourcing perspective yet.

In this chapter, the feasibility of predicting travel conditions from crowdsourced GPS traces is explored by demonstrating a travel time prediction algorithm, implemented as part of the prototype traveler information system, in different scenarios. It also demonstrates how absence of real-time information of a portion of the network affects the prediction of travel condition. A simulation of the live crowdsourced system shows that, even with a sparse real-time picture of the network, the travel condition across the network can be predicted quite well.

4.3 Objectives

This chapter focuses on predicting travel conditions of the network for the current time-segment with sparse crowdsourced data. In order to achieve this objective, the travel time for any origin-destination pair and travel speeds for each of the links in the network are predicted. The other objective is to understand the effect of the network coverage on the prediction quality. The prediction of link travel speeds for all links in the network using sparse information is also useful for rural context where most roads have minimal traffic and it is not feasible to deploy extensive sophisticated traffic sensor networks.

4.4 Literature Review

As mentioned earlier, the implemented prototype system is to be demonstrated on UConn shuttle transit routes but it also intends to remain generalizable for other non-transit modes. Because travelers on transit has to wait at the originating transit stop, generally, they would find the Estimated Time of Arrival (ETA) at desired transit stops a more useful traveler information than the estimated travel time between the origin and the destination. Similarly, non-transit travelers would find ETA and travel time to destination almost synonymous. While travel time and ETA can be derived from one another in theory, their particular implementation often prohibits the conversion. The literature review explores the approaches for predicting ETA and travel time, both for transit and non-transit modes. Eventually it is seen that none of the existing approaches, except that in Trajectory Regression allows prediction of travel condition for the entire network with sparse crowdsourced data. Since this particular approach predicts travel time between origin and destination pair, ETA for any mode can eventually be derived with ease, even between transit stops. But this thesis only demonstrates the travel time prediction tasks.

Predicting ETA from GPS in real-time has been explored in many ways in the past. But predicting ETA for a road link has been only possible when the link has observed real-time data. The focus had mostly been on either for an individual vehicle fitted with an Automatic Vehicle Location (AVL) system, such as turn-by-turn direction services with GPS or for vehicles plying along a predefined path, such as transit vehicles. These approaches are not along the current problem.

ETA, or more generally travel time, problem has been approached with diverse strategies from many different fields. A number of approaches has been proposed and tested specifically for transit routes. Due to fixed route and schedule, predicting ETA at bus stops has been explored using simple algorithms by Lin and Zen *(41)*, where predictions are for fixed known routes. Similarly, Sun et al. *(42)* simply finds route from preset information and re-evaluates speed for travel time prediction along the linearized routes. On the other hand, Shalaby and Farhan *(43)* used Kalman Filtering to re-evaluate the position and update estimates based on current information and combines dwelling time model in the underlying framework. Still, all these approaches are centered on a fixed route and predicts only for that.

On the other hand, Bin et al. *(44)* proposed Support Vector Machine (SVM) to predict travel time between two bus-stops. In this case, the concept of physical roads is not explicitly present and historical training data of travel time between specific transit stops are regressed. So, even though it is suitable for predicting ETA at certain origin-destination pairs, i.e. from transit stop to another stop, it is restrictive for a Trajectory Regression problem. Similarly, Hierarchical Artificial Neural Networks (HANN) models are proposed for the same problem by Lin et al. *(45)*, which has the same limitation in solving the problem at hand. Both of the last two were reported to outperform the Kalman Filter based models. While, GTFS *(46)* based approach by

Zimmerman et al. *(47)* predicts ETA based on schedule and historical data over the previous month, which is not generalizable for other non-transit modes.

In the approach proposed by Tao et al. *(48)*, link traffic conditions are updated dynamically using Kalman Filtering in a non-transit case. This approach finally aggregates the individual link travel times for multiple vehicles and then updates the traffic condition for that time segment. As a limitation, it requires coverage of each link and does not consider spatial and road connectivity. A simple ordinary least square regression technique has been tested for real-time routing suggestion and ETA prediction for non-transit mode by Karbassi and Barth *(49)*.

Jenelius and Koutsopoulos *(50)* uses taxi GPS traces to estimate link travel time in an urban condition with sophisticated models considering time lost at intersection and employing spatial moving average for smoothness in travel time estimation across neighboring links. This consideration of spatial connectivity between links is promising and this methodology can be utilized for future prediction of travel time, but real-time prediction for unobserved links is still a different problem. Most of these models can be termed as Segment Models, since they focus on only the most frequently travelled segments of the road network as says Yang et al. *(51)*, which may always have sufficient real-time information update. Also, a spatio-temporal auto-regressive model has been proposed by Tulic et al. *(52)* where link travel times for the next time-segment is predicted maintaining the spatial smoothness. But network-wide real-time travel condition prediction problem using sparse GPS data is not addressed in this approach.

Rather, the problem at hand is most relevant with 'Trajectory Regression' problem proposed by Ide and Sugiyama *(53)*. In this approach, the algorithm looks into the problem of sparse trip data. In this approach, termed as 'Trip Model' by Yang et al. *(51)*, when the some portion of the

road-links in the network is not observed, the data for observed links are borrowed from neighboring links to infer information based on the network connectivity. Their approach considers the similarity between paths in trips. Trajectory Regression is augmentation of the Gaussian Processes Regression (GPR)-based approach, where path similarity is calculated by a kernel function as proposed by Ide and Kato *(54)*. But in practical sense, selecting best kernel function is an arbitrary process and thus, the Trajectory Regression problem was cast as a Regularized Least Square (RLS) problem. Here the link cost is inferred for all the links in the network, regularizing the least square errors by the Laplacian matrix for the network graph which captures the similarities between the links as does its predecessor, the kernel-based GPR formulation. This RLS formulation reduces to a form which is similar to Kernel Ridge Regression (KRR), allowing much flexibility in defining the graph Laplacian, unlike the rigid ad-hoc kernel matrix in the GPR-based one. Ide and Sugiyama *(53)* is concerned about predicting the travel time between any origin-destination node pair within the network and in this process, it can be also used to annotate each of the unobserved links with a predicted cost value. This is very much along the focus of this crowdsourced system. Once the travel time for any node-pairs is predicted, prediction of ETA can follow easily.

4.5 Trajectory Regression Problem: Overview

4.5.1 Model

In essence, the Trajectory Regression problem, according to Ide and Sugiyama *(53)* is a least square regression problem with the estimated values regularized through a restricted structure. So essentially, this is formulated as a Tikhonov regularization or Kernel Ridge Regression problem, which is also known as a shrinkage method. The implemented travel time prediction

algorithm is based on Ide and Sugiyama *(53)*, so most expression have the same notations from the original formulation, with adjustments applied as required.

Let, the travel time for GPS trace *y*, have a Gaussian distribution based on the trajectory $x = \{e: e \in x\}$, formed by the set of links *e* in the map-matched path

$$p(y|x) = N(y|\mu(x), \sigma^2) \qquad \text{EQUATION 4.1}$$

Where, the mean travel time on the trajectory is given by,

$$\mu(x) = \sum_{e \in x} l_e(\phi_e^0 + f_e) \qquad \text{EQUATION 4.2}$$

The Gaussian assumption is quite realistic for this problem at hand and eases the subsequent formulation. Here, $l_e$is the length travelled on link e. In addition, $\phi_e^0$is the baseline or average cost for the link e, which is the inverse of the average speed (time per unit distance) in this context. The state variable f_e indicates the deviation of cost at current time from the average cost on the link e. Details of these are described later.

4.5.2 Trajectories

Assuming the network has *M*-number of road links and there are *N*-number of real-time map-matched GPS traces, the map-matched trajectories in the GPS traces are stored in the MxN dimensional *link indicator matrix*, Q.

$$Q \equiv [\boldsymbol{q}^{(1)}, \boldsymbol{q}^{(2)}, \ldots, \boldsymbol{q}^{(N)}]^T \in \mathbb{R}^{NxM} \qquad \text{EQUATION 4.3}$$

In each of the elements in the matrix $\boldsymbol{q}^{(n)} \in \mathbb{R}^{Mx1}$, the *m*-th row in the vector $\boldsymbol{q}^{(n)}$ holds the length travelled on the *m*-th link in the *n*-th trajectory $x^{(n)}$. So, $q_m^{(n)} = l_m^{(n)}$, where

$$l_m^{(n)} = 0 \; if \; m \notin x^{(n)}, l_m^{(n)} > 0 \; if \; m \in x^{(n)} \; for \; 1 \leq m \leq M$$ EQUATION 4.4

As the original formulation allows both, link weights or physical lengths of links for non-zero values in $\boldsymbol{q}^{(n)}$, here, if the map-matched GPS traces has travelled the link more than once in the trajectory, $l_m^{(n)}$ may be more than the physical length of the *m*-th road link. Generally, most links would not be traversed during such short GPS traces, so most rows in the $\boldsymbol{q}^{(n)}$ vector remain zero.

4.5.3 Least Square from Observed Travel Times

The least square part of the model minimizes the error between the predicted travel time and the actual travel time. This part is constructed as a scalar value Ψ_{LS}

$$\Psi_{LS} = \; || \, \boldsymbol{y}_N - \; Q^T \boldsymbol{f} ||_2^2$$ EQUATION 4.5

The expression is the squared L_2-norm of the resulting vector $\boldsymbol{y}_N - \; Q^T \boldsymbol{f}$. In this expression, $\boldsymbol{y}_N$ is an N-dimensional vector which stores the deviation of the current real-time travel time from the baseline travel time for the associated trajectories according to EQUATION 4.6.

$$\boldsymbol{y}_N \equiv \; \boldsymbol{y} \; - Q^T \boldsymbol{\phi}^0$$ EQUATION 4.6

The N-dimensional vector $\boldsymbol{y}$, is the observed or experienced trip duration corresponding to the *N*-GPS traces and Q is the link indicator matrix. Where $\boldsymbol{\phi}^0$ is the M-dimensional *baseline cost vector*, whose *m*-th row holds the value of average cost for the *m*-th link in the network. Similarly, the vector $\boldsymbol{f} \in \mathbb{R}^{\boldsymbol{MX1}}$ holds the unknown state variables for all *M*-links in the network, indicating the deviation between current cost on *m*-th link, compared to the baseline cost in $\boldsymbol{\phi}^0$. The baseline cost is described later.

4.5.4 Incorporating Network Connectivity

With the least square part of the objective function Ψ_{LS} , the state variables for the unobserved links cannot be estimated. So Ide and Sugiyama *(53)* also considered a scheme to propagate the information among neighboring links, stabilizing and smoothing the estimation of the state variables. To this end, the neighboring links are assigned an affinity measure, which finally lead to a graph Laplacian matrix. Because in any standard graph representing a road network, physical roads are represented as edges and intersections as vertices, both terms are used in this section depending on the context.

To calculate the affinity between the links (edges), the number of intersections (nodes) in between is used as the criterion. This is intuitive because if the links are adjacent, then one link may influence the other more, but a link may not influence the ones sufficiently farther apart. Also the links whose state variables have values are similar, should experience similar departure from the average condition. Combining these two conditions, a regularizing constraint can be introduced in the objective function. Thus penalty or regularization term in addition to the least square error term in EQUATION 4.5 is

$$\Psi_{Reg} = \sum_{m=1}^{M} \sum_{m'=1}^{M} S_{m,m'} |f_m - f_{m'}|^2 \qquad \text{EQUATION 4.7}$$

The affinity value for a pair of links comes from a decaying function based on an arbitrary fractional weight value *w*, with $0 < w < 1$. The elements in the affinity matrix *S*, are defined as

$$S_{m,m'} = w^{dist(m,m')}, if\ dist(m,m') \leq dist_U, otherwise\ 0 \qquad \text{EQUATION 4.8}$$

$dist(m, m')$ is the distance between the links of the road network graph measured by number of intersections separating the links m and m'. In special case, $dist(m, m') = 0$, if $m = m'$, ensuring that the affinity for the same link has a unit value. The affinity matrix is easily calculated from the edge-to-vertex dual graph of the original network graph. This distance has an upper threshold to ensure that the distant neighbors remain mutually independent. The edge-to-vertex graph is discussed in the following subsection. The threshold and the weight *w* is explained later.

Simplifying EQUATION 4.7 further with $\boldsymbol{f}$ as the state variables vector and *L* as the Graph Laplacian gives,

$$\Psi_{Reg} = \boldsymbol{f}^T L \boldsymbol{f} \qquad \text{EQUATION 4.9}$$

The graph Laplacian *L*, is defined as

$$L_{m,m'} = \delta_{m,m'} \sum_{k=1}^{M} S_{m,k} - S_{m,m'} \qquad \text{EQUATION 4.10}$$

Where the Kronecker's delta function, $\delta_{m,m'} = 1\ if\ m = m', otherwise\ 0$. The graph Laplacian is a diagonally dominant matrix, since all the non-diagonal values are less or equal to zero, because any non-diagonal value is the negative affinity value. More precisely

$$L_{m,m'} = -S_{m,m'}, \qquad for\ all\ m, m' \in M\ if\ m \neq m' \qquad \text{EQUATION 4.11}$$

And only the diagonal values are positive, i.e.

$$L_{m,m} = \sum_{k=1,k\neq m}^{M} S_{m,k} \ , \qquad for\ all\ m \in M$$

EQUATION 4.12

4.5.5 Edge-to-Vertex Dual Graph for Affinity Matrix

From the unweighted edge-to-vertex or Line Graph *LG* of the original graph *G* of the network, the desired $dist(m, m')$ values can be found using a graph traversal algorithm.

For example, let the graph *G=(V,E)* with edges *E* (road links), and vertices *V* (intersections), represents a small road network with one intersection at as shown in FIGURE 4.1.

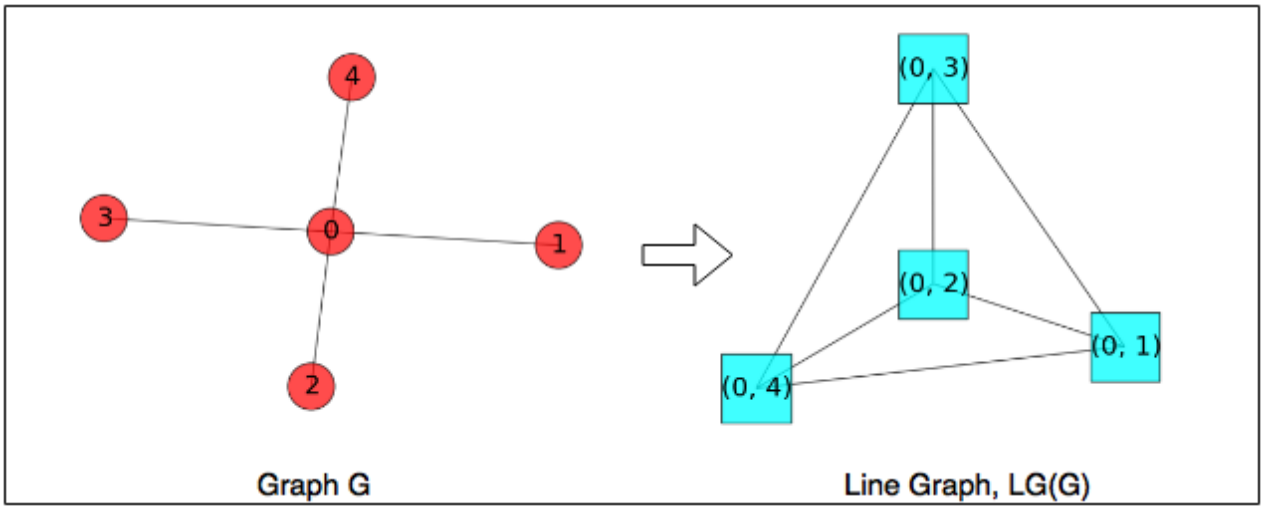

FIGURE 4.1 Primal Graph and Edge-to-Vertex Dual Graph (Line Graph) Illustration

The Line Graph $LG = (V', E')$, of the graph *G*, is constructed by adding vertices corresponding to each of the edge in graph *G*, i.e. $V' = E$, and then connecting each of the vertices in *LG* with edges to represent the adjacency between the edges in graph *G*. The adjacency in graph *G* is determined if there is a common node between the edges. Details on Line Graph can be found in Harary *(49)* and Cvetkovic *(50)*.

This dual graph is useful in this context as the unweighted distance is the number of intersections between two road links in the original network. Thus the distance matrix of this dual graph has elements $dist(m, m')$ and can be used directly to calculate the affinity matrix and the graph Laplacian for EQUATION 4.9.

4.5.6 Combining the Least Square and Regularized Terms

According to Ide and Sugiyama *(53)*, the Trajectory Regression problem formulation minimizes the objective function formed by combining the least square error with the regularization constraint by introducing the *regularization parameter* λ as in EQUATION 4.13.

$$\Psi(\boldsymbol{f}|\lambda) = || \boldsymbol{y}_N - Q^T \boldsymbol{f}||_2^2 + \lambda \boldsymbol{f}^T L \boldsymbol{f} \qquad \text{EQUATION 4.13}$$

Here, the unknown state variable vector $\boldsymbol{f}$, is found from the minimization of the objective function $\Psi(\boldsymbol{f}|\lambda)$. After taking the derivative of the objective function and setting it to zero, it takes the form of a Regularized Least Square (RLS) regression.

$$[QQ^T + \lambda L]\, \boldsymbol{f} = Q\, \boldsymbol{y}_N \qquad \text{EQUATION 4.14}$$

Once the λ is found, the suggested method for solving this equation is by using conjugate gradient method, as in the original text by Ide and Sugiyama *(53)*.

4.5.7 Finding Lambda

An optimum lambda value can be used if known a priori for solving the model, but as in most practical cases, it is not the option. Instead, it is selected by Leave-One-Out Cross Validation, which is the extreme case of k-fold cross validation where k equals the number of observations *N*. With LOO CV, the model is repeatedly re-evaluated holding out one observation at a time, and then testing for error on this held-out observation. The average LOOCV error value can be calculated for the all *N*-trajectories from real-time GPS traces efficiently using this equation which does not require repeated evaluation and prediction of the model *(57)*,

$$E_{LOOCV}(\lambda) = \frac{1}{N}\left\| \frac{(I_N - Q^T(QQ^T + \lambda L)^{-1}Q)\, \boldsymbol{y}_N}{diag_v(I_N - Q^T(QQ^T + \lambda L)^{-1}Q)} \right\|^2 \qquad \text{EQUATION 4.15}$$

Simplifying the expression with, $K = Q^T(QQ^T + \lambda L)^{-1}Q)$, average LOOCV error is

$$E_{LOOCV}(\lambda) = \frac{1}{N}\left|\left|[diag_v(I_N - K)]^{-1}(I_N - K)\, \boldsymbol{y}_N\right|\right|^2 \qquad \text{EQUATION 4.16}$$

Here, $diag_v(I_N - K)$ is the N-dimensional column vector formed by the diagonal of the matrix $I_N - K$.

Repetitive measurement of this LOOCV error expression over a range of λ helps pinpoint the minimum value of the regularization parameter. So, the best value is,

$$\lambda^* = argmin_\lambda\, E_{LOOCV}(\lambda) \qquad \text{EQUATION 4.17}$$

The suitable range for this task is discussed in a later section.

4.5.8 Prediction of Travel Time and Link Speed

It is useful to note that, the mean travel time for the *n*-th trajectory from the map-matched GPS trace can be predicted in a concise vector form once f is found based on the best regularization parameter λ^* from EQUATION 4.14. The predicted mean travel time for *n*-th trajectory is

$$\mu(x) = \widehat{y^{(n)}} = \sum_{m=1}^{M} l_m^{(n)}(\phi_m^0 + f_m) = \boldsymbol{q}^{(n)^T}(\boldsymbol{\phi^0} + \boldsymbol{f}) \qquad \text{EQUATION 4.18}$$

For N-number of trajectories from map-matched GPS traces, the *N*-number travel times can be predicted in an *N*-dimensional vector form

$$\hat{\boldsymbol{y}}_N = Q^T(\boldsymbol{\phi^0} + \boldsymbol{f}) \qquad \text{EQUATION 4.19}$$

Thus for predicting travel time using this model for any test dataset, only the link-indicator matrix Q for the test dataset has to be plugged into the EQUATION 4.19.

Similarly, for *m*-th link, the predicted speed v_m can obtained by

$$v_m = (\phi_m^0 + f_m)^{-1} \in \mathbb{R}_{\geq 0} \; for \; all \; m \in M \qquad \text{EQUATION 4.20}$$

4.6 Case Study

The implemented algorithm was applied on the test network for validation with simulated trajectories. The data composition, simulation procedure and summary statistics of the simulated data for the case study are described in this section.

4.6.1 Data Composition

In the absence of real-time crowdsourced data in this prototyping stage, the case study is based on repeatedly sampling shorter trajectories from the map-matched trajectories of the round-trip GPS traces obtained from the AVL system of UConn shuttle buses. The trajectory dataset were generated separately for all six UConn Shuttle routes at afternoon period for each of Tuesday, Wednesday and Thursday individually. The information regarding the routes and the UConn shuttles' GPS traces can be found in FIGURE 2.2 and TABLE 3.2.

This simulation process of generating synthetic trajectories mimics the behavior of the live crowdsourced data. Repeated random selection of such small trajectories generates short trips on the network at random origin and destination locations.

In reality, there might be some links which are not present in any of the trajectories for that time-segment, which is the case with usual sparse crowdsourced data. The ideal situation when every link in the network is travelled at least in the current time-period, is also generated for experimentation. This is termed as 'continuous' coverage scenario as opposed to sparse coverage. The synthetic trajectories also generate three situations where the trajectories have short, medium and long duration. Additionally, the process is repeated for three different GPS frequencies: 10, 30, 60 seconds. The description of synthesizing the dataset follows next.

Controlling GPS Trace Duration

The duration of the crowdsourced GPS traces would determine the travel time of individual map-matched trajectories. In absence of the crowdsourced GPS traces, the trajectories are synthesized. From a map-matched trajectory $c^*_{1:T}$ corresponding to a long GPS trace $g_{1:T}$ from one of the shuttle routes, where the timestamps are sequenced from 1 through T, let a randomly selected GPS trace be $g_{t:t+d}$. The corresponding trajectory $c^*_{t:t+d}$ is retrieved from $c^*_{1:T}$. Here the travel time for the synthesized trajectory stands for the GPS trace duration *d.* Here, *d* is randomly selected from a uniform distribution

$$d \sim Unif(L, U)\ ; t + d \leq T \qquad \text{EQUATION 4.21}$$

GPS trace duration *d*, which is equal to GPS trace duration, has a uniform distribution with lower and upper limits. Its upper limit, the GPS Trace Duration Max Limit *U*, is a parameter controlled while sampling to observe different scenarios. The parameter of interest *U*, is fixed at 300, 600 and 900 seconds (5, 10 and 15 minutes respectively) in three scenarios for simulation. The lower limit *L*, is fixed at 120 seconds (2 minutes) to ensure that the trajectories are realistically long enough.

The uniform distribution of *d*, is due to lack of information about the distribution of the riders on the network. With information about the travel time distribution in the network, this assumption can be updated. With repeated sampling from the map-matched long GPS traces from UConn Shuttle routes, many small GPS traces $g_{t:t+d}$, are created to mimic the map-matched trajectories from crowdsourced data. For each of the generated trajectories the starting timestamp and duration are randomly determined.

Controlling Sparsity in Network Coverage

The simulation process for synthesizing the crowdsourced GPS traces simulates both the ideal and realistic conditions. The ideal situation, continuous coverage, indicates each link is travelled at least once. On the other hand, the realistic sparse situation for crowdsourced data is when some of the links in the network have missing information. To effectively generate these situations, the consecutive GPS traces along the same shuttle route are imposed constraints.

The ideal case of continuous coverage is ensured by imposing these constraints while synthesizing trajectories

$$t' = (t + d) - kb \qquad \text{EQUATION 4.22}$$

$$b \sim Unif(0, U) \qquad \text{EQUATION 4.23}$$

Here t' is the starting timestamp for next trajectory $c^*_{t':t'+d}$, given the current trajectory is $c^*_{t:t+d}$. The *overlap duration* is *b*, randomly selected from a uniform distribution, along with the *directionality indicator* $k = 1$, ensure that the synthetic trajectories are continuous and has no gap in coverage because the next one starts at the end of the current one, or earlier. This upper bound of *b* is chosen to mimic the cases of redundancy in crowdsourced data. When randomly $b = d = U$, this is the case when more than one user start trips at the same time. On the other hand, when $b = 0$, the trajectories are simply continuous, there is no temporal-overlap between the consecutive ones.

Similarly when sparse coverage case is to be generated, setting *k = -1* effectively ensures that the trajectories are sparse, as the next selected is always after the previous trace ends on that shuttle route. So there are some links which are not on any trajectories.

The same repeated random sampling scheme described in the last two subsections is applied on all the six shuttle routes available for the afternoon period for the three days individually. Thus a number of short realistic trajectories are generated across the network with six possible combinations of coverage and trajectory length when GPS frequency is constant.

Measuring Network Coverage

For the given traces, the number of times a link is reported can be useful for understanding the network coverage intensity of the crowdsourced data. For the link indicator matrix $Q \in \mathbb{R}_{\geq 0}^{MXN}$, the number of times the link m is reported out of *M* links in the network can be calculated in a vector form. This is variable is termed as Link Count Redundancy for reporting summary statistics of the synthesized trajectory datasets. Given that there are *N*-number of GPS traces, the Link Count Redundancy value for *m*-th link is the *m*-th row in this vector $LC \in \mathbb{N}^{Mx1}$

$$LC_m = \sum_{n=1}^{N} [Q_{mn} > 0] \ , for\ 1 \leq m \leq M, 1 \leq n \leq N \qquad \text{EQUATION 4.24}$$

Here Q_{mn} is the value in the *m*-th row and *n*-th column in the link indicator matrix and the expression in Iverson bracket *(59)* is a binary indicator, where

$$[\text{condition}] = 1 \quad if\ condition\ is\ True, \quad 0\ if\ False \qquad \text{EQUATION 4.25}$$

So the sum of the incidence of non-zero lengths observed in the traces for that *m*-th link is properly accounted for in the simple vector form. For each link, this count value is 0 only if it was not observed in the GPS traces for that time-segment. An integer value greater than zero indicates that it was observed in the reported synthetic trajectories. Link Count Redundancy values greater than 2 indicate that the links have redundant data for the travel time prediction.

Similarly, a very high value tells that those links are frequently travelled. These values in the vector can be plotted on the actual road network as in FIGURE 4.5 and FIGURE 4.6, shown in a latter section for visual inspection.

To have a summary measure reflecting the overall network coverage, the visual inspection is not sufficient. For numeric comparison when network has sparse or continuous coverage, a summary statistic is required. This Average Link Count Redundancy values are thus computed with simple averaging to convey the overall network coverage information,

$$Avg\ LC\ = \frac{\sum_{m=1}^{M} LC_m}{M} \qquad \text{EQUATION 4.26}$$

This expression alone cannot capture if some links are not reported at all or if there is a gap in the coverage, it simply gives a numerical value useful for understanding the coverage intensity of the network using the GPS traces.

Summary Statistics of the Synthetic Data

The table below displays the summary statistics of the synthetic trajectory datasets.

TABLE 4.1 Summary Statistics of the Synthetic Data

Sparsity	Duration Max Limit (min)	Duration (min)			Average Number of Trajectories Covering a Link (Average Link Count Redundancy)	Number of Traces	Average Trajectory Length (m)
		Minimum	Average	Maximum			
Continuous	**5**	0.77	2.87	4.70	9.29	203.04	796.23
	10	0.71	5.20	9.62	12.73	172.83	1417.9
	15	0.74	7.34	14.43	14.49	138.16	2000.16
Sparse	**5**	0.91	2.85	4.61	1.75	119.34	791.07
	10	1.27	5.10	9.15	1.68	98.03	1407.72
	15	1.56	7.27	13.69	1.65	76.54	1995.87

In TABLE 4.1, when simulated trajectories provide continuous coverage of the entire network, the average trajectory is 5.20 minutes long, while the average maximum duration is 9.62 minutes. Similarly the average minimum duration is 0.71 minutes or almost 45 seconds. Even in its counterpart scenario, when the crowdsourced data is sparse, similar values are observed. These values are reasonable and expected, because the durations are randomly determined from a uniform distribution. The values are averaged based on 180 synthetically generated trajectory datasets. These 180 datasets are from three GPS frequency scenarios for three days and repeated 10 times for both training and testing at afternoon period.

The Average Link Count Redundancy values, which is the summary measure of how many times the links in the network were repeated in the map-matched path, appears to be higher for the scenario when all links in the network was continuously traversed by the users in the current time-segment. For example, when the GPS trace Duration for the trajectories are limited at 10 minutes, the Average Link Count Redundancy is 12.73 for continuous network coverage. On the other hand, for the sparse datasets, the value is just 1.68. This indicates that for continuous coverage, the links in the network were travelled 12.73 times on average, while for sparse trajectory datasets, the links are travelled only 1.68 times on average.

Similarly, the Number of Traces increase as the Duration Max Limit is reduced. Lowering this threshold generates shorter, more trajectories in number, while the Average Trajectory Length reduces gradually. When only short GPS traces with duration 5 minutes or below are generated, the Average Trajectory Length is close to 800 meters. On the other hand, when longer

trajectories are allowed, the length increases up to 2 kilometers. These key trends in these values indicate the characteristics of the synthetic datasets.

4.6.2 Experiment Setup

The prediction models are trained on the simulated trajectory data for afternoon period for three different days and tested on another set of simulated trajectory dataset for the same time period on that same day. In addition to testing the trained model on an another dataset in the same afternoon period, the trained model is also tested on a simulated dataset for morning and evening period to test transferability of the model to different time-periods of the day. The training and testing process is repeated 10 times for each of the six combinations of network coverage (continuous and sparse) and trajectory duration (5, 10, 15 minutes limits) conditions on a particular day, each time the simulation process randomly generates a set of trajectories. The repeated prediction tasks enable more robust observation through redundancy and reduces the chance of concluding based on outlier results.

This process is also repeated for different GPS sampling rates. The scenarios considered for this analysis are high (10 sec), medium (30 sec) and low (60 sec) GPS frequency.

4.6.3 Implementation of the Prediction Model and Simulation Process

The implementation details of the travel time prediction algorithm and the simulation of the crowdsourced systems is discussed in this section in detail. The Trajectory Regression problem requires some inputs for the specific situation, such as graph Laplacian and baseline cost estimation. So the details of the implementation and the process of synthesizing trajectories are discussed in this section together for the continuity.

Speed Information from Map-Matched GPS Traces

One input that goes into the travel time prediction model is the baseline cost for each of the links in the network. The experiment by Ide and Sugiyama *(53)* used the legal speed limits to calculate the baseline cost for the links. In the current rural context, the available historical GPS data are used to compute the inverse of the link speed to be used as the baseline cost.

In GPS traces, the time difference between two consecutive GPS points can account for both running time on a link and stopped time. As in this case, the focus is to determine feasibility of predicting travel time, the average travel speeds calculated for baseline cost account for both running and stopped time. So the baseline travel speeds from the GPS traces are lower than the running speed of the vehicles.

Using the historical map-matched GPS traces, speed information are extracted for each of the links. For a pair of consecutive timestamps $t, t+1$,the associated GPS points are g_t, g_{t+1} respectively. For these GPS points, the respective candidate points are c_t and c_{t+1}. If the map-matched path has the roads $\{r_t, r', r'', \ldots, r_{t+1}\}$ in the retrieved path from c_t to $c_{t+1}$and the lengths travelled in each of the roads are given by the sequence, $l_{t,t+1} = \{l_t, l', l'', \ldots, l_{t+1}\}$, the average speed between g_t and g_{t+1} is calculated as,

$$v_l = \frac{\sum_{l \in l_{t,t+1}} l}{timedelta(t, t+1)} \; \forall l \qquad \text{EQUATION 4.27}$$

Here, $timedelta(t, t+1)$ is the function returning the actual time difference between the timestamps.

Assuming the intersections between road links are frictionless i.e., there is no significant time lost in the intersections, each of the links in $\{r_t, r', r'', \ldots, r_{t+1}\}$ are appended with this speed information. Subsequently, these calculated speed values from the individual GPS traces are then averaged for the links to obtain baseline cost value, which is explained in the next section.

Travel time allocation to links and estimation of average speed for a link based on GPS traces remains a novel field of exploration. It has been shown that, simple harmonic mean for averaging spot speeds directly reported by GPS device tends to estimate the travel time more than the actual. This is because the sampling is based on fixed time interval, not fixed distance, unlike the traditional fixed-sensor networks used for speed measurement *(58)*. Since the GPS data used in the current experiment case does not include spot speed measurements from GPS devices, the search for a suitable method of allocating and estimating average speed from map-matched trajectories is of concern.

Nonetheless, the simple harmonic mean approach was tested for the observed data. But, since simple harmonic mean is only defined for positive real values, the observed zero-speed values in the GPS traces do not allow for such calculation without removing them, altering the link travel speed values. So finding a proper method for estimating link speed from map-matched trajectories is left for future work. Instead, a simple averaging of speeds (including zero speed values) was used when constructing the average link speeds from individual GPS traces.

Empirical observations from the available GPS traces show that GPS points can jump to a long distance in a very short time due to inherent noise in GPS data. This sometimes leads to a very high speed value, which is unreasonable. To tackle this issue, outliers above 60 miles per hour (≈30 meters per second) are assigned a value of 60 mph before calculating average speed for

each link. For the links not observed in historical GPS traces, the average link speed value from all the other links is used as the link average speed. Inverse values for the link average speeds are the baseline costs used in the implemented travel time prediction model.

Average Link Speed for Specific Time Period of Day

There are GPS traces from three weekdays in the dataset. Assuming homogeneity in the speed values for a the time-period (afternoon) across all available weekdays, average link speeds for that particular time of the day is calculated using the historical GPS traces of that time-period for all the three days of the week. For example, average links speeds for afternoon period uses all the historical GPS traces of all the three days' (Tuesday, Wednesday, Thursday) afternoon period to calculate the average travel speeds for the links.

Parameters

For the prediction model, the affinity calculation uses empirical weight value $w = 0.5$ and the upper threshold for the distance between links $dist_U = 3$, as suggested Ide and Sugiyama *(53)*. However, it remains a future task to estimate the optimum parameter values for the model on the data and network.

Construction of Graph Laplacian

The test network has 169 links, which are connected at 165 intersections. Accordingly, the NetworkX *(39)* multigraph object representing this network has 169 edges and 165 nodes. The edge-to-vertex dual graph or Line Graph of the original graph has 169 nodes representing the edges in the primal graph.

Estimating Lambda Value from LOO CV Error

Ide and Sugiyama *(53)* report optimum lambda to be 4900 for a large network of Tokyo and 600 for a small test network, accordingly the search range for the optimum value of the regularization parameter is first set below 5000. Using such narrow search range, the predicted travel times for the simulated trajectories were observed to hold a strong correlation coefficient with the actual. But as the link travel speed values were calculated from the state variables are of concern, some links were predicted to have extremely high negative or positive speed values. This instability in estimation can be attributed the insufficient weight on the regularization structure in the objective function. To handle this issue, the regularization parameter needs to be higher. So the search range for regularization parameter was expanded up to the maximum eigenvalue of the matrix QQ^T. With this empirical range, the speed values were all consistent within the reasonable speed range and the correlation coefficients were strong as well. The associated results are discussed later.

4.6.4 Performance Metrics

For the analysis and reporting of the prediction results, a number of metrics are considered. These metrics are introduced here before moving to the analysis of the results.

Pearson's Correlation Coefficient, r

The predicted travel time values and the actual ones should have a non-negative correlation. Pearson's correlation coefficient for the *N*-travel times corresponding to the same number of trajectories can be found using the equation below, where $y^{(n)}$ indicate the actual travel time n-th map-matched trajectory and $\hat{y}^{(n)}$ indicate the predicted one for the same. The bar above the variables denote their mean values corresponding to all the predicted and the actual travel times.

$$Pearson's\ r\ =\ \frac{\sum_{n=1}^{N}(y^{(n)}-\ \bar{y})\ (\hat{y}^{(n)}-\bar{\hat{y}})}{\sqrt{\sum_{n=1}^{N}(y^{(n)}-\ \bar{y})^2}\sqrt{\sum_{n=1}^{N}(\hat{y}^{(n)}-\bar{\hat{y}})^2}}$$ EQUATION 4.28

Good prediction performance is indicated by 1, while 0 meaning that there is no linear relationship between the predicted the travel time and the actual travel times.

Coefficient of Determination, R^2

This is an indicator for the ability of the prediction model to correctly explain the variability in the dataset. In other words, it is the squared value of the Pearson's correlation coefficient.

Root Mean Squared Error (RMSE)

The RMSE is used to find the average deviation of the estimation from a given actual value in the same unit. In this case, the RMSE are all reported in seconds. A lower RMSE value indicates that the estimated values are likely to be closer to the actual, so values closer to 0 are better.

$$RMSE\ =\ \sqrt{\frac{\sum_{n=1}^{N}(y^{(n)}\ -\hat{y}^{(n)})^2}{N}}$$ EQUATION 4.29

Mean Absolute Error or Mean Absolute Deviation (MAD)

It indicates the average magnitude of the difference of the predicted value from the actual. This measure weighs each of observations equally.

$$MAD = \frac{\sum_{n=1}^{N} \left| y^{(n)} - \hat{y}^{(n)} \right|}{N}$$ EQUATION 4.30

The difference between RMSE and MAD values indicate that some errors are not uniform for all observations. Otherwise, MAD should be equal to RMSE.

Mean Absolute Percentage Error (MAPE)

MAPE is similar to MAD, but it calculates the deviation measure on the basis of the actual values. This measure is useful for comparing the prediction performance when actual values have different magnitude and scale in different datasets.

$$MAPE = \frac{\sum_{n=1}^{N} \left| \frac{y^{(n)} - \hat{y}^{(n)}}{y^{(n)}} \right|}{N} x100\%$$ EQUATION 4.31

4.7 Results

This part introduces the metrics used for evaluation of the performance explains the results observed in the case study using the simulation experiment.

The Prediction results are analyzed and presented with an interest in understanding the sensitivity of the parameters involved with the prediction process. For this purpose, first, the results are explored keeping the GPS frequency value constant at 10 seconds and finally the impact of GPS frequency is presented in the last part.

4.7.1 Effect of Sparsity and GPS Trace Duration

TABLE 4.2 summarizes the impact of duration of the trajectories calculated from the randomly selected GPS traces and sparsity on travel time prediction performance when GPS frequency is

10 seconds. From the table, the effect of GPS Trace Duration is clearly visible from the Correlation Coefficient and Coefficient of Determination. It is obvious that when trajectories are allowed to have longer duration, the association between the predicted and actual travel times become stronger. Simply looking at the Pearson's *r* value for *continuous* coverage condition, the values increase from 0.85 to 0.96 and then 0.98 for prediction on the training dataset. Similarly, the values increase for the prediction tasks on test dataset, using the trained models, but the values are a little weaker for testing, which is expected. For any GPS Trace Duration, the correlation coefficients are better for sparsely covered network, both for training and testing.

TABLE 4.2 Mean Performance Metrics for Real-Time Scenario

Sparsity	GPS Trace Duration Max Limit (min)	Pearson's r		R-squared		RMSE (sec)		MAD (sec)		MAPE (%)	
		Train	Test	Train	Test	Train	Test	Train	Test	Train	Test
Continuous	5	0.85	0.82	0.72	0.68	33.69	37.51	25.94	29.39	14.33	16.36
	10	0.96	0.95	0.93	0.91	38.98	45.40	30.16	35.33	11.47	13.34
	15	0.98	0.98	0.97	0.96	43.18	50.05	32.98	38.77	10.24	11.94
Sparse	5	0.89	0.78	0.79	0.61	30.54	43.34	23.10	34.24	13.82	18.51
	10	0.96	0.93	0.92	0.87	38.68	57.90	29.20	44.60	11.40	15.86
	15	0.98	0.97	0.97	0.93	38.55	63.54	30.30	48.65	9.73	13.58

Note: Value in each cell is averaged over 30 travel time prediction tasks (10timesX3days) on simulated train/test dataset at GPS frequency =10 seconds

From the table, R^2 values are a little better for the *sparse* scenario at GPS Trace Duration Max Limit at 5 minutes for training than for the Continuous. While it works well for short trips (0.79 versus 0.72), the average Coefficient of Determination values for *continuous* scenario is not much different for medium to long GPS trace duration (0.93 versus 0.92 and both 0.97).

While the *sparse* trajectory datasets have better average predictive ability for known training set, it performs rather poorly in *continuous* scenario (0.68, 0.91, 0.96 versus 0.61, 0.87, 0.93 respectively) for the test dataset which is likely to contain unseen trajectories. This indicates that when the network has *sparse* coverage, the regularization parameter assigns more weight on the network structure, which stabilizes the predictions for unseen links at the expense of overfitting the model. But realistically, the *sparse* coverage is the more expected for crowdsourced data and *continuous* coverage is only the ideal scenario. Establishing the proper search range for the regularization parameter or improving the affinity function may improve the predictive ability on unknown trajectories in real-world sparse crowdsourced trajectories.

In summary, the correlation between the predicted and the actual travel times for the randomly simulated trajectories, seem to be quite strong. From the Coefficient of Determination values, the models appear to be more responsive to the variation in the travel time data with increased trajectory duration. As they become long enough (GPS Trace Duration Max Limit=15min), 97% of the variability in the data is captured by the model. While the original proponent of Trajectory Regression did compare performances for network size and model suitability, this kind of analysis on real-world applicability is not explored and thus few comparative results can be discussed in this discussion.

Another family of performance metrics, RMSE and MAD indicate that the relative errors in the prediction gradually increase with the GPS Trace Duration Max Limit which is apparently counter-intuitive. But the RMSE values from TABLE 4.2 plotted in FIGURE 4.2 helps explain the reason.

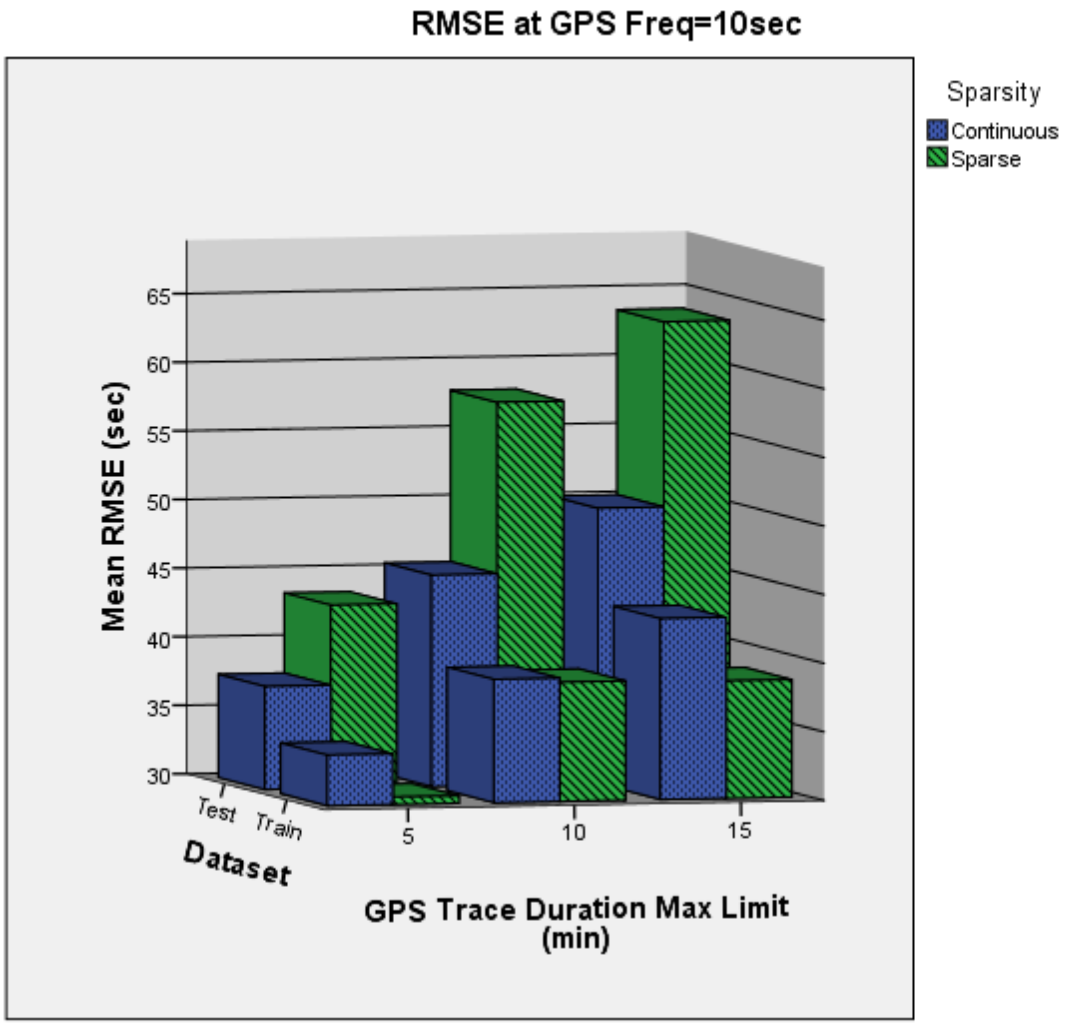

FIGURE 4.2 Effect of Sparsity and GPS Trace Duration on RMSE Values

From FIGURE 4.2, the RMSE values increase with the GPS Trace Duration Max Limit, for prediction on both train and test dataset. Prediction on training set has lower RMSE value as expected. As test data has more error and is more reflective of real-world prediction on unknown traces, it deserves more focus than the train case here. So, for *continuous* coverage of the network, the average errors on test datasets increases from 38 seconds when GPS Duration Max Limit is 5min(=300sec), to 50 seconds when GPS Duration Max Limit is 15min(=900sec). So the increase of errors is proportionally diminishing as the GPS traces become longer. For the

same two short and long GPS traces, the RMSE values for *sparse* coverage are 44 and 64 seconds. From this RMSE plot, the over-fitting in *sparse* scenario is clearly visible. In this scenario, the error values are lower for training and disproportionately high for testing. On the other hand, *continuous* coverage of network helps train a balanced model which is comparatively robust for a new test trajectory.

Similar to RMSE, MAD values are also not scaled for the magnitude of the actual travel times, so the MAD values cannot be used to understand how GPS trace duration affects performance but it can be juxtaposed with RMSE values to see if the errors are uniform across travel times. In TABLE 4.2, average MAD values for prediction on test set ranges from 30 to 39 seconds for *continuous* coverage and it ranges between 34 to 49 seconds for *sparse*. But noticeably, in every case, the average MAD values are lower than average RMSE values. This indicates, travel time prediction have non-uniform errors, which is not a surprise.

Considering the increase of actual travel time based on GPS Trace Duration Max Limit, the average magnitudes of errors in terms of RMSE and MAD are not sufficient to show the complete picture. Rather, the scaled MAPE values in FIGURE 4.3 are more appropriate in understanding the effect of the sparsity and GPS trace duration.

Unlike the RMSE plot which showed the average errors increase with GPS trace duration, the scaled MAPE values indicate the opposite; the errors are relatively less when GPS traces are longer. In association with FIGURE 4.2, this MAPE plot tells that the longer GPS traces help minimize the prediction errors.

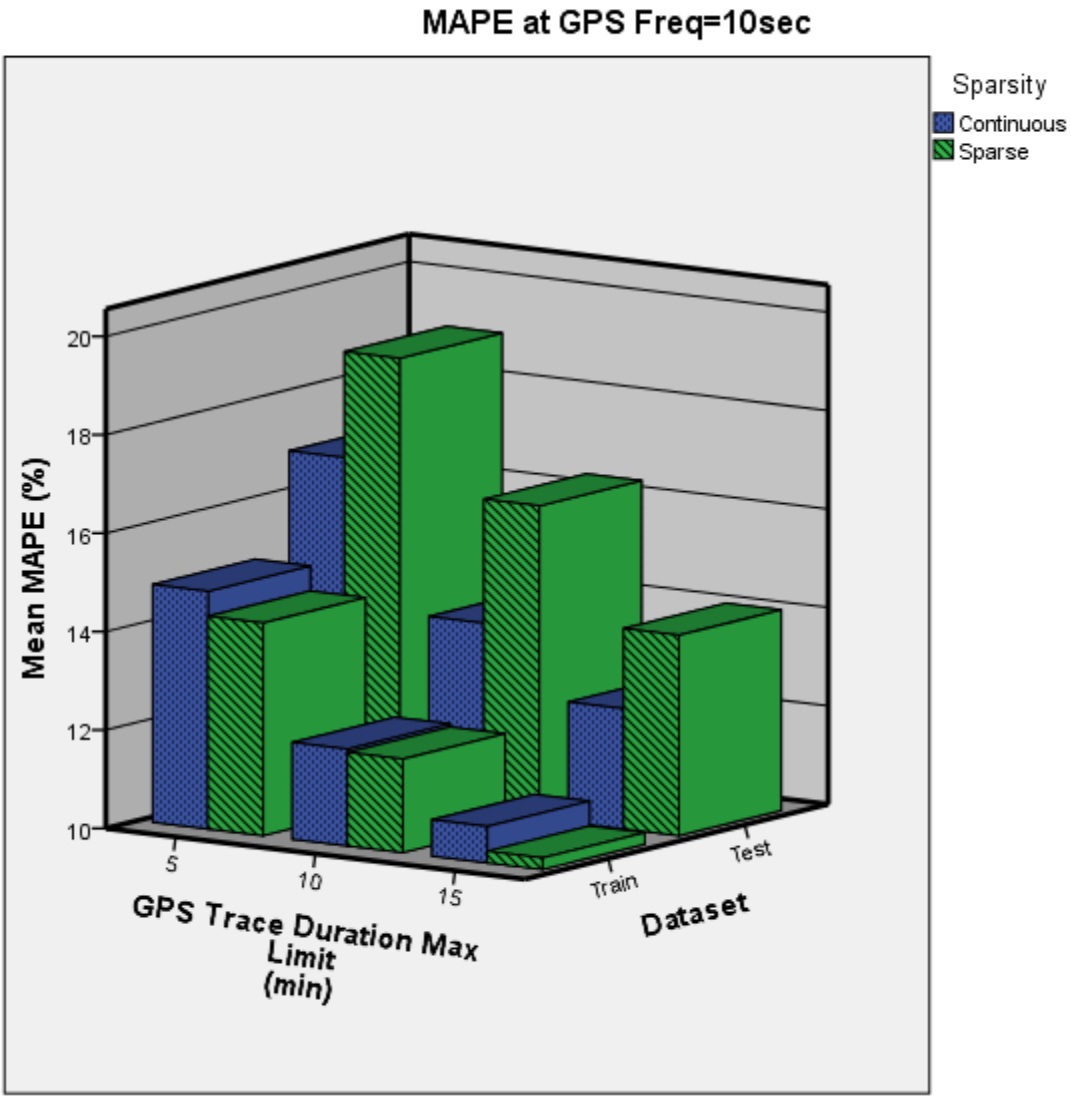

FIGURE 4.3 Effect of Sparsity and GPS Trace Duration on MAPE Values

The MAPE values show that when short GPS traces are generated, average percentage prediction error for the *sparse* datasets is less than that of *continuous* for training and it deteriorates for testing, indicating over-fit once again. On the other hand, when road network has *continuous* coverage, the errors are relatively balanced for testing. For this case, the average prediction for testing has average absolute percentage error of 16.36%, 13.34% and 11.94% for

GPS Trace Duration Max Limit 5, 10 and 15 minutes, respectively. For the same prediction on testing data, these average values are 18.51%, 15.86% and 13.58% in the same order.

So looking at the values, which are straight from TABLE 4.2, the conclusion is clear: longer GPS traces are more conducive to lowering the prediction accuracy and greater network coverage with crowdsourced data results in more stable prediction.

4.7.2 Disaggregate View of the Travel Time Prediction

The results presented so far, captures the overall aggregate impact of sparsity and duration of trajectories on performance. Since GPS Trace Duration Max Limit varies between 5 to 15 minutes, the comparison between them is not straight-forward. To understand the true performance, a single instance is considered in FIGURE 4.4. In this figure, the prediction tasks are shown for Wednesday afternoon period, with GPS frequency 10 seconds. The left column is for the scenario when network coverage is *continuous* and the right column is for *sparse*. The rows correspond to three GPS Trace Duration Max Limit scenarios: 5, 10 and 15 minutes from top to bottom. The matrix of the scatterplot thus holds 6 different scatter plots of actual travel time from simulated test datasets. The scatterplots have the actual time on x-axis and the predicted time on y-axis. All the scatterplots are plotted using the same scale. For the same time period there are 9 other similar prediction tasks for each of the six sparsity and network coverage scenarios.

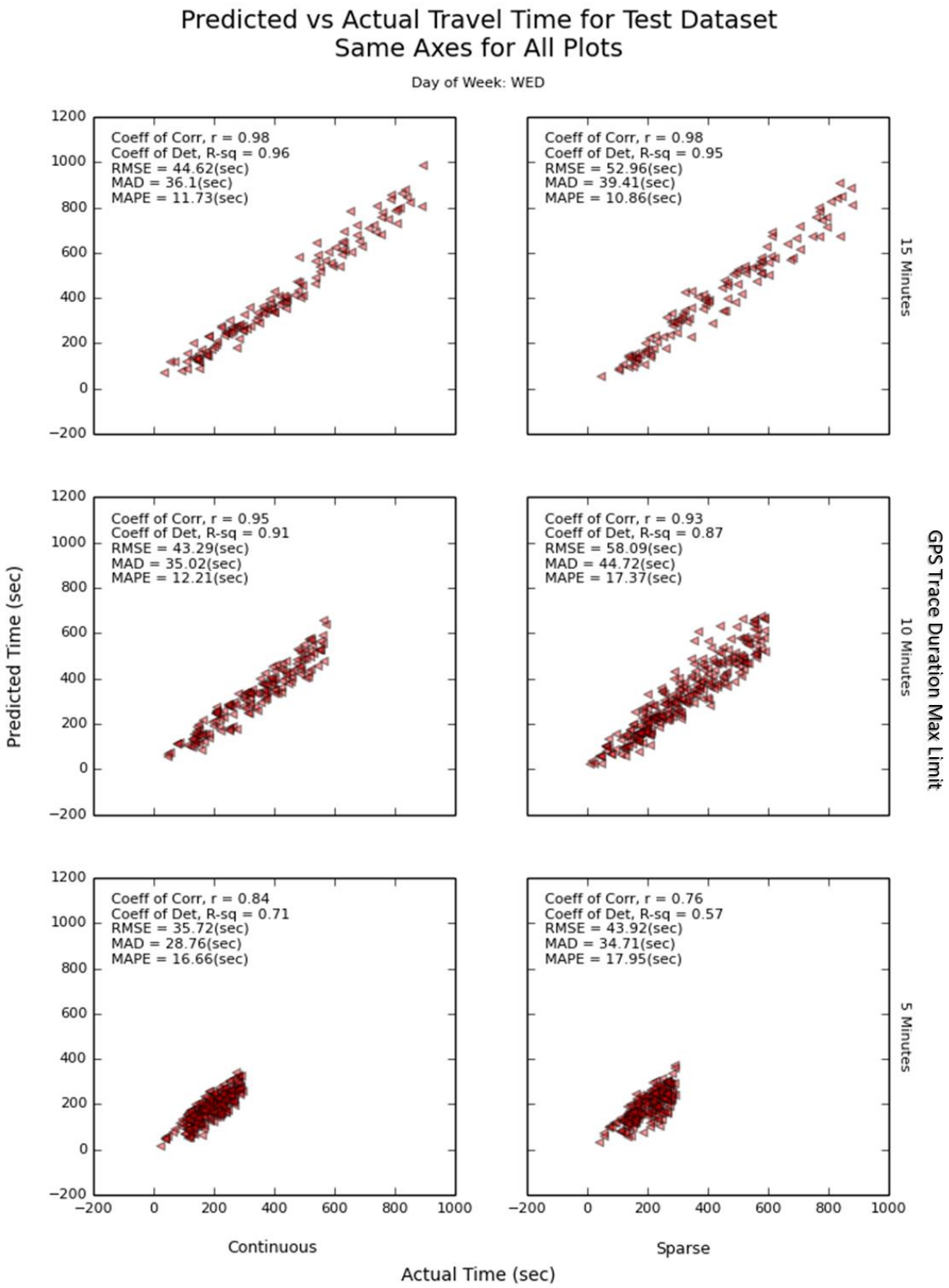

FIGURE 4.4 Disaggregate analysis of prediction performance

From FIGURE 4.4, it shows that correlation coefficients are higher for *continuous* case for any GPS Trace Duration Max Limit, similarly coefficient of determination values also follow same trend, since it is simply the squared value of the correlation coefficient. The RMSE and MAD values here cannot be used to infer trends from this random instance of the prediction process. Rather the scaled MAPE values are more reliable as they do not depend upon the range of the actual travel times. For any row, the lower MAPE values for *continuous* trajectory dataset indicate lesser error relative to actual travel time. Also the MAPE values increase from the bottom to the top row. The trend supports the conclusion from the aggregate observations made in the previous subsection.

The range of the travel time values in the scatterplot offers visual evidence why unscaled RMSE and MAD values are inappropriate in this case. It also explains the low correlation coefficient in the bottom row (short GPS traces) as seen in to the dense cloud within a narrow range. It also depicts the wide actual travel time range when GPS trace duration increases (10 and 15minutes) and the associated strong correlation coefficient.

4.7.3 Link Speed Prediction from Trajectory Regression

In FIGURE 4.5 and FIGURE 4.6, the predicted travel speeds for all the links are shown with a color-map in the top rows of both figures. Accordingly, the (Link Count Redundancy-1) value for each of the links is also shown in similar fashion in the bottom rows of the figures. The Link Count Redundancy plots show the network coverage intensity from the GPS traces. Out of many other similar plots, only a pair of instances for *continuous* and *sparse* trajectory datasets is shown here. The unobserved links in the *sparse* case is shown with dark blue shade. The shade of the red increase with the number of times the link was traveled in the dataset.

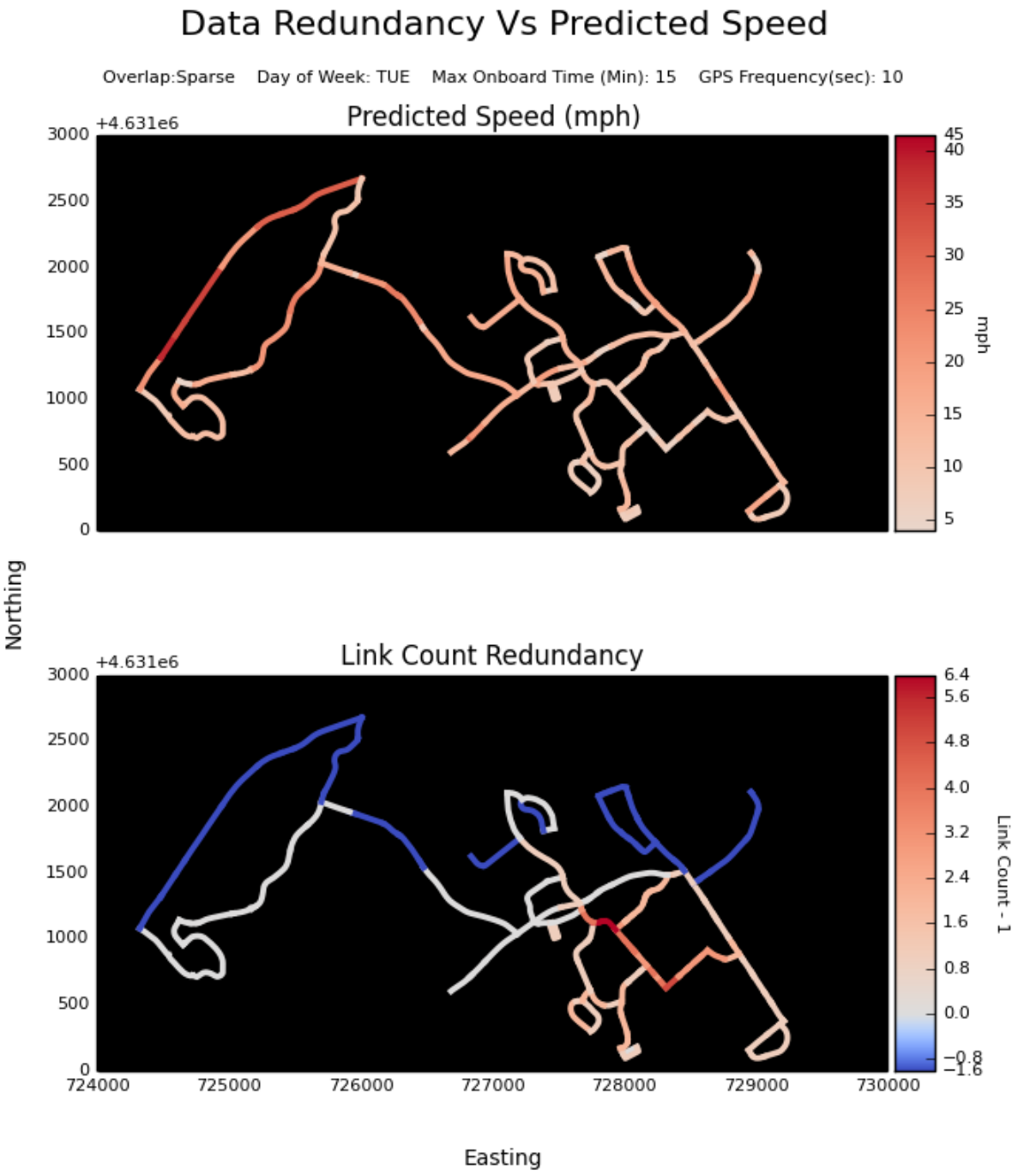

FIGURE 4.5 Link Travel Speed Prediction - Sparse

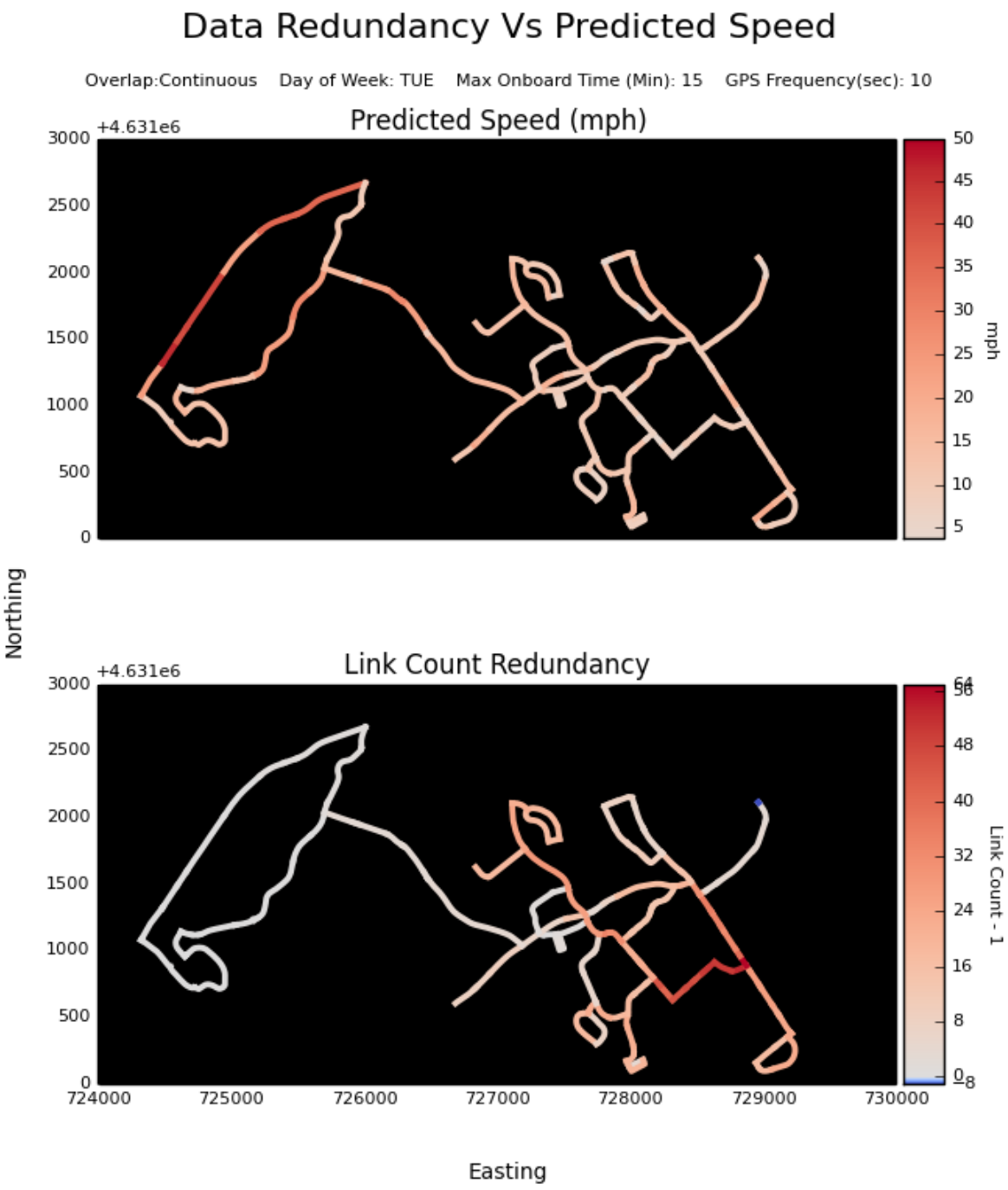

FIGURE 4.6 Link Travel Speed Prediction - Continuous

From the plots of the predicted speeds, the darker shades show that those links have a high predicted travel speed value for the current time-segment. In FIGURE 4.5, it is noteworthy how there are many road links which do not have any GPS information for the time-segment, but has been annotated with the predicted travel speed values from Trajectory Regression algorithm, using EQUATION 4.20.

Speaking of sparsity in network coverage, the range of Link Count Redundancy varies significantly for *continuous* from *sparse*, as also reported in TABLE 4.2. The bottom row of FIGURE 4.5 has a range up to 7 while the continuously covered network in FIGURE 4.6, has a range up to 65. This means, for this *sparse* case, the links are traveled maximum 7 times, while for *continuous*, the links are visited as frequently as 65 times in the synthetic trajectory dataset. Qualitative comparison of the predicted speed plots in the top rows in the FIGURE 4.5 and 4.6 shows that the spatial smoothing from the affinity values actually works well to predict the link speeds even for the links with no current updates. The darker shades tell which roads are predicted to have higher speed for the current time-segment. The predicted speed range is quite similar in both network coverage conditions, showing stability of the algorithm.

4.7.4 Effect of GPS Frequency on Prediction and Testing on Different Time Periods

From TABLE 4.3 the effect of GPS Frequency on travel time prediction performance is analyzed keeping GPS Trace Duration Max Limit fixed at 5 min. First, the focus is on the travel time prediction performance on the testing dataset. Using the average value of the useful metrics from the table, it is seen that the average R^2 values increases as GPS frequency is lowered (60sec). But average RMSE values decrease accordingly. On the other hand, the MAPE values indicate that the average relative errors actually worsen when GPS frequencies are reduced (60 sec). One plausible reason could be the small network used in the map-matching process. Even

with the small network, the lower frequency does affect the map-matching performance. With such erroneous map-matched trajectories, the speed calculation and true paths may have errors and can affect the travel time prediction accuracy. In conclusion, lower GPS frequency is expected to affect the travel time prediction performance. But the effect appears to occur in a complex manner and requires more exploration.

TABLE 4.3 Effect of GPS Frequency and Testing on Different Time Periods

Metrics	GPS Freq	TOD	Train	Test
R^2	**10**	mo		0.82
		af	0.75	0.64
		ev		0.84
	30	mo		0.82
		af	0.76	0.67
		ev		0.83
	60	mo		0.82
		af	0.79	0.70
		ev		0.82
RMSE (sec)	**10**	mo		116.88
		af	32.11	40.83
		ev		103.22
	30	mo		118.60
		af	31.47	38.79
		ev		107.62
	60	mo		119.47
		af	29.84	36.76
		ev		113.22
MAPE (sec)	**10**	mo		18.83
		af	14.08	17.44
		ev		17.06
	30	mo		19.28
		af	15.92	18.56
		ev		17.78
	60	mo		19.06
		af	18.12	21.85
		ev		18.72
Note: Duration Max Limit kept constant at 5min				

To see how the performance varies when the model trained on afternoon time-period is used for predicting travel time in morning and evening periods, the same metrics in the rightmost column in TABLE 4.3 are useful. From the R^2 values it appears the model flatly explains almost 80% of the variability in the data, but is not flexible and responsive to the new dataset. The same

flatness is also observed in the RMSE values which tells that the predicted travel time is off by almost 2 minutes when GPS Trace Duration Max Limit is only 5 minutes (Average Observed GPS Trace Duration = 2.87 minutes). Thus the error in prediction is very high. As also the MAPE values indicate around 20% average error for such prediction on different time-period.

But in all three metrics, the model trained on afternoon period appears to perform slightly less worse for evening period than in the morning. This is plausibly because afternoon and evening time periods are closer in time so the baseline speeds are also more alike.

4.8 Conclusions, Limitations and Future Research on Trajectory Regression

In conclusion, the Regularized Least Square regression based algorithm in the Trajectory Regression problem can be used in tandem with the HMM-based map-matching algorithm for real-time prediction of travel time and link speeds for the crowd-sourced real-time traveler information system. The application of real-world data and performance analysis hold promises for applicability of the approach in a broader level.

1. This implementation of the algorithm demonstrates that travel time prediction can be reliably performed circumventing the sparsity issue associated with crowd-sourced real-time GPS traces. Long average trips on the network can generate GPS traces with long duration. So the network with long trips has a better travel time prediction. The experiment confirms that the more links are reported in the crowdsourced GPS traces, the better would be the predictive ability. GPS frequency affects the travel time prediction, but the effect is not quite clear.
2. Travel time prediction can be done quite reliably for a trip with origin and destination located anywhere within the network. The link speed prediction for the entire network can be predicted as well even though the real-time GPS data is scarce and sparse.

Similarly it can predict travel time between transit stops as well. Similarly time of arrival can also be predicted for both transit and non-transit modes.

But the experiment has the following limitations as well

1. This work uses historical GPS traces from shuttle buses which are available at the moment. Due to lack of real-time in-streaming GPS traces from a range of GPS sources, as should be the real crowdsourced system, this simulation process is limited in scope.
2. The extraction of travel speed between consecutive GPS points is solely based on the map-matched GPS traces and has observed unreasonable values in a few cases which were masked using a realistic speed value limit. So the link speed estimation is not completely automated and fool-proof. It also lacks sufficient theoretical foundation.
3. The affinity values used in the Laplacian matrix uses empirical parameter values. There is limited theoretical background on the choice of these values.
4. The regularization parameter found over a range of values is not theoretically established. A complete guideline to establish the search range depending on the network characteristics is needed to avoid manual tuning.

Even with the limitations, the implemented travel time prediction algorithm has a wide prospect of future research.

1. Though results appear quite promising, applying the model on a larger and denser network using real-time streaming GPS traces may provide more insight on how the crowdsourced system would behave in reality.
2. Exploring suitable probabilistic approaches for allocating speeds calculated from GPS traces to the appropriate links would be interesting. Also finding a suitable approach for estimating the average link speeds from the map-matched GPS traces needs explorations.

Collecting spot speed information from GPS system can offer a basis for comparison with the current simple approach of averaging.

3. The parameter values in affinity function estimated through a theoretical formulation can introduce more credibility to the prediction framework. Also the function can be tested for various forms to find the best one.
4. Finding the optimum distribution of the link speed and function expressing the distribution can be incorporated into the model to make it predict ahead of time, instead of just in time. It can also employ methods to tackle with temporal sparsity in crowdsourced data as in Zheng and Ni (60).

CHAPTER 5 CONCLUSIONS AND FUTURE RESEARCH

5.1 Summary

The thesis focuses on exploring the feasibility and developing a prototype of a crowdsourced Real-Time Traveler Information System (RTIS) for rural context. To develop the prototype system, GPS data from smartphones of the participants of the system has been selected as the suitable source of real-time traveler information updates. It outlines the system in Chapter 2, where sketch of the system is introduced. In Chapter 3, it implements a robust map-matching algorithm to associate GPS traces to the underlying road network and shows that the algorithm performs quite well under the test conditions. Subsequently in Chapter 4, it discusses a travel time prediction algorithm implemented for the prototype. The algorithm is particularly suitable for crowdsourced system as it overcomes the sparsity issue of crowdsourced data. The demonstration of the travel time prediction algorithm in different simulated contexts shows the impact of the network condition and level of participation on the travel time prediction quality. All these implementations are consistent along the conceptual framework in Chapter 2.

5.2 Future Scalability

While the initial tests on the prototype system is restricted by data, scale and infrastructure, the development keeps scalability in mind. Even though the implemented framework is tested on GPS data from UConn shuttle, it is readily applicable for non-transit modes as well.

The implemented system is scalable because it is a) not restricted to a particular mode, b) capable of predicting the travel condition across the network in real-time even when real-time traveler information is scarce and sparse c) flexible enough to allow trips with origin and destination at any point on the network.

For part a, it assumes that the true path of the crowdsourced GPS traces is not known in advance and then it finds likely actual path using the map-matching algorithm in Chapter 3. Thus it does not restrict itself within transit realm, where the true paths are predefined.

For part b and c, it employs a recent approach titled Trajectory Regression to predict the link speed conditions for the network in real-time.

For part c, it allows travel time prediction for any random pair of origin and destination in the network, which allows prediction of the travel time between transit stops-to-stop which is one specific application of this approach. But it also retains the possibility of predicting travel time for other modes which do not rely on specific start-stop locations.

Thus this implemented mechanism is quite liberal and generalizable for transit and non-transit case.

5.3 Future Research

This section discusses future direction of the overall development of the backend of the crowdsourced real-time traveler information system in addition to the suggested improvements of the algorithm implementations mentioned separately in the concluding sections in Chapter 3 and 4.

In future, a pilot project on a large network can be useful in understanding possible issues. Interfacing the implemented algorithm with the smartphone application would be the first step. Then integrating and deploying the algorithms in the RETTINA backend system, by connecting the two implemented algorithms, would enable the map-matching and travel condition

prediction to function in synchronization. The codebase of the implemented algorithms is to be released as an open-source project in future.

REFERENCES

1. FHWA. http://www.ops.fhwa.dot.gov/tdm/, Accessed July 17, 2015

2. Boulos, M. N. K., B. Resch, D. N. Crowley, J. G. Breslin, G. Sohn, R. Burtner, W. A. Pike, E. Jezierski, and K. S. Chuang. Crowdsourcing, Citizen Sensing and Sensor Web Technologies for Public and Environmental Health Surveillance and Crisis Management: Trends, OGC Standards and Application Examples. *International Journal of Health Geographics*, Vol. 10, No. 1, 2011, pp. 67.

3. Kandarpa, R., J. Sangillo, L. Burgess & A. Toppen (2010). *Real-Time Traveler Information Market Assessment White Paper* . Publication FHWA-JPO-10-055, U.S. Department of Transportation, 2010.

4. Waze. https://www.waze.com/., Accessed June 16, 2015.

5. INRIX. http://www.inrix.com., Accessed June 16, 2015.

6. NAVTEQ. http://www.navteq.com/., Accessed June 16, 2015.

7. . Zeimpekis, V., G. M. Giaglis, and G. Lekakos. *A Taxonomy of Indoor and Outdoor Positioning Techniques for Mobile Location Services*. *SIGecom Exch.*, Vol. 3, No. 4, 2002, pp. 19-27.

8. Watzdorf, S. V., and F. Michahelles. Accuracy of Positioning Data on Smartphones. In *Proceedings of 3rd International Workshop on Location and the Web*, ACM, 2010.

9. Zandbergen, P. A., and S. J. Barbeau. Positional Accuracy of Assisted GPS Data from High-Sensitivity GPS-Enabled Mobile Phones. *Journal of Navigation*, Vol. 64, No. 03, 2011, pp. 381.

10. GPS Logger for Android. https://github.com/mendhak/gpslogger, Accessed June 16, 2015.

11. Thiagarajan, A. Probabilistic Models for Mobile Phone Trajectory Estimation. Phd Dissertation, Electrical Engineering and Computer Science, Massachusetts Institute of

Technology, 2011.

12. Rahmani, M. Path Inference of Sparse GPS Probes for Urban Networks: Methods and Applications (Licentiate Thesis), Department of Transport Science, KTH,Stockholm, Sweden. , 2012.

13. Quddus, M. A., 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, No. 5, 2007, pp. 312-328.

14. White, C. E., D. Bernstein, and A. L. Kornhauser. Some Map Matching Algorithms for Personal Navigation Assistants. *Transportation Research. Part C: Emerging Technologies*, Vol. 8, No. 1-6, 2000, pp. 91-108.

15. Davidson, P., J. Collin, and J. Takala. Application of Particle Filters to Map-Matching Algorithm. *Gyroscopy and Navigation*, Vol. 2, No. 4, 2011, pp. 292.

16. . Quddus, M. A., R. B. Noland, and W. Y. Ochieng. A High Accuracy Fuzzy Logic Based Map Matching Algorithm for Road Transport. *Journal of Intelligent Transportation Systems*, Vol. 10, No. 3, 2006, pp. 103-115.

17. Velaga, N. R., J. D. Nelson, P. Edwards, D. Corsar, S. Srip, N. Sharma, and M. Beecroft. Development of a Map-Matching Algorithm for Rural Passenger Information Systems through Mobile Phones and Crowd Sourcing. *Journal of Computing in Civil Engineering*, Vol. 27, 2012.

18. Najjar, M. E. E., and P. Bonnifait. A Road-Matching Method for Precise Vehicle Localization using Belief Theory and Kalman Filtering. In *Autonomous Robots Archive*, Vol. 9, No. 2, 2005, pp. 173 – 191.

19. Chen, B. Y., H. Yuan, Q. Li, W. H. K. Lam, S. Shaw, and K. Yan. Map-Matching Algorithm for Large-Scale Low-Frequency Floating Car Data. *International Journal of Geographical*

Information Science, Vol. 28, No. 1, 2014, pp. 22-38.

20. Marchal, F., J. K. Hackney, and K. W. Axhausen. Efficient Map Matching of Large Global Positioning System Data Sets. *Transportation Research Record: Journal of the Transportation Research Board 1935*, 2005, pp. 93-100.

21. Newson, P., and J. Krumm. Hidden Markov Map Matching Through Noise and Sparseness. In *17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, November 2009, pp. 343.

22. Feng, T., and J. P. Timmermans. Map Matching of GPS Data with Bayesian Belief Networks. In *Eastern Asia Society for Transportation Studies*, 2013.

23. Brakatsoulas, S., D. Pfoser, R. Salas, and C. Wenk. On Map-Matching Vehicle Tracking Data. In *Proceedings of the 31st VLDB Conference*, Seattle, WA, 2005, pp. 864.

24. Hummel, B. *Dynamic and Mobile GIS: Investigating Changes in Space and Time*, Chapter Map Matching for Vehicle Guidance. CRC Press, 2006.

25. Lou, Y., C. Zhang, Y. Zheng, X. Xie, W. Wang, and Y. Huang. Map-Matching for Low-Sampling-Rate GPS Trajectories. In *Proceedings of 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, ACM, New York, 2009, pp. 361.

26. Yuan, J., Y. Zheng, C. Zhang, X. Xie, and G. Sun. An Interactive-Voting Based Map Matching Algorithm. In *Proceedings of the 2010 Eleventh International Conference on Mobile Data Management*, May 23, 2010, pp. 52.

27. C.Y. Goh, J. Dauwels, N. Mitrovic, M. T. Asif, A. Oran, and P. Jaillet. Online Map-Matching Based on Hidden Markov Model for Real-Time Traffic Sensing Applications. In *2012 15th International IEEE Conference*, September 16, 2012, pp. 781.

28. Osogami, T., and R. Raymond. Map Matching with Inverse Reinforcement Learning.

Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, 2013, pp. 2547-2553.

29. Bierlaire, M., J. Newman, and J. Chen. A Probabilistic Map Matching Method for Smartphone GPS Data. *Transportation Research Part C*, Vol. 26, 2013, pp. 78–98.

30. Oran, A., and P. Jaillet. A Precise Proximity-Weight Formulation for Map. In *10th Workshop on Digital Object Identifier Positioning Navigation and Communication (WPNC)*, 2013, pp. 6.

31. Oran, A., and P. Jaillet. An HMM-Based Map-Matching Method with Cumulative Proximity-Weight Formulations. In *2nd Annual International Conference on Connected Vehicles & Expo*, December 2013, pp. 480-485.

32. NAVSTAR Global Positioning System Surveying. *US Army Corps of Engineers*, No. 1110-1-1003, 2011, pp. 4–1 – 4–18.

33. Javanmard, A., M. Haridasan, and L. Zhang. Multi-Track Map Matching. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems, ACM*, 2012, pp. 394-397.

34. Hunter, T., 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, No. 2, 2014, pp. 529.

35. Rabiner, L. R. A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, Vol. 77, No. 2, 1989, pp. 286.

36. Murphy, K. P. *Machine Learning: A Probabilistic Perspective.* MIT press, 2012.

37. van Diggelen, F. GNSS Accuracy: Lies, Damn Lies and, Statistics. *GPS World*, 2007, pp. 32.

38. Map and Geographic Information Center (MAGIC). http://magic.lib.uconn.edu/connecticut_data.html#roads, Accessed June 28, 2015.

39. Hagberg, A. A., D. A. Schult, and P. J. Swart. Exploring Network Structure, Dynamics, and Function using NetworkX. In *Proceedings of the 7th Python in Science Conference (SciPy2008)*, Pasadena, CA, USA, August 2008, pp. 11–15.

40. Fawcett, T. An Introduction to ROC Analysis. *Pattern Recognition Letters*, Vol. 27, No. 8, 2006, pp. 861-874.

41. Lin, W. H., and J. Zeng. Experimental Study of Real-Time Bus Arrival Time Prediction with GPS Data. *Transportation Research Record: Journal of the Transportation Research Board*, Vol. 1666, 1999, pp. 109.

42. Sun, D., H. Luo, L. Fu, W. Liu, X. Liao, and M. Zhao. Predicting Bus Arrival Time on the Basis of Global Positioning System Data. *Transportation Research Record: Journal of the Transportation Research Board*, Vol. 2034, 2007, pp. 62–72.

43. Shalaby, A., and A. Farhan. Prediction Model of Bus Arrival and Departure Times using AVL and APC Data. *Journal of Public Transportation*, Vol. 7, No. 1, 2004, pp. 41-61.

44. Bin, Y., Y. Zhongzhen, and Y. Baozhen. Bus Arrival Time Prediction using Support Vector Machines. *Journal of Intelligent Transportation Systems: Technology, Planning, and Operations*, Vol. 10, No. 4, 2006, pp. 151-158.

45. Lin, Y., X. Yang, N. Zou, and L. Jia. Real-Time Bus Arrival Time Prediction: Case Study for Jinan, China. *Journal of Transportation Engineering*, 2013.

46. General Transit Feed Specification (GTFS). https://developers.google.com/transit/gtfs/, Accessed June 30, 2015.

47. Zimmerman, J., A. Tomasic, C. Garrod, D. Yoo, C. Hiruncharoenvate, R. Aziz, N. R. Thiruvengadam, H. Huang, and A. Steinfeld. Field Trial of Tiramisu: Crowd-Sourcing Bus Arrival Times to Spur Co-Design. In *Proceedings of the SIGCHI Conference on Human Factors*

in Computing Systems, ACM, 2011.

48. Tao, S., V. Manolopoulos, S. Rodriguez, and A. Rusu. Real-Time Urban Traffic State Estimation with A-GPS Mobile Phones as Probes. *Journal of Transportation Technologies*, Vol. 2, No. 01, 2012, pp. 22.

49. Karbassi, A., and M. Barth. Vehicle Route Prediction and Time of Arrival Estimation Techniques for Improved Transportation System Management. In *Proceedings of IEEE Intelligent Vehicles Symposium, 2003*, IEEE, Columbus, OH, USA, 2003, pp. 511-516.

50. Jenelius, E., and H. N. Koutsopoulos. Travel Time Estimation for Urban Road Networks using Low Frequency Probe Vehicle Data. *Transportation Research Part B: Methodological*, Vol. 53, 2013, pp. 64-81.

51. Yang, B., M. Kaul, and C. S. Jensen. Using Incomplete Information for Complete Weight Annotation of Road Networks. *IEEE Transactions on Knowledge and Data Engineering*, Vol. 26, No. 5, 2014, pp. 1279.

52. Tulic, M., D. Bauer, and W. Scherrer. Link and Route Travel Time Prediction Including the Corresponding Reliability in an Urban Network Based on Taxi Floating Car Data. *Transportation Research Record: Journal of the Transportation Research Board*, No. 2442, 2014, pp. 140-149.

53. Ide, T., and M. Sugiyama. Trajectory Regression on Road Networks. In *Proceedings of Association for the Advancement of Artificial Intelligence (AAAI) Conference on Artificial Intelligence*, 2011.

54. Ide, T., and S. Kato. Travel-Time Prediction using Gaussian Process Regression: A Trajectory-Based Approach. In *Proceedings of the 2009 SIAM International Conference on Data Mining (SDM)*, SIAM, 2009, pp. 1185-1196.

55. Harary, F. *Graph Theory.* Addison-Wesley, Reading, MA, 1972.

56. Cvetkovic, D., P. Rowlinson, and S. Simic. *Spectral Generalizations of Line Graphs: On Graphs with Least Eigenvalue-2.* Cambridge University Press, 2004.

57. Rifkin, R. M., and R. A. Lippert. *Notes on Regularized Least Squares.* Center for Biological and Computational Learning (CBCL), Massachusetts Institute of Technology, Massachusetts Institute of Technology, Cambridge, MA, USA, 2007.

58. Jenelius, E., and H. N. Koutsopoulos. Probe Vehicle Data Sampled by Time Or Space: Consistent Travel Time Allocation and Estimation. *Transportation Research Part B: Methodological*, Vol. 71, 2015, pp. 120-137.

59. Knuth, D. E. Two Notes on Notation. *American Mathematical Monthly,* 1992, pp. 403-422.

60. Niu, X., Y. Zhu, Q. Cao, X. Zhang, W. Xie, and K. Zheng. An Online-Traffic-Prediction Based Route Finding Mechanism for Smart City. *International Journal of Distributed Sensor Networks*, Vol. 501, 2015, pp. 970256.