LOCATION PRIVACY: TRACKING DRIVING ROUTES USING SPEED DATA

by

XIANYI GAO

A dissertation submitted to the

School of Graduate Studies

Rutgers, The State University of New Jersey

In partial fulfillment of the requirements

For the degree of

Doctor of Philosophy

Graduate Program in Electrical and Computer Engineering

Written under the direction of

Prof. Janne Lindqvist

And approved by

New Brunswick, New Jersey

October 2018

ABSTRACT OF THE DISSERTATION

Location Privacy: Tracking Driving Routes Using Speed Data

By XIANYI GAO

Dissertation Director:

Prof. Janne Lindqvist

Advances in technology have provided ways to measure driving behavior. Recently, this technology has been applied to usage-based automotive insurance. Policy holders may opt-in to monitoring for the hope of reduced insurance premiums. Although some of these monitoring devices are based upon GPS information and offer no location privacy protections, several companies are aware of the privacy concerns and therefore measure only speed data. However, does collecting the speed data really preserve privacy? Our work investigates how much location information we can actually obtain from the speed data and why the speed data should also be protected against malicious third parties. In this thesis, we present our algorithm to track drivers' locations when only speed data and starting locations are known. The starting locations are mostly home addresses that insurance companies know. The algorithm fits the speed data to a trajectory path on a map and evaluates which path should be the actual driving route. To demonstrate the algorithm's real-world applicability, we evaluated its performance with driving datasets from New Jersey and Seattle, Washington, representing suburban and urban areas.

We present the Elastic Pathing algorithm to track drivers, the enhanced version of Elastic Pathing algorithm with several optimizations, and a final machine learning approach by learning how a speed pattern can indicate the driving direction. Our Elastic Pathing algorithm can estimate destinations with error within 250 meters for 17% traces and within 500 meters for 24% traces in the New Jersey dataset (254 traces). For the Seattle dataset (691 traces), we similarly estimated destinations with error within 250 and 500 meters for 16% and 28% of the traces respectively. At the end, based on the challenge from previous approach, we designed and implemented the machine learning approach for the current New Jersey dataset in order to achieve higher accuracy. With machine learning, our algorithm was able to estimate destinations with error within 250 and 500 meters for 25% and 30% of traces respectively in our New Jersey dataset. This work shows that speed data enable a substantial breach of privacy.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my Ph.D. advisor, thesis committee members, collaborators, colleagues, and my friends and family. I would not be able to complete my Ph.D. study without their guidance and support.

First, I would like to specially thank my Ph.D. advisor, Prof. Janne Lindqvist. I have known about Janne since my undergraduate summer internship at Rutgers WINLAB. His enthusiasm for discovering innovative research ideas and finding solutions for challenging problems has left a great impression on me. With my interest on human-computer interaction and security engineering, I joined Janne's research group for my Ph.D. study. During these years, I truly appreciate Janne's advising. Starting from when I did not know about academic research to my final thesis, Janne has provided the major support and guidance throughout my Ph.D. study. He is very dedicated on directing my research topic and resolving any issue that I encountered. He also plays the major role in creating a friendly and helpful research group in our lab.

My great appreciation also goes to my collaborators and colleagues. Thanks for Prof. Antti Oulasvirta's mentoring and advise on one of my research projects. He is a very nice person that has a great enthusiasm towards research. He also has a good understand of the research field and provided me many essential advises on study design and result analysis. I would also like to thank Prof. Richard Howard for providing insightful comments for my research. I am also very lucky to have nice colleagues working in our lab and helping each other. Thanks for the help from Yulong Yang, Huiqing Fu, and Fengpeng Yuan on the starting of my projects when I first joined the group. Thanks for the help from Christos Mitropoulos, Hua Deng, and Can Liu who often provide useful technical suggestions. Thanks for the help from Gradeigh Clark and Shridatt Sugrim on providing comments, editing and polishing papers. I would also like to thank Meghan McLean for providing useful comments from the view of a physiologist.

I would like to sincerely thank Prof. Athina Petropulu, Prof. Roy Yates, Prof. Waheed Bajwa, Prof. Antti Oulasvirta, and my advisor Prof. Janne Lindqvist for writing recommendation letters for my graduate fellowships and job applications. I would also like to thank my thesis committee members for their valuable comments towards my thesis work.

In addition, I would like to thanks all my friends for keeping me happy while being productive on my research. Special thanks to my friend, Talia Waltzer. As a friend who knows my research work, my culture, and my personality, she often provides important emotional support when I was frustrated or stressed. Her hardworking and optimism toward life difficulties often motivate me to work hard and not to give up.

Last but not least, I would like to thank my parents who have been supporting me all the time, despite their financial difficulty. I would also like to thank my aunt's family for providing me a nice place to stay for my Ph.D. study years.

ACKNOWLEDGMENT OF PREVIOUS PUBLICATIONS

- Xianyi Gao, Bernhard Firner, Shridatt Sugrim, Victor Kaiser-Pendergrast, Yulong Yang, and Janne Lindqvist, “Elastic Pathing: Your Speed is Enough to Track You”, In ACM International Joint Conference on Pervasive and Ubiquitous Computing (Ubicomp’14), 2014

- Xianyi Gao, Bernhard Firner, Shridatt Sugrim, Victor Kaiser-Pendergrast, Yulong Yang, and Janne Lindqvist, “Transforming Speed Sequences into Road Rays on the Map with Elastic Pathing”, (under review for journal publication).

ACKNOWLEDGMENT OF FUNDING SUPPORT

This material is based upon work supported by the National Science Foundation under Grant Numbers 1228777 and 1211079. In addition, I, Xianyi Gao, was supported by the National Science Foundation Graduate Research Fellowship Program under Grant Number 1433187. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

TABLE OF CONTENTS

ABSTRACT ii

ACKNOWLEDGEMENTS iv

ACKNOWLEDGMENT OF PREVIOUS PUBLICATIONS vi

ACKNOWLEDGMENT OF FUNDING SUPPORT vii

LIST OF TABLES x

LIST OF FIGURES xii

CHAPTER 1. INTRODUCTION 1
1.1 Overview 1
1.2 Organization 4
1.3 Contributions 5

CHAPTER 2. BACKGROUND AND RELATED WORK 6
2.1 Location Data and Privacy Protection 6
2.2 Location Tracking with Various Sensors 8
2.3 Location Tracking with Speed Data 9

CHAPTER 3. ELASTIC PATHING ALGORITHM AND ESTIMATION ACCURACY 13
3.1 Chapter Overview 13
3.2 Requirements for a Generic Algorithm 13
3.3 Elastic Pathing Algorithm 15
3.4 Driving Data 22
3.4.1 Central New Jersey Dataset 24
3.4.2 Seattle Dataset 25
3.5 Results 26
3.5.1 New Jersey Dataset Result 27
3.5.2 Seattle Dataset Result 27
3.5.3 Analysis 29

CHAPTER 4. ENHANCED ELASTIC PATHING ALGORITHM WITH IMPROVED ACCURACY 34
4.1 Chapter Overview 34
4.2 Optimization for Elastic Pathing Algorithm 34
4.2.1 Visualization Tool 41
4.3 New Driving Data 42
4.3.1 Identical-Route Dataset 43
4.4 Results 44
4.4.1 Overall Accuracy 45
4.4.2 Comparison with Naive Guessing 50
4.4.3 Effect of Applying OSM Routing 52
4.4.4 Effect of Applying Road Speed Limits 54
4.4.5 Results for Identical-Route Dataset 55
4.4.6 Analysis on Traces with Low Estimation Accuracy 60

CHAPTER 5. COMBINING MACHINE LEARNING WITH ELASTIC PATHING 64
5.1 Chapter Overview 64
5.2 Introduction of Machine Learning Approach 64
5.3 Algorithm Design: Machine learning with Elastic Pathing 67
5.3.1 Overall Structure of Our Algorithm 68
5.3.2 Implementation of Our Machine Learning Model 70
5.3.3 Path Selection and Probability Estimation 73
5.3.4 Simplified Elastic Pathing for Ranking of Selected Paths 78
5.4 Results 82
5.4.1 Overall Accuracy 82
5.4.2 Evaluation of Machine Learning Model 89

CHAPTER 6. DISCUSSION 94

CHAPTER 7. CONCLUSION 98

BIBLIOGRAPHY 99

LIST OF TABLES

3.1 Results for both central New Jersey dataset and Seattle dataset from our elastic pathing algorithm. Table shows the number of traces having endpoint (or destination) error ranging from 0 to greater than 3.25 km, separated into 14 error intervals as shown in rows. First two columns give endpoint error representations in meters and miles. 28

4.1 Results from the enhanced version of our elastic pathing algorithm for both the New Jersey and the Seattle dataset. Table shows the number of traces having destination error ranging from 0 to greater than 3.25 km, separated into 14 intervals as shown in rows. First two columns give destination error representations in meters and miles. 46

4.2 Comparison of estimation accuracy between two versions of our algorithm. Table shows percentages of traces that has destination error within certain range for both New Jersey and Seattle dataset. 47

4.3 Comparison of estimation accuracy when OSM routing is applied to our algorithm. Table shows percentages of traces that has destination error within certain range for both New Jersey and Seattle dataset. 54

4.4 Table shows number of traces with wrong estimation caused by each one of the six factors. 63

5.1 Summary of Features for Machine Learning. Table shows all the features and corresponding value types. Therefore, each training sample contains 13 features: 10 for speed segment, one for number of path options, one for selected driving direction, and one for turn angle. 72

5.2 Overall results of our machine learning approach for New Jersey dataset. Table shows the number of traces that our algorithm can estimate with a destination error within different intervals. For example, the first error interval is 0-250 meters. Traces that has destination error within this interval are considered as the most accurate traces predicted by our algorithm. Column "ML+SEP" indicates the part of driving traces that have candidate paths selected through our machine learning model and then these candidate paths are passed to our Simplified Elastic Pathing to find the solution path. Column "CEP" indicates the part of driving traces that do not have any highly likely candidate path selected through our machine learning model. Then, they are re-run to find the solution path through our complete Elastic Pathing algorithm. "Total" column shows the total number of traces that are estimated with destination error within the corresponding error interval. 83
5.3 Comparison of overall results for New Jersey dataset: the enhanced version of our Elastic Pathing algorithm vs. the machine learning version that we present in this chapter. 84
5.4 Overall accuracy of our machine learning approach for Identical-Route dataset. We use a different scale for destination error measurement because the estimation accuracy for this dataset is lower than other datasets. Intervals are: 0-1 mile, 1-2 miles, 2-3 miles, 3-4 miles, 4-5 miles, and greater than 5 miles. We mostly consider the 0-1 mile error interval as our algorithm predicts to the right direction when the error is within one mile (given that each trip length is about 5 miles). Again, column "ML+SEP" indicates the part of driving traces that have candidate paths selected through our machine learning model and then these candidate paths are passed to our Simplified Elastic Pathing to find the solution path. Column "CEP" indicates the part of driving traces that do not have any candidate path selected by our machine learning model and the solution path was found through rerunning our complete Elastic Pathing algorithm. 89
5.5 Comparison of overall results for Identical-Rout dataset: the enhanced version of our Elastic Pathing algorithm vs. the machine learning version that we present in this chapter. 89

LIST OF FIGURES

1.1 Number of paths within a given distance of a starting location. The pattern of growth on a log-linear scale shows that the number of paths increases exponentially. Within one mile there are over 10^6 paths diverging from a grocery store and over 10^4 paths from a residential area. Locations central to major transportation routes, such as a grocery store, will see a higher increase in paths compared to a residential location which is more likely to lead to a dead end. Residential roads are also more likely to lead to a dead end: in this example the number of paths from the residence actually falls at the beginning because only one path does not dead-end immediately. . . . 3

3.1 The number of driving traces for each volunteer in the New Jersey dataset (top figure) and the Seattle dataset (bottom figure). 24

3.2 Percentage of individual driving traces that have endpoint error less than 250 meters and 500 meters. For NJ volunteer 2, there is no bar, which means 0% of the traces had the corresponding error. 30

3.3 The best possible case, a path from a grocery store to a home with no error. The ground truth is indicated with a solid line while the nodes along the predicted path are shown as dots. In this case the predicted and actual paths match perfectly. 32

4.1 (a) The figure in the left presents our initial approach to estimate the turn radius which is set to be the radius of the circle inscribing the triangle (shaded gray in the figure). (b) The right figure shows our enhanced turn radius calculation method. The turning angle does not have to be 90 degrees and it has the maximum turn radius among all possible turns made within the path boundary. 35

4.2 The visualization tool for our elastic pathing algorithm: The upper marker indicates the starting location and the lower marker indicates the current location the algorithm was exploring. The yellowish green line shows the ground truth GPS points plotted with a heatmap (e.g. the more the color shifts towards red, the denser the GPS points are – thus the slower the car moves). Gray lines show several top candidate paths that the algorithm has explored. The red line shows the best candidate path up to the current time point. 42

4.3 Route A shows the driving route from point a to point b, which comprises residential and commercial areas with some traffic lights. Route B shows the returning route from point b back to point a. This route consists of a long highway segment which connects local paths with a few traffic lights. . 45

4.4 The cumulative distribution function curve for destination error within one mile from our New Jersey dataset. 48

4.5 The cumulative distribution function curve for destination error within one mile from our Seattle dataset. 49

4.6 Box plots of estimation accuracy (or destination error) vs. driving trip length for the enhanced version of elastic pathing algorithm. The destination error is split into 14 intervals (e.g. interval 0-0.16 miles, interval 0.16-0.31 miles) with each interval equal 0.155 miles or (250 meters). Trip length distribution for each error interval is presented with a box-plot bar, showing: minimum, maximum, first quartile to third quartile range (blue box), median (red line), and mean (blue circle). 51

4.7 Effect of applying speed limits on the New Jersey and the Seattle dataset. Figure shows the percent of driving traces from the corresponding dataset with destination error within different intervals: from 0-250 meters (0.16 mile) to greater than 3250 meters (2 miles). 55

4.8 Comparison of estimation accuracy when using the original map and the fixed map. Figure shows the percent of traces with destination error within different intervals: from 0-1 mile to 6.5 miles. There is no driving trace with destination error greater than 6.5 miles. 57

4.9 The effect of average speed and average braking deceleration to the estimation accuracy. The range of average speed or average braking deceleration is split into four intervals. Values in horizontal axis show both the interval and the number of traces within the interval. The percentage value in vertical axis shows the percent of driving traces within each interval having destination error within one mile. 61

4.10 The effect of brakes and stops to the estimation accuracy. We use “brakes” in the figure to denote “braking events”. Each range is split into four intervals with integer values. Values in horizontal axis show both the interval and the number of traces within the interval. The percentage value in vertical axis shows the percent of traces within each interval having destination error within one mile. 62

5.1 Overall Structure of Our Algorithm. Figure shows how we apply and combine machine learning model to our Elastic Pathing algorithm. 69

5.2 Intersection on the Map and Selected Direction. Figure shows the intersection that a driver is passing through. The marker 1 indicates the starting point of entering the intersection. The marker 2 indicates the intersection point. The blue line marked the selected direction. In this figure, we assume the driver is turning right with 72 degree of turning angle. The ground-truth label indicates whether this is an actual driving direction picked by the driver. 72

5.3 Example of how our path selection algorithm explore and select. We assume that driver enters this region from point A. 76

5.4 Figure shows the cumulative distribution function curves for destination error within one mile from our New Jersey dataset. The CDFs for both machine learning version and enhanced version of previous Elastic Pathing algorithm are shown in the figure for comparison. 85

5.5 Figure shows the cumulative distribution function curves for destination error within one mile from our Identical-Route dataset. The CDFs for both machine learning version and enhanced version of previous Elastic Pathing algorithm are shown in the figure for comparison. 87

5.6 Receiver Operating Characteristic (ROC) curves and EER values from six drivers in the New Jersey dataset. The horizontal axis shows the false positive rate (FPR) and the vertical axis shows the true positive rate (TPR). . . . 92

5.7 The Receiver Operating Characteristic (ROC) curve and the EER value for the machine learning model for the Identical-Route dataset. The horizontal axis shows the false positive rate (FPR) and the vertical axis shows the true positive rate (TPR). 93

CHAPTER 1

INTRODUCTION

1.1 Overview

Technological advances have created ways to observe driving behaviors using odometer readings and in-vehicle telecommunication devices. Taking advantage of this technology, some US-based automotive insurance companies (Allstate, 2013; GMAC, 2015; Progressive, 2013; StateFarm, 2016) offer consumers the ability to opt-in to the usage-based insurance (UBI) for reduced premiums by allowing companies to monitor their driving behavior. These policies are different from traditional insurance policies, which use record of past driving violations to differentiate between safe and aggressive drivers. The UBI policy applies devices to monitor directly while people are driving and provide insurance companies with real-time driving data.

Although the UBI policy has advantages for both insurers and consumers (e.g. insurers monitor consumers' driving behaviors as a way to encourage safe driving, and consumers get lower premiums), the privacy concern of collecting driving behavior data should not be neglected. Some of these monitoring devices are even based upon GPS information and offer no privacy protection, such as OnStar (General Motors, 2016). Many insurance companies are aware of the privacy issue and only use devices that collect time-stamped speed data, making the claim that it is privacy-preserving. However, it is still possible to deduce location data from speed traces. In this thesis, we demonstrate that logging this time-stamped speed data is also not privacy-preserving, despite the insurance companies' claims.

The privacy problem introduced in this thesis is significant because the data collection by insurance companies is an always on activity. Data may be logged forever, and thus, deducing location data from speed traces represents a huge breach of privacy for drivers having these types of insurance policies. Even if insurance companies are not currently obtaining location traces from their data today, this does not guarantee that it will not present problems in the future. The data is not considered to be sensitive data, and therefore is also unlikely to be treated as sensitive. This means that the data may eventually be obtained by any number of antagonists that do know how to process the data traces to obtain location information. For example, it is foreseeable that law enforcement agencies would request the information as they have done with other driving related data, for example, electronic toll records (Apuzzo, 2003).

Tracking drivers and estimating their trip destinations are very challenging with only a starting location and the time-stamped speed data, otherwise insurance companies would not claim that the information being gathered is privacy-preserving. Indeed, matching a single speed trace to all of the roads in a country or state is intuitively extremely difficult. However, the home address of a person is available to insurance companies, making a starting location available for some paths.

It is not obvious that these data are sufficient to reproduce an exact driving path. Speed data does not indicate if a person is turning at an intersection or merely stopping and going straight after. Without information about driving directions, there may be multiple alter-

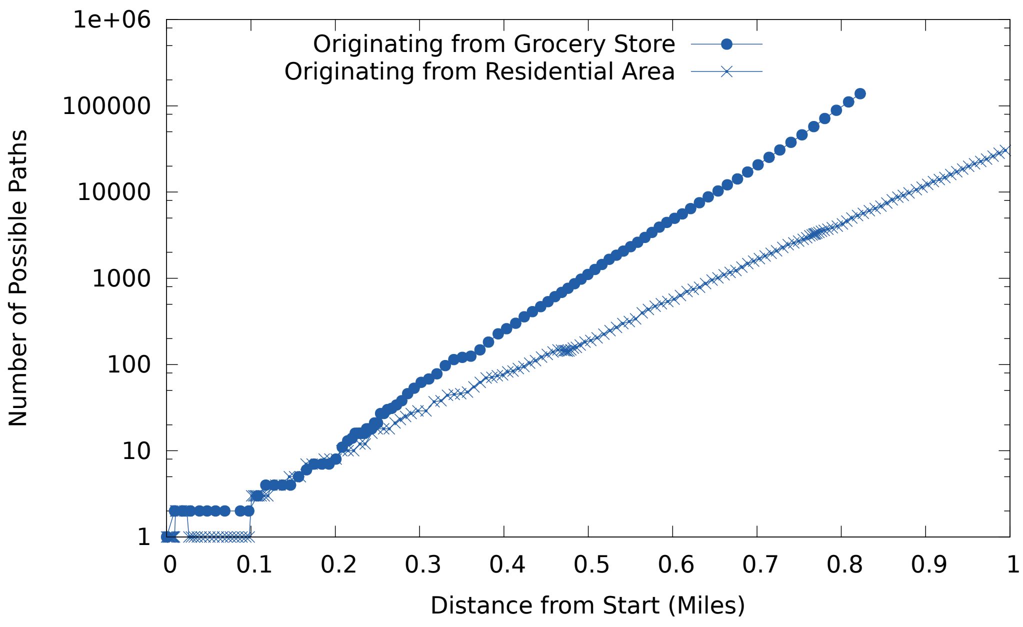

Figure 1.1: Number of paths within a given distance of a starting location. The pattern of growth on a log-linear scale shows that the number of paths increases exponentially. Within one mile there are over 10^6 paths diverging from a grocery store and over 10^4 paths from a residential area. Locations central to major transportation routes, such as a grocery store, will see a higher increase in paths compared to a residential location which is more likely to lead to a dead end. Residential roads are also more likely to lead to a dead end: in this example the number of paths from the residence actually falls at the beginning because only one path does not dead-end immediately.

native paths that match with the speed data. Of course multiple routes can be explored. However, even within a few minutes drive of a person's home, there may be thousands of turns the person could have taken, so blindly exploring every single possible path is not feasible. Figure 1.1 shows the growth of possible paths within a distance of just one mile from a starting location. Within that one mile, there are over 100,000 possible paths the driver could have taken when the trip starts from a grocery store, and over 10,000 paths from a residential area.

A general approach to reducing the space of possible paths would be to build an error metric for all paths and choose the path that minimizes the error. Due to the rapid growth rate of paths, the number of possible paths at the end of a trip is very large. Simple solutions, such as greedy algorithms, will end in failure without even finding any solution; for instance when the path chosen by a greedy algorithm suddenly comes to a dead end.

This thesis presents and applies several innovative ways (e.g. use of speed limit, turn-speed limitation, error metric, priority first search, and machine learning) to approach the problem. Despite the challenge, with our enhanced Elastic Pathing algorithm, we are able to estimate destinations with error within 250 meters for 17% of the traces and within 500 meters for 24% of the traces in the New Jersey dataset, and with error within 250 and 500 meters for 16% and 28% of the traces, respectively, in the Seattle dataset. After analyzing on our algorithm and factors that can mislead our algorithm, we designed and implemented the machine learning approach that adds probabilistic information for driving directions at intersections. With this final machine learning approach, we were able to further improve our algorithm's estimation accuracy: 25% traces from New Jersey dataset were estimated with destination error less than 250 meters, and 30% traces were estimated with destination error less than 500 meters.

1.2 Organization

Chapter 2 describes the background and related work of location prediction based on sensor data. Chapter 3 presents the design and implementation of our innovative Elastic Pathing algorithm and the corresponding accuracy with our datasets. Chapter 4 presents the enhanced version of the Elastic Pathing algorithm. Chapter 5 presents the machine learning approach which further improves the estimation accuracy. Chapter 6 discusses our algorithm, results, and location privacy with driving data. Chapter 7 concludes this thesis.

1.3 Contributions

This work makes following contributions:

1. We present our innovative Elastic Pathing algorithm which estimates location traces from speed data and starting locations.

2. We built a tool to visualize how our algorithm tracks drivers. Our tool, utilizing Google Maps APIs, can be applied to any application having driving traces.

3. We tested our algorithm on real world traces to show how the data collected by many insurance companies is not privacy-preserving despite their claims.

4. We explored the effect of adapting OpenStreetMap routing into our algorithm and showed it does not benefit our algorithm but can be potentially useful on estimating routes for a subset of drivers.

5. We analyzed how information about speed limits would affect our estimation accuracy in two distinct driving environments: urban and suburban areas.

6. We prepared three large datasets for testing: New Jersey dataset representing suburban driving traces, Seattle dataset representing urban driving traces, and an identical-route dataset of 30 drivers.

7. We explored how our algorithm performs for various drivers and driving behaviors.

8. We presented the innovative machine learning approach and how this can be combined with our algorithm to achieve an improved estimation accuracy.

CHAPTER 2

BACKGROUND AND RELATED WORK

2.1 Location Data and Privacy Protection

Many studies have shown the importance of location data for personal behavioral information. For example, Brush et.al. did a study with 32 participants from 12 house holds about their location privacy concerns (Brush, Krumm, & Scott, 2010). Different location obfuscation schemes were presented to participants to find the most comfortable one to help with privacy protection. Their findings suggested that designers may be able to provide an informed choice about location obfuscation based on user's personal privacy concerns. In addition, Ludford et. al. investigated on the sharing and protecting of the location information through conducting two empirical studies (Ludford, Priedhorsky, Reily, & Terveen, 2007). They found that new local knowledge about a variety of places can be discovered from location data. Meanwhile, Patil and Lai's study showed that people are sensitive to location information but they are still willing of sharing location information to close groups (e.g. family, team) (Patil & Lai, 2005). Their findings suggested that some group-level based mechanisms can be used to balance privacy control and configuration burden. On the other hand, Tsai et. al. focused on the impact of feedback in the context of mobile location sharing (Tsai et al., 2009). They found participants having very strong concern on their privacy information. They showed that feedback is an important factor to improve user comfort levels and peer opinion plays an important role on determining whether participants would continue to use a mobile location technology.

The importance of location privacy has encouraged a large body of work to concentrate on protecting, anonymizing, or obfuscating location traces. For example, Brush's work focused on location obfuscating methods (Brush et al., 2010). Popa et. al. developed a system called "PrivStats" to guarantee the protection of location privacy (Popa, Blumberg, Balakrishnan, & Li, 2011; Gruteser & Grunwald, 2003). The system achieved effective protection by using an novel protocol to upload data anonymously. Gruteser and Grunwald's work also focused on location data anonymization (Gruteser & Grunwald, 2003). Through adjusting resolution of location information spatially or temporally, the algorithm was able to meet specified anonymity constraints for location-based services. Another study used false location reports combined with a probabilistic model producing realistic driving trips to confuse location attacker (Krumm, 2009a). Similarly, a study used spatial cloaking, noise, and rounding to obscure location data in order to prevent location attacks (Krumm, 2007; Zhou et al., 2013). In addition, Krumm has written an overview of computational location privacy techniques (Krumm, 2009b), and Zang and Bolot have recently questioned the possibility of releasing privacy-preserving cell phone records while still maintaining research utility in those records (Zang & Bolot, 2011).

Based on analysis of location data, researchers have shown that the nature of individual mobility patterns is bounded, and that people visit only a few locations most of the time (e.g. just two (Gonzle, Hidalgo, & Barabasi, 2008; Eagle & Pentland, 2009; Golle & Partridge, 2009)). In most cases, there is high level of predictability for future and current locations (e.g. (Song, Qu, Blumm, & Barabsi, 2010; Farrahi & Gatica-Perez, 2011) – most trivially, “70% of the time the most visited location coincides with the user’s actual location” (Song et al., 2010)). Mobility patterns can also be used to predict social network ties (e.g. (Wang, Pedreschi, Song, Giannotti, & Barabasi, 2011)). In a more applied domain, GPS traces have been used to learn transportation modes (Zheng, Chen, Li, Xie, & Ma, 2010), to predict routes (Ziebart, Maas, Dey, & Bagnell, 2008), to predict family routines (e.g. picking up or dropping off children) (Davidoff, Ziebart, Zimmerman, & Dey, 2011), to distinguish when people are home or away (Krumm & Brush, 2011), and to recommend friends and places (Zheng, Zhang, Ma, Xie, & Ma, 2011). However, none of the above work can be used to discover locations based solely on speed data and a driver’s starting location.

2.2 Location Tracking with Various Sensors

The world’s first navigating system from Etak in 1985 showed how well a car could be located without using GPS (Zavoli & Honey, 1986). The navigating system used a map database, a dead reckoning system with motion sensors, and a magnetic compass to track the driving direction, distance, and ultimately the real time location. This system cannot be simply applied to our work as we do not have the information about driving direction. Other related work in the field of dead reckoning also suggests that speedometer traces should have some level of information to extract. Dead reckoning works by using speed or movement data starting from a known location to deduce a movement path. Dead reckoning has been previously used for map building with mobile robots (Golfarelli, Maio, & Rizzi, 1998) or as an addition to Global Navigation Satellite Systems (GNSS) such as GPS (Krakiwsky, Harris, & Wong, 1988; Zhao, Ochieng, Quddus, & Noland, 2003).

These systems, however, require very accurate measurements of speed and acceleration, and utilize very precise gyroscopes to determine changes in direction. When supplementing GNSS data, dead reckoning was only used when GNSS information is unavailable, such as when a vehicle passes through a tunnel.

In the specific area of usage-based insurance, the privacy concerns of these insurance programs have been studied before by Troncoso et al. (Troncoso, Danezis, Kosta, Balasch, & Preneel, 2011). However, this work dates back to schemes which would send raw location (GPS) coordinates to either insurance providers or brokers. Troncoso et al. proposed a cryptographic scheme PriPAYD to address the problem (Troncoso et al., 2011). Our work shows that speedometer-based solutions, which were not considered by Troncoso, are not privacy-preserving either. Technically similar to our work are side-channel attacks using accelerometer data from smartphones to infer user location (Aviv, Sapp, Blaze, & Smith, 2012). Projects such as ACComplice (Han, Owusu, Nguyen, Perrig, & Zhang, 2012) and AutoWitness (Guha et al., 2010) have used accelerometers and gyroscopes for localization of drivers. However, the information from the smartphone can be used to detect when turns occur. In contrast, we have only a time series of speed data available. It does not indicate if any turn is taken.

2.3 Location Tracking with Speed Data

Finally, the most closely related work to our own is a project to infer trip destinations from driving habits data by Dewri et al. (Dewri, Annadata, Eltarjaman, & Thurimella, 2013; Bellatti et al., 2017). Their work used time, speed, and driving distance data to estimate destinations. Their algorithm is based on depth-first search (DFS) to explore all the paths within the range. After generating candidate paths, post-processing is needed to calculate the actual ranking of paths. The average time complexity is equivalent to the DFS exploration plus the postprocessing, which increases exponentially with the length of speed data. The algorithm also needs to store all the exploring paths within the range. In contrast, our algorithm has much less time and space overhead because of the use of error metrics and prioritizing the search of best paths.

Dewri et. al. tested the algorithm with only 30 trips in Denver, Colorado area and mentioned the estimated destinations are close to the actual one (within 0.5 mile) (Dewri et al., 2013; Bellatti et al., 2017). However, each of the 30 trips was pre-processed to "remove data points that may correspond to driving in traffic condition" (Dewri et al., 2013). The algorithm also assumes that "all left turns happen at a speed of 15 mph and all right turns happen at 10 mph" (Dewri et al., 2013; Bellatti et al., 2017). It is unclear how this assumption stands for different drivers. The algorithm has not been tested in a large set of non-ideal driving routes with a collection of different drivers and driving habits. In contrast, our elastic pathing algorithm estimated about 37% of all traces within 0.5 miles using datasets with nearly one thousand daily driving traces. Our algorithm works in realistic driving conditions without manual pre-processing or pre-selecting the collected driving traces.

Another advantage of our elastic pathing algorithm over Dewri's algorithm is the applicability to real-time tracking, since our search algorithm is only based on the current and past speed samples. No post-processing or information of future speed samples are needed to estimate the current location. The efficient search algorithm can generate the estimations quickly with some optimization. Therefore, it is potentially useful for this type of adaption.

The most recent work on driving location tracking based on speed data was in 2017 through using a particle-based framework to estimate a driver's location (Wahlstrm, Skog, Rodrigues, Hndel, & Aguiar, 2016). The main idea was to implement a particle filter to propagate estimated driver locations, modeled by particles with estimated yaw angles, yaw rates, and target speeds, by using two measurement functions for weight updates. The first function was derived based on the assumed condition that the lateral force on the vehicle does not exceed critical limits (e.g. usually smaller than 0.6 g and larger than a lower bound) and assuming the uniform probability distribution between the two limits. The second function was based on the assumption of a target speed that a driver was probably going to reach. The idea and the estimation of the target speed were based on assumptions that drivers tend to maintain a speed as much as possible near the speed limit and driver's acceleration is proportional to the difference between the current speed and the target speed.

This work was very innovative of using probability model combined with proper particle filters for measurement value updates (Wahlstrm et al., 2016). While the information provided to the particle filters appear to be quite limited (e.g. without the use of speed limits and other features of speed data), the model was able to prediction with relatively good accuracy with 18 traces (e.g. 25% traces with error less than 200 meters with the best probability model). However, similar to Dewri's work (Dewri et al., 2013), this model was only tested among very small set of driving traces (Wahlstrm et al., 2016), making the comparison to our algorithm not feasible (i.e. very different approach with datasets in very different size and driving area). The model processing time and complexity were also not reported in the paper. When the driving dataset gets more comprehensive, assumptions from the model may not always hold. For example, the assumption of target speed and the idea of drivers maintaining speed near speed limit may vary in different driving routes (e.g. highway vs. local), different traffic conditions, and different drivers (e.g. aggressive vs. defensive drivers). Therefore, it is unclear how the model performs with large collection of driving datasets with various driving routes and drivers. However, this work shows a good potential to solve this problem through statistical modeling.

CHAPTER 3

ELASTIC PATHING ALGORITHM AND ESTIMATION ACCURACY

3.1 Chapter Overview

In this section, we describe the Elastic Pathing algorithm used to recreate a person's driving route when the time-stamped speed data and the starting location are given. In the following, we start with discussing the basic requirements for a generic algorithm and continue by presenting the design of our Elastic Pathing algorithm. Then, we present the datasets used to test the algorithm. At the end, we show the estimation accuracy with this initial version of our algorithm. This section of the work has also been published (Gao et al., 2014).

3.2 Requirements for a Generic Algorithm

To solve this path recreation problem, any generic algorithm needs to consider: resources, restrictions, and scoring metric for ranking. Resources are all of the contextual data that will help us deduce a path from speed data. For example, a map is a useful resource to apply. Restrictions are physical or behavioral limitations to any automotive driving behavior. Finally, a scoring metric is needed to choose the best path out of several possible candidates.

Applying Available Resources: First, we would need the information about how different paths are connected in the area. OpenStreetMap (OSM) is a great option for this purpose. A path (or a way (OpenStreetMap, 2016b)) in OSM is represented with an ordered list of nodes (OpenStreetMap, 2016a). From the paths of the OSM, we know which nodes are adjacent, and once a node is reached, we know which nodes we can move towards next. For example, a four-way intersection will have a single node at the intersection and four adjacent nodes. Since we reach the intersection from one direction, we have three possible next nodes from which to choose. The distance between road segments and the angles required to make a turn from the latitude and longitude pairs of the nodes can also be calculated.

OSM also includes other information such as speed limits that are useful for path prediction. Having speed limits for paths should improve the algorithm accuracy and efficiency by eliminating impossible candidate paths given speed trace. Other information included in OSM such as turn restrictions and way types are also helpful for the algorithm.

At the same time, time-stamped speed data also contain essential information. For example, we can easily obtain the driving distance and acceleration for the speed data. Therefore, a generic algorithm should use these quantities as part of the match criteria when fitting a speed trace to paths in the map.

Adapting to Driving Restrictions: There are physical limitations to a vehicle's turning radius at a high speed. When a vehicle travels a particular path, it must travel at a speed at or below the maximum speed possible to make the turn. Also, when driving in a road, drivers need to obey speed limits. Although driving under maximum speed limit may not be always true for each driver, it still provides useful boundary cases for the algorithm.

During normal driving, people only stop when they have to. There are only a few scenarios when people would commonly stop: (1) they encounter red traffic lights or stop signs, (2) they start off from a place, (3) they reach to a destination, and (4) other cases such as road constructions, traffic, and pedestrians crossing the street. Our algorithm does not consider the last case since it happens randomly and it is not a common artifact in all traces.

Using an Error Metric for Paths: A comparison metric is needed for the algorithm to determine the best path. There should be a scoring system representing how well a path matches with the speed trace. It is difficult to find a path that perfectly matches the speed data, because drivers may swerve around objects in the road, or take turns more widely or sharply than we expect. Thus, the actual driving distance may not perfectly match with the path length calculated using the map. The challenge is that in some scenarios, these errors might cause even correct path to seem impossible. For example, if the model progresses past the vehicle's actual position along a segment of road, then the model may conclude that the vehicle is moving too quickly to make a turn. If we corrected the distance traveled to account for some of these distance estimation errors, then we may find that the speed traces line up perfectly with the turn. Thus, any algorithm must correct paths while it explores them in an attempt to take into account these variations in the travel distance.

3.3 Elastic Pathing Algorithm

The elastic pathing algorithm is based on compressing or stretching the estimated distance that was traveled as we attempt to match the speed data to the path. We match zero speeds to intersections in the map and slow speeds to turns. Based on the daily driving scenario, these can only be one-way matching – if the speed is zero, there needs to be an intersection; but if there is an intersection, the speed does not have to be zero (e.g. when the traffic light is green). Similarly, if there is a turn, the speed needs to be slow enough to make the turn; but if the speed is slow (greater than zero), the car may not be turning. The compressing or stretching is done when there is a mismatch between the calculated distance from speed data and the length of a path segment in the map. Compressing means subtracting from the estimated distance by a specific amount. Stretching means adding to the estimated distance by a specific amount.

After reconciling differences between a section of road and the speed trace, we must *pin* the path at that point (which we call a landmark) because any movement would cause a mismatch between the two. For instance, if the driving speed in a trace is reduced to zero, indicating a stop, where there is no intersection we might pull the path forward by some distance to reach an intersection. All points that are pinned cannot be moved since they align with features in the road. We call this approach *elastic pathing* because the stretching and compressing of the speed traces to fit the road is conceptually similar to stretching a piece of elastic along a path while pinning it into place at different points. To understand the algorithm, we first introduce a set of definitions:

- **Calculated Distance:** The distance a vehicle traveled calculated from speed and time values in the speed trace.

- **Predicted Distance:** The distance along a possible route on the road at a certain time in the speed trace.

- **Error:** The difference between calculated distance and predicted distance of a possible route.

```
Input: The starting point-StartNode, a set of speed samples- Samples, and a
       threshold within the best possible score to accept- $\delta$
Result: Complete = list of possible paths within $\delta$ of the best possible
begin
    Partial $\longleftarrow$ {[StartNode ]}
    Complete $\longleftarrow \emptyset$
    while Complete $= \emptyset$ OR
    Partial.first.error $< \delta \times$ Complete.first.error do
        // Pop out the partial path with smallest error to
           advance
        P' $\longleftarrow$ gotoBranch (Partial.pop)
        // New paths are found after gotoBranch, then
           check for completed paths
        join (Complete, $\{x \in$ P' $\mid x$ complete $\}$)
        join (Partial, $\{x \in$ P' $\mid x$ incomplete $\}$)
        // Sort by error:  the first partial path
           (Partial.first) has the smallest error
        sort (Partial)
    end
    // The first completed path found has the smallest
       error
end
```

Algorithm 1: Pseudocode for the elastic pathing algorithm. Starting with the StartNode, partial paths are added and processed through *gotoBranch* function. In each iteration, partial paths are sorted so that the one with smallest error gets prioritized in search.

- **Feature:** A vehicle stop in the speed trace, an intersection in the road, or speed slow enough to make a turn.

- **Landmark:** A place (e.g. intersection or turn) where the speed trace and road data need to be checked to match.

The pseudocode for our elastic pathing algorithm is given in Algorithm 1. We measure the fit of a path by the amount of stretching or compressing that needs to be done for the speed data to match the path. The more stretching or compressing along a path, the worse this path scores. In Algorithm 1, the *error* attribute for each candidate path measures this quantity. The algorithm stops exploring when it finds a set of possible solution paths (stored in “Complete” array) and the smallest error from other left-over uncompleted paths (stored in “Partial” array) is much worse than the best path that we have already found.

At each iteration of the algorithm we sort all of the partial paths by their current error and then explore the path with the smallest error (see while loop in Algorithm 1). This path is advanced until it reaches a feature that requires the pin operation. At this point there may be multiple ways to advance the path so several new paths may be created. Each new path’s error is adjusted to reflect the stretching or compressing of the distance traveled from the last pinned landmark. The algorithm proceeds to the next iteration and follows the path with the smallest error, and thus, the first path to finish cannot be worse than any other path. The value of parameter δ (greater than one) determines the number of finished paths we can obtain. This allows us to observe the top n paths. However, its value does not affect the selection of the best path, since the first solution path found is always the best. We set δ to be 1.1 when only interested in finding the best path.

Our algorithm uses maximum speed limits from OSM to determine whether a candidate path is possible. We have attempted several methods to apply speed limits to our algorithm such as segmenting out steady speed intervals for speed limit checking, separating highway and non-highway segments, and segmenting transient speed intervals for way entrance speed checking. However, they did not work out well experimentally. In the end, a simple solution of comparing vehicular speed to the maximum speed limit worked well. If the speed exceeds the maximum limit by more than 20 mph, the path is no longer considered possible. After trying different upper bounds, we found that speed limit plus 20 mph provides the best result overall in our testing stage. This seemed to be the upper bound

```
Input: A path-P, speed samples-S, and the current index into the speed samples-i
while i < S.length do
    // Match turns at intersections to slower speeds.
    if P at intersection then
        P' ⟵ ∅
        foreach Edge ∈intersection (P) do
            if S[i].speed ≤ maxSpeed (P, Edge) then
                Pin (P)
                join (P', P ∪ Edge)
            end
            else
                Fore ⟵ compressA (P)
                Back ⟵ stretchA (P)
                join (P', Fore)
                join (P', Back)
            end
        end
        return P'
    end
    // Match 0 speeds to intersections.
    if S[i].speed ≈ 0 then
        while S[i].speed ≈ 0 AND
        i < S.length do
            ++i
        end
        if P at intersection then
            Pin (P)
            return P
        end
        else
            Fore ⟵ compressB (P)
            Back ⟵ stretchB (P)
            return {Fore, Back }
        end
    end
    // Process normally
    if speed higher than way max speed + 20 then
        drop the current path
    end
    else
        update total traveled distance
    end
    ++i
end
```

Algorithm 2: Pseudocode for the *gotoBranch* function used in the elastic pathing algorithm. This code advances a single path until it reaches discrepancy between the speed trace and road. When this happens the path is corrected with compression or stretching and the path is *pinned* at what we call a landmark and the path's error is recomputed. Multiple possible paths may be returned at intersections.

that drivers in our datasets were willing to exceed the speed limit on the roads traveled.

The most complicated operation done in the *ElasticPathing* algorithm is the *gotoBranch* function (pseudocode appears in Algorithm 2). The *gotoBranch* function first verifies that the current speed does not exceed the speed limit of the road by more than 20 mph. It then advances the path until it comes to a feature on the map, finds every possible path branching from that feature, and returns those new paths. There are two types of feature matching we consider: a zero speed for an intersection and a slow enough speed for a turn.

When a possible path calculated by the algorithm reaches an intersection it explores every possible direction. If a vehicle on the possible path is going at a speed that can move along the curve in the road then a landmark is set at that location with the *Pin* function. If the curve in the road is too great for the current speed, we can rewind the speed cursor by a few samples to find a past speed sample that is slow enough to make the turn. This is equivalent to compressing the traveled distance. The amount of compressed distance can be calculated and added to the path error. This is done through our *compressA* function. We can also advance the speed cursor to find a speed sample that is slow enough to make the turn. Our *stretchA* function is used in this situation. Therefore, when there is a mismatch between speed samples and road data, there are two ways to resolve it, and thus, two new possible paths.

A similar situation occurs when the speed data indicates that the vehicle has come to a stop. If the path is already at an intersection then the landmark is set with the *Pin* function. Otherwise the two solutions are found with the *compressB* and *stretchB* functions. These two functions do not require rewinding or advancing the speed cursor. They directly compare the calculated distance from speed data to the road distance in the map and compute the difference (or error). They are still based on the concept of compressing and stretching the distance but have different implementation compared with *compressA* and *stretchA* functions mentioned above.

After any landmarks are set, the amount of stretching or compressing from the last landmark increases the error of each path. This means that an existing path may have less error than the current paths and should be explored instead. Therefore, all possible branches are returned to the elastic pathing function and the paths are placed in sorted order by their error. The pathing then continues with the new best path.

There are several details that we do not show in the pseudo-code. For instance, when we check if we are at an intersection, we allow space for the number of lanes in the road and offset for another car in front of our vehicle. Determining the minimum safe turn radius for a speed value is done by assuming the maximum allowed incline in the road (8%) (Roess, Prassas, & McShane, 2011). The equation that we use comes from Roess et al. (Roess et al., 2011), and it is given as:

$$r\_safe = \frac{speed^2}{(15 \times (0.01 \times \text{DEFAULT ELEVATION} + \text{friction}))} \tag{3.1}$$

In the safe turn radius equation, friction is the dry coefficient of sliding side friction for tires. Maximum default elevation is assumed to be 8% which is the maximum allowed for areas where ice can form on the road. Setting maximum elevation allows the minimum radius to make a safe turn, allowing more paths than rejecting them. The minimum safe turn radius for a speed is then compared with the actual turn radius of the road which can be determined based upon the number of lanes in the road and typical lane widths (AASHTO, 2011). If the actual turn radius is smaller than the safe turn radius, the vehicle cannot make the turn at the given speed.

3.4 Driving Data

To test how well our algorithm can reconstruct a driving route from a known starting point and a speed trace, we applied driving traces with different drivers and various driving environments. We required both a ground truth of GPS data and a sample set of speed data. This data was collected with the approach that the insurance companies use, by connecting a device to the On-Board Diagnostics standard (OBD-II) connector. To log the required data we used two devices: a GPS-enabled smartphone, and a Bluetooth-enabled ODB-II device. We recorded the timestamped speed data and the corresponding GPS positions simultaneously. We also obtained a much higher volume of GPS-only data, from which we reconstructed the speed traces.

To determine the appropriate sampling rate for data collection, we referred to how insurance companies collect speed data. Although companies do not directly disclose their data sampling rate on their websites, Allstate (Allstate, 2013) stated that “hard braking events are recorded when your vehicle decelerates more than 8 mph in one second (11.7 ft/s^2); extreme braking events are recorded when your vehicle decelerates more than 10 mph in one second (14.6 ft/s^2).” We know that the sampling rate must be faster than the duration of the feature the insurance company wishes to observe. In the United States, federal vehicular safety regulations mandate that vehicles traveling at 20 mph must be able to brake to a full stop in 20 feet at a deceleration rate of 21 feet/second^2 (U.S. Department of Transportation, 2013).These events that should be detected occur over a time interval of one second ($\tau = 1$). Using the Nyquist sampling theorem, we know that they must sample at twice this rate (two samples per second) to detect these features. To ensure the lower bound of sampling rates any company may use, we used sampling rate of one sample per second. This is much slower than what insurance companies use. Slower sampling rate means less information due to fewer samples, thus it is harder to estimate routes.

Using the speedometer data from the OBD-II device was straightforward because raw speed data was immediately available. However, some driving traces were only available as GPS traces. This data required additional processing to obtain speed values from latitude and longitude pairs. This was a two step process: (1) we applied the haversine formula (Sinnott, 1984) to approximate distances from the raw GPS coordinates; (2) we divided this value by the time interval to get an instantaneous speed value for each time interval. Each speed value was then given a timestamp and was converted into the same format as the speedometer traces. We note that we did not see differences in the accuracy of our algorithm between the two approaches of data collection. We also verified that the GPS data approximates actual speedometer data very well, and thus, does not affect our results.

After obtaining driving data, we slip them into driving traces as units of inputs to our algorithm. Our algorithm assumes that each driving trace does not have intermediate stops, meaning that all stops between a start location and a destination can only be caused by road conditions (e.g. traffic light, stop sign, or turns). We split the trip when the driver location does not change for more than 5 minutes, since stops due to traffic conditions are usually much less than 5 minutes.

We used two datasets to evaluate how our algorithm works with the real-world driving

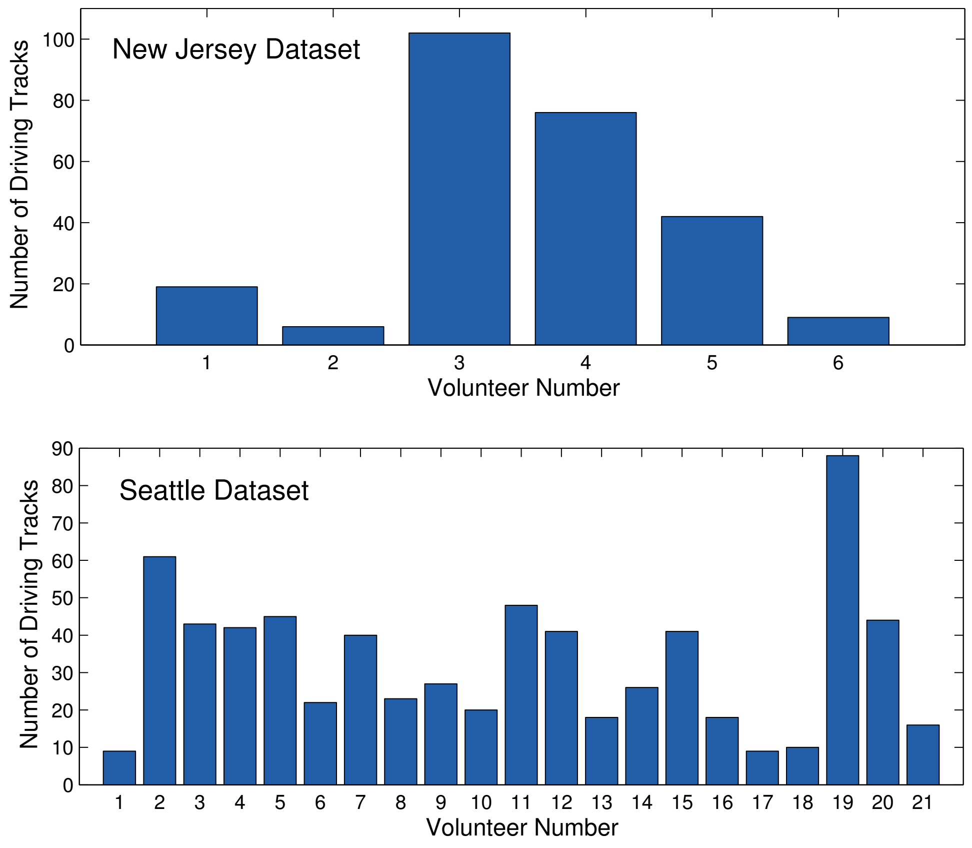

Figure 3.1: The number of driving traces for each volunteer in the New Jersey dataset (top figure) and the Seattle dataset (bottom figure).

traces: the central New Jersey dataset representing suburban areas and the Seattle city dataset representing urban areas with nearly one thousand testing traces in total.

3.4.1 Central New Jersey Dataset

We had six volunteers collecting data. Four volunteers collected GPS-only data over a period of three months, and two volunteers gathered both GPS and speedometer data with an OBD-II device over a period of one month. Examples of vehicles that were used in New Jersey dataset collection were small sedans, sports utility vehicles, and a pickup truck. The sampling rate for both the GPS and the speedometer was one second. Figure 3.1 shows counts of individual driver traces collected. There were 254 data traces that we used for pathing with nearly 1250 miles (nearly 2012 km) in total, with a median trip length of 4.65 miles (7.5 km), minimum trip distance of 0.38 miles (0.62 km), and maximum trip length of 9.96 miles (16.0 km). These traces comprise 46 unique destinations, with more than half of these destinations visited multiple times by individual drivers. 28 locations were visited more than once during data collection and ten locations were visited more than five times. The total driving time for 254 data traces was about 77 hours with an average driving time of 18 minutes for average driving distance of 4.9 miles. The driving areas were mostly not dense urban areas, but residential, suburban and commercial areas connected by highways.

3.4.2 Seattle Dataset

To test our algorithm performance on denser urban areas, we used an external GPS dataset from Microsoft Research (MSR) (Krumm & Brush, 2009). The MSR GPS data was collected by 21 volunteers carrying a GPS logger for about eight weeks in the fall of 2009 in the region of Seattle, Washington. We had looked through several datasets available for access from various research centers and selected this particular dataset for its large size and great potential of extracting valuable driving traces for our testing. However, not all the traces were driving traces, since volunteers also carried around the GPS logger while they were walking or traveling by train or plane. This required the design of a trace filter to pick out all the driving traces. We extracted all the GPS traces that matched the following criteria: the trace consisted of driving in Seattle, and lasted for at least three minutes. This was done in order to have a reasonable distance with all traces and exclude trivial examples.

Another issue we needed to consider with this dataset was how to handle the loss of GPS signal while driving. For example, a car driving in a tunnel may lose GPS signal temporally.

To ensure the accuracy of the speed calculation using GPS trajectories, two sequential GPS readings should not be separated by more than five seconds. If this happens, they should be considered as two different driving traces. The possibility of losing signal is the only disadvantage of using a GPS device instead of using a speedometer. The dataset from New Jersey did not have this issue.

After trace filtering, 691 traces were extracted with total driving distance of 1778 miles. Figure 3.1 shows the counts of traces per participant. The mean driving distance per trace was 2.6 miles with minimum 0.59 miles and maximum 19.9 miles. The average driving time per trace was 11 minutes. Comparing with our own dataset collected by six volunteers, the Seattle dataset has relatively shorter average driving distance. Since it is an urban area, the participants' average driving distance per trip may be shorter than in suburban areas. We had no control of or information on the vehicles used since the dataset was from MSR and was collected independently to our research efforts.

3.5 Results

We ran our algorithm and generated results for both the New Jersey and the Seattle datasets. A Ruby implementation of the elastic pathing algorithm processed all of the New Jersey traces (254 traces) in under 30 minutes on a two-core 2.2 GHz machine, with an average running time of less than ten seconds per trace. For the Seattle dataset (691 traces), it took about one hour with average running time of less than ten second per trace as well. Even without an implementation in a high-performance language, the algorithm is able to process traces far faster than they are generated. Since data analysis could essentially be done in real-time as a vehicle's speed trace is being collected, the limiting factor on the time delay between data recording and path prediction is likely the time overhead of collecting, transmitting, and preprocessing (e.g. matching a trace to a specific driver and their known locations) speed traces. This means that it may be entirely feasible to do near real-time tracking of drivers with just speed data, although the tracking will not be 100% accurate.

3.5.1 New Jersey Dataset Result

For the New Jersey dataset, our algorithm predicted 14% of traces with destination error less than 250 meters and about 24% of traces with destination error less than 500 meters. Table 3.1 shows the distribution of the number of traces over endpoint error intervals. Figure 3.2 shows the percentage of individual driving traces with destination error less than 250 meters and 500 meters for each volunteer. The algorithm performance varies on different individual drivers. We refer to the six volunteers as P1 to P6. For P2, our algorithm failed to find any match with endpoint error within 2 miles. In contrast, our algorithm prediction did relatively well for participant P4's driving traces. 20 out of 76 (26%) of traces had endpoint error less than 250 meters. 31 out of 76 (41%) of traces had endpoint error less than 500 meters. These results suggest that some driving styles are easier to localize than others.

3.5.2 Seattle Dataset Result

Our elastic pathing algorithm gave destination prediction error within 250 meters for 13% of traces in the Seattle dataset and within 500 meters for 26% of traces. Table 3.1 shows the number of traces within each endpoint prediction error interval and the corresponding percentage for the total of 691 traces. For 60% of traces, our algorithm gave the endpoint (or destination) prediction error of less than one mile. However, 22% of traces had endpoint

Endpoint Error		New Jersey Dataset		Seattle Dataset	
meters	miles	Num. of Traces	Percent	Num. of Traces	Percent
0-250	0-0.16	36	14%	90	13%
251-500	0.16-0.31	24	9.5%	92	13%
501-750	0.31-0.47	14	5.5%	59	8.5%
751-1000	0.47-0.62	11	4.3%	59	8.5%
1001-1250	0.62-0.78	11	4.3%	49	7.1%
1251-1500	0.78-0.93	35	14%	44	6.4%
1501-1750	0.93-1.09	2	0.79%	25	3.6%
1751-2000	1.09-1.24	6	2.4%	21	3.0%
2000-2250	1.24-1.40	5	2.0%	29	4.2%
2251-2500	1.40-1.55	6	2.4%	21	3.0%
2501-2750	1.55-1.71	8	3.2%	25	3.6%
2751-3000	1.71-1.86	4	1.6%	17	2.5%
3001-3250	1.86-2.02	6	2.4%	12	1.7%
Greater than 3251	Greater than 2.02	86	34%	148	21%
Total traces		254	100%	691	100%

Table 3.1: Results for both central New Jersey dataset and Seattle dataset from our elastic pathing algorithm. Table shows the number of traces having endpoint (or destination) error ranging from 0 to greater than 3.25 km, separated into 14 error intervals as shown in rows. First two columns give endpoint error representations in meters and miles.

error more than 2 miles. For those traces having large endpoint error, their speed patterns were very noisy. This could be due to heavy traffic which forced cars to stop frequently.

Figure 3.2 shows the percentage of individual driving traces with endpoint error less than 250 meters and 500 meters for each volunteer. Volunteers are referred as P1 to P21 in the following description. For P2, about 26% of the traces had endpoint error less than 250 meters. However, for P1, no trace had prediction error less than 250 meters. P20 had about 5% traces having prediction error less than 250 meters. Besides different driving habits, the percentage difference between P1 and P2 may also due to the difference in number of total traces (see Figure 3.1). From 21 volunteers, P2's driving traces produced the best results with 26% traces having endpoint error less than 250 meters and 48% traces having endpoint error less than 500 meters, while P1 driving traces had the worst result with 0% having endpoint error less than 250 meters and 22% having endpoint error less than 500 meters.

3.5.3 Analysis

The relative accuracy of our approach does not go down with trip distance through our testing on both datasets. Errors were not highly correlated with the path distance. However, in general, shorter trips had smaller variance in error than most of the longer trips. The percent of predicted endpoints within 250 meters of the actual endpoint also does not decrease with distance in our dataset, with trips as long as 10.5 miles still having endpoints correctly predicted to within 250 meters. Our approach clearly does much better than random guessing and almost always correctly identifies at least the general direction of travel. The direction of travel is not a very serious privacy concern though, so we will use another

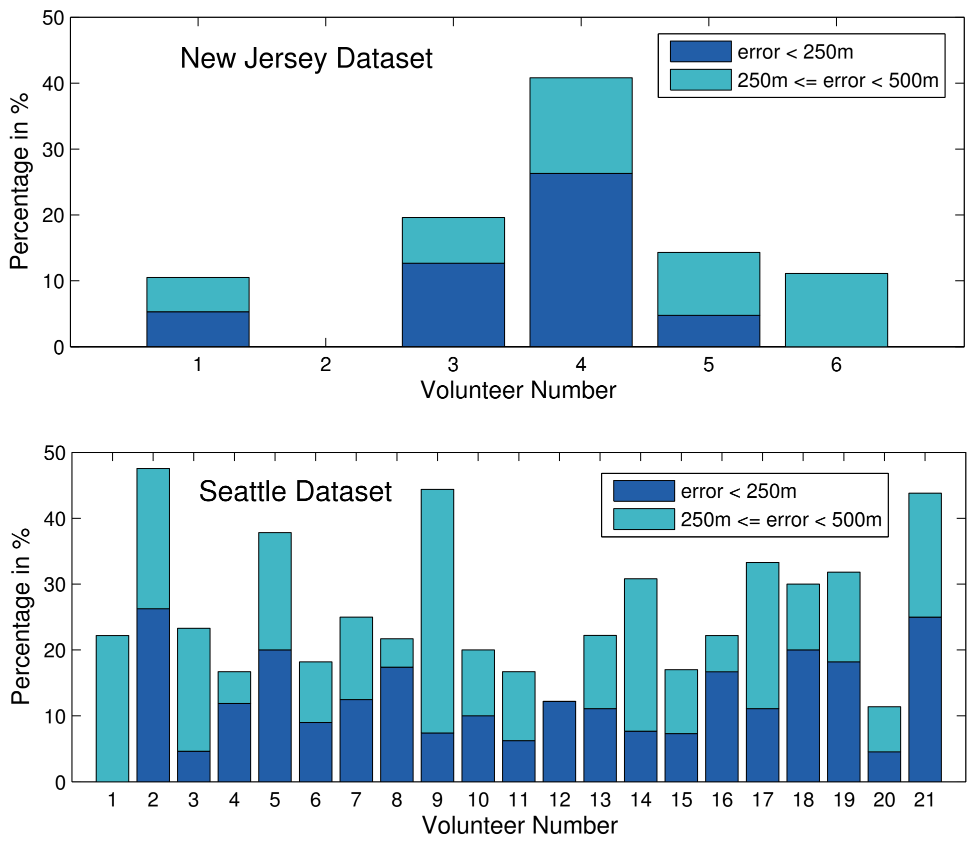

Figure 3.2: Percentage of individual driving traces that have endpoint error less than 250 meters and 500 meters. For NJ volunteer 2, there is no bar, which means 0% of the traces had the corresponding error.

filtering step to screen out noise from incorrectly predicted points.

Our algorithm's performance is dependent on the individual driving habit. Given the same driving location, some individuals may have average speed much higher than others. We assumed that drivers followed the speed limits to some extent to help prune possible paths. This assumption is somewhat incorrect as many drivers speed on highways. However, most drivers obey local (residential) speed limits because of unpredictable driving environments (e.g. children running in the streets). Additionally, local enforcement policies encourage adherence to said speed limits as traffic rules are more strongly enforced in residential areas and fines in these areas are higher. The speed limits in these areas are used as an admissibility criteria for possible paths (e.g. if the driver is traveling at 55 mph (88.5 km/h), it is unlikely that they are on a 25 mph (40.2 km/h) residential road).

We can further take advantage of multiple available traces for each driver to screen out points that may be noise. We note that erroneous predictions are unlikely to repeatedly find the same wrong location and instead usually make mistakes in different locations. Thus, we can use the frequency that locations are predicted as a way to remove poor predictions and focus only on common destinations.

The pathing algorithm is able to do more than just identify the endpoint of a trip – it also eliminates a large number of locations where the trip could not have ended. This is also a serious privacy concern because it allows an antagonist to identify changes in regular routines very easily. For example, if a person usually goes home after work but the predicted path goes in a different direction then it is highly likely the person is breaking their routine.

When the algorithm has enough information and a route is very unique, for instance when the spacing between intersections on the road traveled does not match the spacing in any other road, the algorithm can do very well. This is illustrated in Figure 3.3. However, this case is not normal, and at the very end when the algorithm chooses the correct left turn, it may just as well have chosen to turn right or gone straight. However, even if the algorithm made the wrong choice it would still be within 100 meters of the actual destination.

We believe the probability of error is closely tied to a few factors:

- Homogeneity of roads. If every road seems the same (same speed, similar intersection intervals, especially as in areas built in a grid), then they cannot be distinguished.

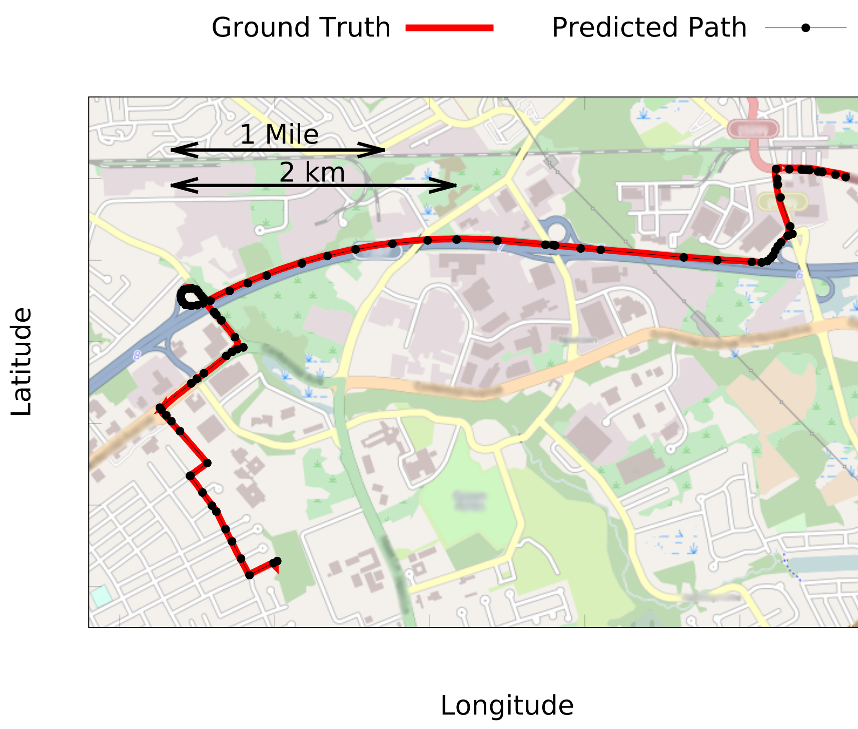

Figure 3.3: The best possible case, a path from a grocery store to a home with no error. The ground truth is indicated with a solid line while the nodes along the predicted path are shown as dots. In this case the predicted and actual paths match perfectly.

- Traces with only slow speeds. Without high speeds to rule out turns and constrain paths to a few major roads, the correct path is indistinguishable from any other.

- Unpredictable stops, caused by traffic, construction, etc.

The largest barrier to improved performance is distinguishing one road from another. If two roads have similar features (e.g. same speed, similarly spaced intersections) then there is little possibilities for the algorithm to distinguish between the two. This is also affected by the traffic pattern of the trip; if a vehicle stops at every intersection because of red lights, then there is a great deal of information about the spacing of those intersections. If, however, the vehicle stops at no lights, then there is no information about the spacing of intersection on the current road. With more prior information we may be able to do better.

For example, if we knew the average waiting time at different lights then the waiting time at an intersection might distinguish one road from another even if a vehicle only stops at one or two lights.

This additional information, such as the average wait time at certain lights, or the average traffic speeds of different roads during different times of day, may already be available to insurance companies and this information is not difficult to gather. If the algorithm had this information available, it could score different turning choices with higher accuracy, leading to improved results.

CHAPTER 4

ENHANCED ELASTIC PATHING ALGORITHM WITH IMPROVED ACCURACY

4.1 Chapter Overview

In this section, we describe the enhanced version of the Elastic Pathing algorithm with improved accuracy. First, we present the new modifications and approaches made in this new version to enhance our algorithm. Next, we show the visualization tool that we built using Google Maps APIs to visualize how the algorithm tracks drivers. Then, we describe the new dataset that we collected to analyze how our algorithm performs for various drivers and driving behaviors. At the end, we present the improved results and analyze how different aspects of the speed data and our algorithm can affect the prediction accuracy.

4.2 Optimization for Elastic Pathing Algorithm

In this section, we describe the key improvements and algorithm enhancements that we made over the initial work (Gao et al., 2014).

Applying a New Model for Turn Radius Determination: The turn radius of the intersection determines the maximum driving speed at which a vehicle could make the turn. This provides an estimate of the feasibility that a driver can be in the turning location given the speed in the corresponding time instance. Thus a good model to determine the actual turn radius is essential to the algorithm. Our previous attempt was to assume a sharp turn

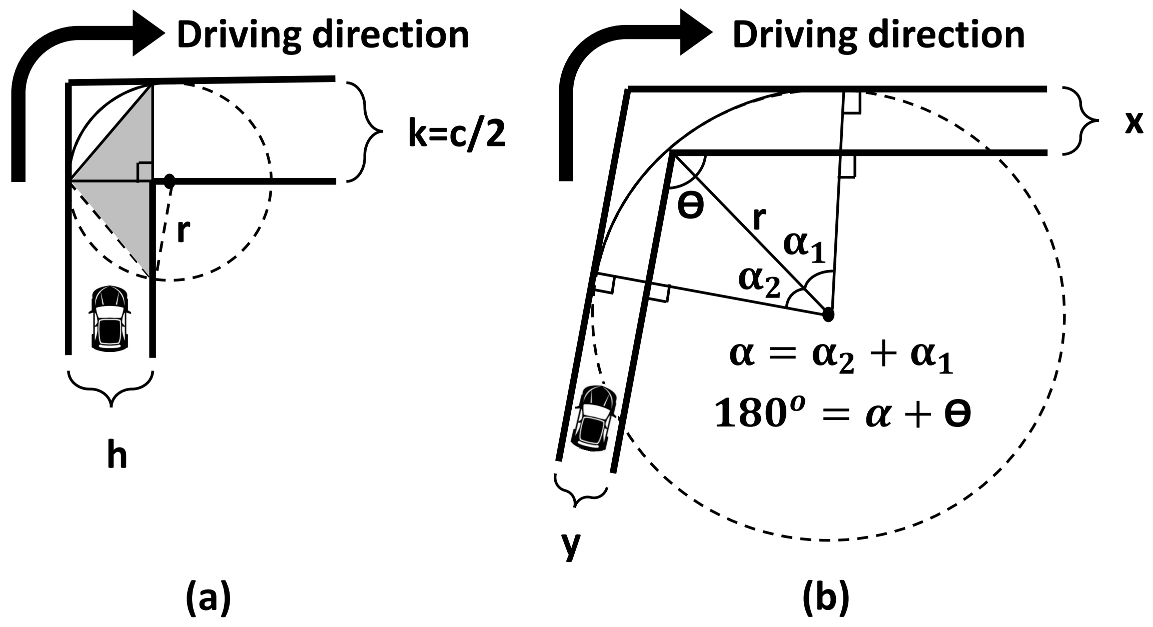

Figure 4.1: (a) The figure in the left presents our initial approach to estimate the turn radius which is set to be the radius of the circle inscribing the triangle (shaded gray in the figure). (b) The right figure shows our enhanced turn radius calculation method. The turning angle does not have to be 90 degrees and it has the maximum turn radius among all possible turns made within the path boundary.

by drawing a circle to inscribe a triangle. Then, the radius of the circle was set to be the turn radius (see Figure 4.1a): $r = h/2 + c^2/(8h)$ (Gao et al., 2014). Although a simple sharp turn happens frequently in most four-way intersections, we realized that a better model is necessary to account for different types of intersections (e.g. three-way intersections, intersections with turning angles not equal to 90 degrees). In addition, the resulting radius from the old model was not the maximum radius allowed given the geometric structure. This would cause the calculated maximum speed to be smaller than the actual possible speed, causing the algorithm to falsely reject candidate paths that may have been correct.

We used a new approach to model this turning event. Instead of having a sharp turn in the intersection, one can gradually turn from the outer side of the path, passing the intersection corner, and arriving to the outer edge of the other path (see Figure 4.1b). This model gives the largest possible turning radius. In this model, the turning angle α does not have to be 90 degrees. The beginning path width x and the ending path width y can also be different. To solve for the turn radius r, we just need to apply some simple trigonometric functions and obtain the following equation:

$$\cos\alpha = \frac{r-x}{r} \cdot \frac{r-y}{r} - \frac{\sqrt{r^2-(r-x)^2}}{r} \cdot \frac{\sqrt{r^2-(r-y)^2}}{r} \tag{4.1}$$

Thus the turn radius r can be solved (e.g. there are two solutions, but we need the solution with larger value in this case), which is:

$$r = \frac{x+y+\sqrt{2xy(1+\cos\alpha)}}{1-\cos\alpha} \tag{4.2}$$

This turn radius is then compared with the minimum safe turn radius to determine whether the speed is too fast to make the turn.

Routing Method as an Optional Customization: OpenStreetMap (OpenStreetMap, 2016) provides a routing (or navigating) feature to help users moving from one place to another. The ruby version of the OSM routing is implemented in Mormon (Mormon, 2016) based on pyroutelib2 (OpenstreetmapWiki, 2015). OSM routing, based on A* search algorithm with weights, finds the shortest path between the two locations (Mormon, 2016). We applied OSM routing as it is open source and provides free unlimited amount of usage on a given amount of time period.

The idea is to run our main elastic pathing search function first and use routing information to refine the score of our top candidate paths. After the main search function, there is a list of candidate paths ranked by their error. We only consider the top 10 candidate paths to guarantee the quality of the selected candidate paths and avoid run-time overhead. Each candidate path has an estimated route, estimated destination, and an error. By comparing the length of a candidate path to the length of a navigated path, we can further evaluate the candidate path: the smaller the difference between lengths, the better the candidate path is.

This is under the assumption that people usually take an efficient driving route which should have a good match with the navigated route when comparing trip distance. However, this may not be always true. For example, one may go with a much longer route to avoid heavy traffic. The effect on estimation accuracy of adding this method can only be tested with real-world datasets (see Section 5.5 for analysis).

We constructed our routing method as an optional function that could be easily switched on or off. To explain how we modified the error metric to consider the routing distance, we introduce the following parameters:

- **CalcDist:** This is the distance calculated from the speed and time values in the speed trace as defined previously.

- **RoutingDist:** This is the routing distance, shortest distance in this case, obtained using OSM routing method.

- **Error:** As defined previously, this is the difference between calculated distance and predicted distance, accumulated while compressing and expanding the path.

- **MaxError:** This is the maximum **Error** among the top 10 candidate paths for a driving trace.

- **CombScore:** This is the final combined score used to determine how good a candidate path is – larger score (ranging from 0 to 1) indicates better path.

To obtain the final combined score, we need to properly weight the error from the two factors: the difference between calculated distance and predicted distance, and the difference between calculated distance and routing distance.

There are three cases based on the value of the routing distance. The first case happens when the routing distance is smaller than the calculated distance. Since the routing distance is based on finding a shortest path, this case is expected to happen for some traces. We apply the following formula to calculate the combined score:

$$CombScore = \beta \cdot \frac{MaxError - Error}{MaxError} + \omega \cdot \frac{RoutingDist}{CalcDist} \tag{4.3}$$

where β and ω are weight coefficients for these two fractions such that $\beta \in [0,1]$, $\omega \in [0,1]$, and $\beta + \omega = 1$. The first fraction is simply the same as $1 - (Error/MaxError)$. $Error/MaxError$ gives how bad the trace error is in range of $[0,1]$ (the larger the value, the worse the trace). As we want the fraction to reflect how good a trace is (the larger the value, the better the trace), we use $1 - (Error/MaxError)$ to convert to what we need. The second fraction we use for routing score is $RoutingDist/CalDist$. As $RoutingDist$ is smaller than $CalDist$, this gives value ranging in [0,1] with larger score representing better routing match.

The second case is when routing distance is larger or equal to the calculated distance. Theoretically, the routing distance should not be larger than calculated distance as routing is the shortest one. This case may still happen considering the calculation rounding error and the possibility of map data not exactly matching the actual road length. In this case, we use the following formula:

$$CombScore = \beta \cdot \frac{MaxError - Error}{MaxError} + \omega \cdot Ratio_2 \tag{4.4}$$

where

$$Ratio_2 = 1 - \frac{RoutingDist - CalcDist}{CalcDist} \tag{4.5}$$

$(RoutingDist - CalcDist)/CalcDist$ measures the proportion of the difference to the calculated trip distance: $CalcDist$. We restricted this range to also be in [0,1] (the larger the value, the worse the trace). Note that if the RoutingDist is more than twice as long as CalcDist, we set it to equal to twice of CalcDist as this is the worst case. Then, we similarly use $1 - (RoutingDist - CalcDist)/CalcDist$ to convert to what we need: the larger the score, the better the trace.

The reason we use this formula ($Ratio_2$ formula) is to ensure we rate both conditions (routing distance being larger and shorter) in the same way. For example, if the $CalcDist$ is equal to 1 mile, we need to make sure the rating is the same when $RoutingDist$ is equal to 0.8 miles and when $RoutingDist$ is equal to 1.2 miles, as they are both 0.2 miles difference to the $CalcDist$. Inevitably, this requires cutting off the $RoutingDist$ at $2 * CalcDist$. An alternative approach would be simply using $CalcDist/RoutingDist$ instead of our $Ratio_2$ formula, but such metric will have distorted rating and lose the consistency between these two conditions.

The third case is when routing distance cannot be found. This case happens when the

OSM fails to find any route between two given points. However, this may not mean that there is no route connecting two points based on our testing. Sometimes, OSM API just failed to find the routes even for regular nodes. Thus we leave out the routing factor from our formula in this case:

$$CombScore = \frac{MaxError - Error}{MaxError} \tag{4.6}$$

In general, this combined score metric works for different types of routing methods (e.g. shortest path, fastest path, path with light traffic), as it considers the matching with two distances in general. In this paper, we only use it for the OSM routing of shortest path due to the limitation of the OSM routing API.

Other Enhancements: We re-implemented the landmark setting function to match with the new model for turn radius. We also adjusted the way we duplicate landmarks for newly generated candidate paths. We did minor modifications on the error accumulation functions to rematch beginning and ending points while calculating error based on speed traces. Finally, we corrected some minor issues while moving the time index in the speed trace to eliminate any offset it may produce during the path expanding and compressing.

Space and Time Complexity: Our elastic pathing algorithm is essentially a search algorithm optimized with a priority search queue. Different from depth-first search and breadth-first search, our algorithm prioritize the search on paths that are most likely to be the solution path. Therefore, our algorithm does not require searching through the whole path space. The first path found is the best path, and the first n path found are the top n ranked paths.

To estimate the space and time complexity, we can assume that there are N paths within the range that can be reached by the given driving distance. There are K nodes for each path (recall that nodes are locations representing a path). There are M samples in the given speed trace. M is much larger than K. The worst case happens when all paths are equally bad matches. The algorithm has to advance and shift among all the paths, resulting exploring all the paths within the range. The time complexity is M$\times$N, which can be noted as O(MN). The algorithm needs to at least store all the paths and the speed trace, so the space complexity in this case needs to be M+K$\times$N. The best case happens when there is only one distinct path matching well with the given speed trace. Then, the time complexity is about M, or O(M). The space complexity is about M+K. The average case time and space complexity of our algorithm depends on the actual driving environment or the path structure. With our datasets, fewer than 100 paths are explored in average based on the examination of our candidate path array list.

4.2.1 Visualization Tool

We built a visualization tool to show how our algorithm matches a speed trace to different paths along the map. The visualization interface was implemented using HTML5 and JavaScript for its convenience to host either locally or on the web. It takes an input file generated from our elastic pathing algorithm recording how each decision was made on each step. Then, using Google Maps APIs, we transferred the input data into the map with animation showing how our algorithm explores and selects paths (e.g. see Figure 4.2). The ground truth GPS data were plotted in the map with markers showing the starting location and the current position where the algorithm was exploring. Several top candidate paths

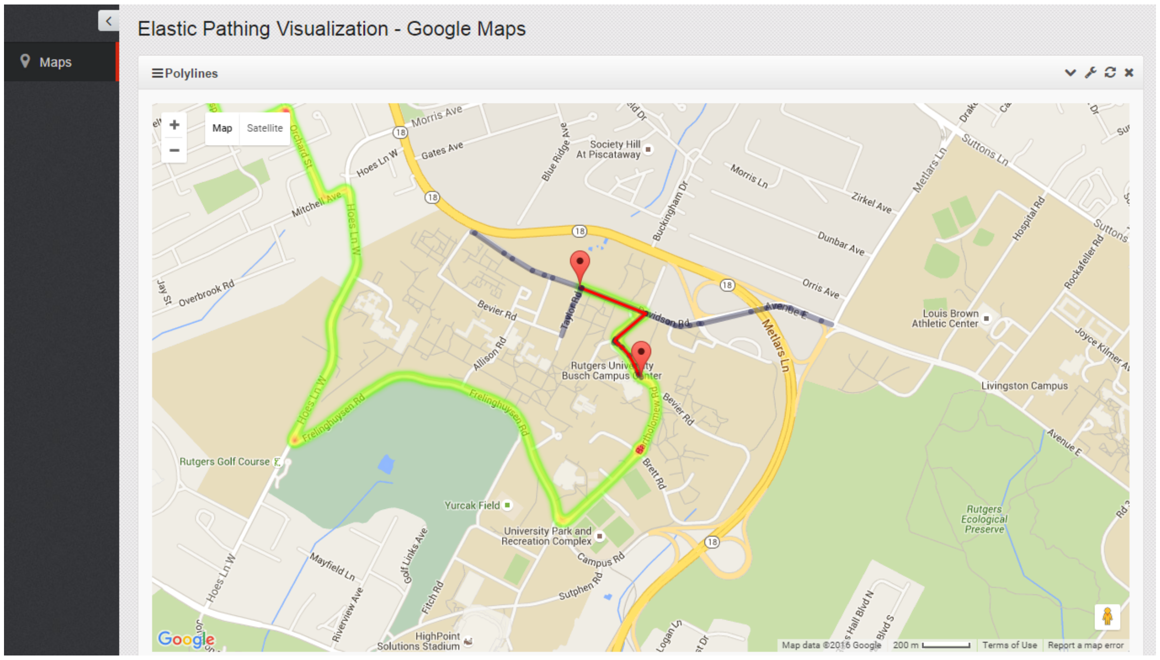
Elastic Pathing Visualization - Google Maps
Maps
Polylines
Map
Satellite
Rutgers University Busch Campus Center
Livingston Campus
Rutgers Golf Course
Yurcak Field
University Park and Recreation Complex
Louis Brown Athletic Center
Rutgers Ecological Preserve
HighPoint Solutions Stadium

Figure 4.2: The visualization tool for our elastic pathing algorithm: The upper marker indicates the starting location and the lower marker indicates the current location the algorithm was exploring. The yellowish green line shows the ground truth GPS points plotted with a heatmap (e.g. the more the color shifts towards red, the denser the GPS points are – thus the slower the car moves). Gray lines show several top candidate paths that the algorithm has explored. The red line shows the best candidate path up to the current time point.

that were explored upto the current time were also shown in the map.

The visualization of our algorithm is essential for the algorithm testing and debugging. Since our visualization tool takes separate input files instead of tying the implementation to our algorithm, it can be easily customized to similar applications that require the display of driving routes in the map.

4.3 New Driving Data

We used three datasets to evaluate how our algorithm works with the real-world driving traces: two datasets, the central New Jersey dataset representing suburban areas and the Seattle city dataset representing urban areas with nearly one thousand testing traces in total, to estimate the overall estimation accuracy of our algorithm, and one identical-route dataset to further explore how different driving behaviors may affect our algorithm performance. The central New Jersey dataset and the Seattle dataset were used to test our algorithm in our initial work (Gao et al., 2014). The description of these two datasets can be found in previous chapter of this thesis. The third data set, named the identical-route dataset, was newly added to assist further analysis on individual driving behavior in this part of the work.

4.3.1 Identical-Route Dataset

To investigate how individual difference may affect our algorithm accuracy, we recruited 30 participants (18 males and 12 females) to collect driving data. 20 of them were recruited in December 2014 and 10 were recruited in January and February of 2015. All of them were at least 18 years old and had a valid driving license. They were required to be an active driver during the past three months. Although our recruitment was done through online social media which was open to the general population, all our participants were affiliated with our university. Their educational background varies, however, most of them were undergraduate students while six had or were pursuing graduate degree. Each of them was compensated with a $50 visa gift card for completing the driving study. We labeled these participants as P1, P2, ..., P30.

We required all participants to drive in the same route twice. The route was for a round trip where participants needed to drive from point a to point b, and then return to point a from a different path. We split this round-trip route into the leaving route (denoted as route A) and the returning route (denoted as route B). Thus both route A and route B were traveled twice for each participant. These two routes were across different areas and have different driving environments: Route A consists of more traffic lights and intersections than route B. Figure 4.3 shows route A and route B in the map.

A 2002 Hyundai Accent car was used for all the driving traces in this dataset. Both the speed data and GPS trace were collected using a smartphone and an OBD-II connector. Before the driving data collection, participants were allowed to drive with the test car in the corresponding area to get familiar with the vehicle and the driving environment. This was done to familiarize the participants with the vehicle and road ways so that any adjustment or learning effects were eliminated from consideration. To avoid heavy traffic and ensure similar driving condition, all the driving traces were collected during late morning and early afternoon. The weather condition was also similar (e.g. either slightly cloudy or sunny) for all participants.

In this dataset we used a sampling rate of one sample per second. There are 60 driving traces in route A (4.8 miles) and 60 driving traces in route B (5.1 miles) by 30 different drivers. However, one participant (e.g. P19) once drove to a wrong route during the study, resulting only 59 valid driving traces for route A.

4.4 Results

We present results of the enhanced algorithm in this section. We start by presenting the overall accuracy of our algorithm when testing with two comprehensive datasets: New Jersey dataset (representing sub-urban areas) and Seattle dataset (representing urban areas). Then, we discuss the effect of applying OSM routing and road speed limits by comparing estimation accuracy. Next, we investigate how different drivers and driving behaviors may

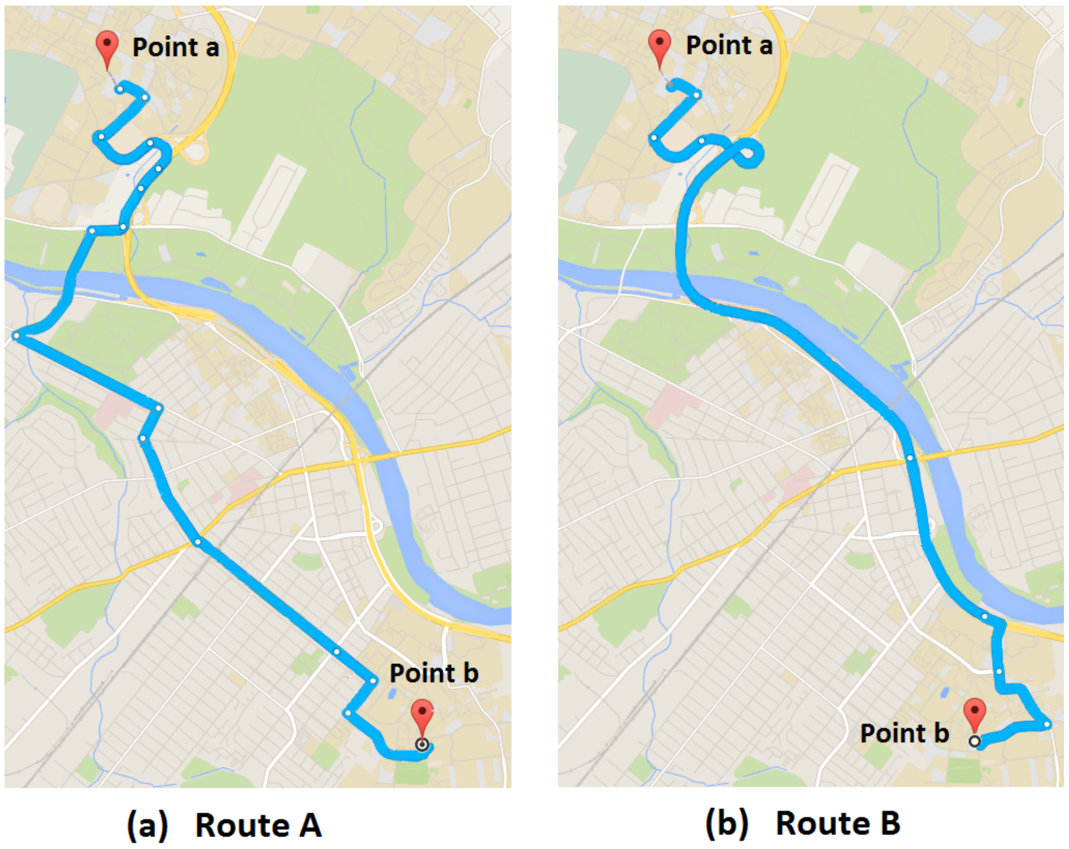

Figure 4.3: Route A shows the driving route from point a to point b, which comprises residential and commercial areas with some traffic lights. Route B shows the returning route from point b back to point a. This route consists of a long highway segment which connects local paths with a few traffic lights.

affect our algorithm by analyzing the result from our new dataset: identical-route dataset. Finally, we analyze the traces with poor estimation accuracy and point out major factors that may mislead our algorithm.

4.4.1 Overall Accuracy

The Ruby implementation of our algorithm processed all of the New Jersey traces (254 traces) in about one hour on a two-core 2.2 GHz machine. For the Seattle dataset (691 traces), it took about two hours. With nearly one thousand traces in total, about 80% of them had running time within just a couple of seconds per trace while remaining ones took

Destination Error		New Jersey		Seattle	
Meters	Miles	Trace Count	Percent	Trace Count	Percent
0-250	0-0.16	44	17%	107	15%
250-500	0.16-0.31	18	7.1%	83	12%
500-750	0.31-0.47	15	5.9%	82	12%
750-1000	0.47-0.62	12	4.7%	65	9.4%
1000-1250	0.62-0.78	7	2.8%	43	6.2%
1250-1500	0.78-0.93	14	5.5%	26	3.8%
1500-1750	0.93-1.09	7	2.8%	31	4.5%
1750-2000	1.09-1.24	14	5.5%	34	4.9%
2000-2250	1.24-1.40	8	3.2%	17	2.5%
2250-2500	1.40-1.55	10	3.9%	16	2.3%
2500-2750	1.55-1.71	3	1.2%	26	3.8%
2750-3000	1.71-1.86	4	1.6%	9	1.3%
3000-3250	1.86-2.02	3	1.2%	10	1.5%
>3250	>2.02	95	37%	142	21%
Total traces		254	100%	691	100%

Table 4.1: Results from the enhanced version of our elastic pathing algorithm for both the New Jersey and the Seattle dataset. Table shows the number of traces having destination error ranging from 0 to greater than 3.25 km, separated into 14 intervals as shown in rows. First two columns give destination error representations in meters and miles.

Destination Error (meters)	New Jersey		Seattle	
	Initial Version	Enhanced Version	Initial Version	Enhanced Version
<250	14%	17%	13%	15%
<500	24%	24%	26%	28%
<750	29%	30%	35%	39%
<1000	33%	35%	43%	49%
<1250	38%	38%	51%	55%

Table 4.2: Comparison of estimation accuracy between two versions of our algorithm. Table shows percentages of traces that has destination error within certain range for both New Jersey and Seattle dataset.

much longer and usually had much worse accuracy. The processing speed for traces can be done much faster than the rate at which those same traces are generated. This may enable real-time tracking of drivers with just speed data, however it will not be 100% accurate.

Roughly half of the traces can be estimated with destination error less than one mile for both the New Jersey dataset and the Seattle dataset (see Table 4.1). A significant percent of traces can be estimated with very high accuracy: 17% of traces with destination error less than 250 meters and 24% of traces with destination error less than 500 meters for the New Jersey dataset, and 16% and 28% of traces with destination error less than 250 meters and 500 meters respectively for the Seattle dataset. There is a noticeable improvement (especially for error less than 250 meters) in the estimation accuracy compared to our initial version (Gao et al., 2014) (see Table 4.2 for comparison). Overall, 150 traces having destination error within 250 meters, comparing to 125 traces in our initial work: there are 25 more traces (or 20 percent improvement of 125 traces) in this highly accurate estimation range. The detailed cumulative distribution functions (CDFs) of the destination error (within one mile) for New Jersey and Seattle datasets are shown in Figure 4.4 and Figure 4.5.

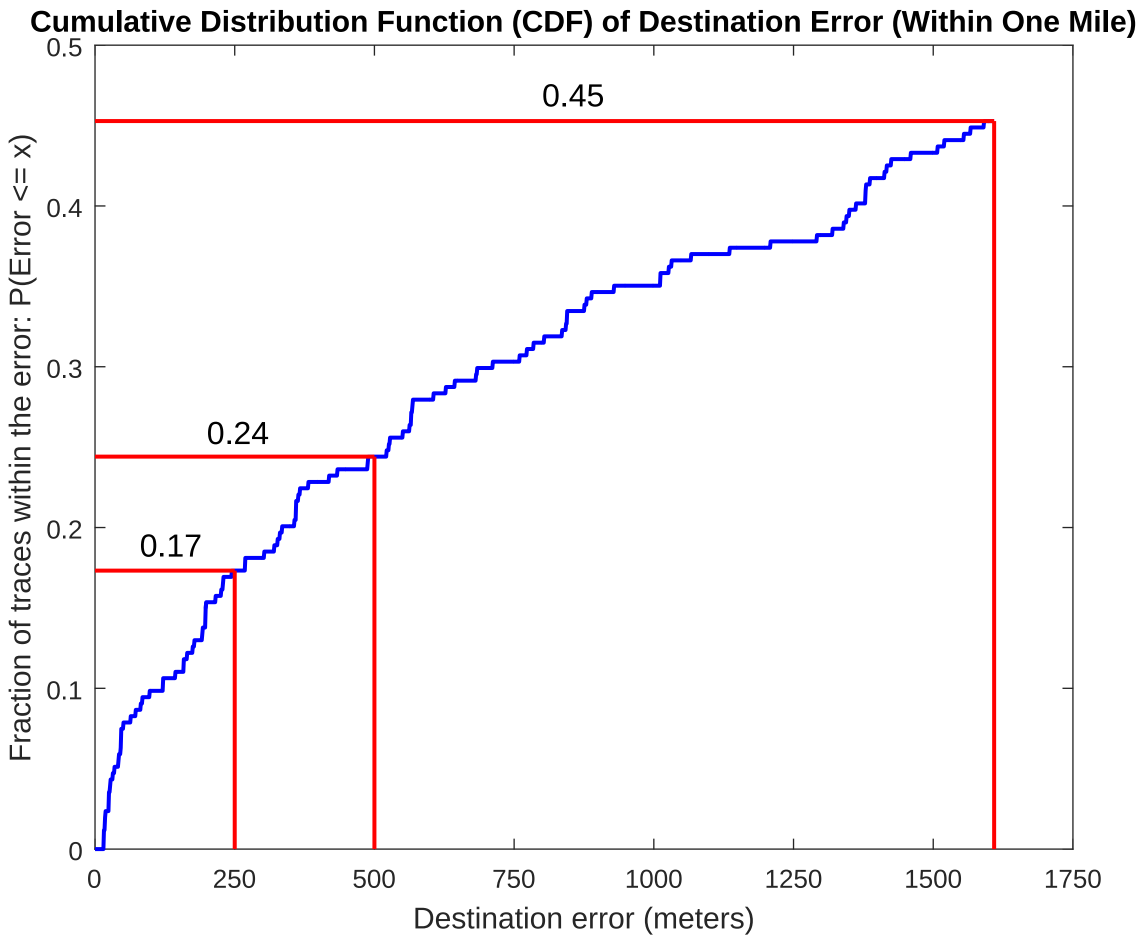

Figure 4.4: The cumulative distribution function curve for destination error within one mile from our New Jersey dataset.

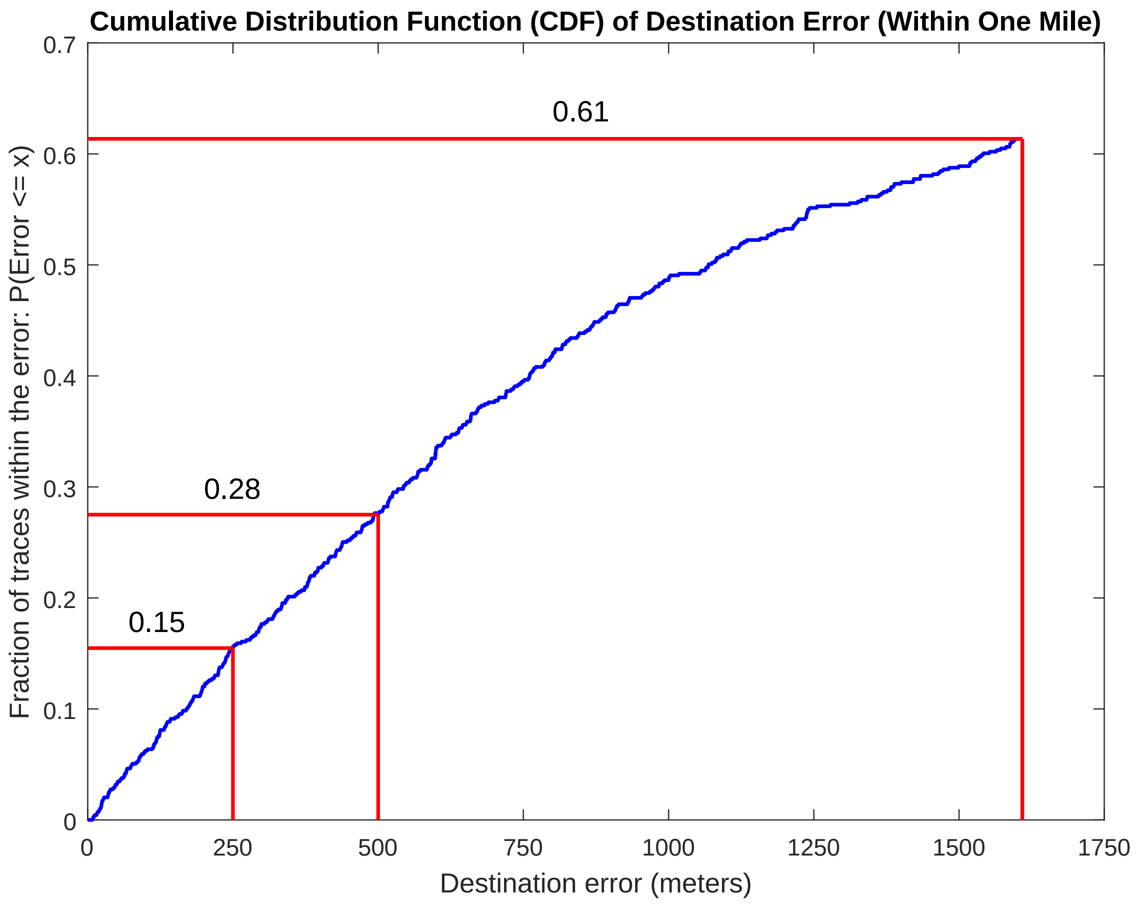

Figure 4.5: The cumulative distribution function curve for destination error within one mile from our Seattle dataset.

To provide additional context of our results from enhanced version of our elastic pathing algorithm, we analyze how estimation accuracy depends on the trip length. Figure 4.6 shows the distribution of trip lengths within each interval of estimation accuracy – from highly accurate interval (destination error less than 0.16 miles or 250 meters) to very rough estimation interval (error greater than 2 miles). In New Jersey dataset, for traces estimated with high accuracy (error less than 0.16 mile or 250 meters), trip length varies from 0.38 miles to 9.64 miles. This means that not only can our algorithm predict short traces with good accuracy, it can also predict very long traces with good accuracy. The average trip length for those with error less than 0.16 mile is about 4 miles. While the average trip length varies for traces in different error intervals, we did not see any clear dependence of our algorithm's performance to the trip length based on the figure. For Seattle dataset, the general variance of trip length is smaller than the one in New Jersey dataset, since Seattle traces have much shorter average trip length. Similarly, our algorithm can predict traces with both short and long trip lengths with good accuracy. While the average trip length for error larger than 2 miles is slightly longer than others, we did not find any consistent trend of trip length being larger in larger error interval. For example, traces in 0-0.16 mile interval has longer average trip length than those in 1.71-1.86 mile interval; the longest trace with 20 mile length has error between 0.31 and 0.47 mile.

4.4.2 Comparison with Naive Guessing

We implemented a naive guessing algorithm to provide a baseline accuracy to compare with elastic pathing algorithm. The naive guessing algorithm takes the map data, a starting location, and the speed data as inputs, but it does not utilize any logic about matching zero

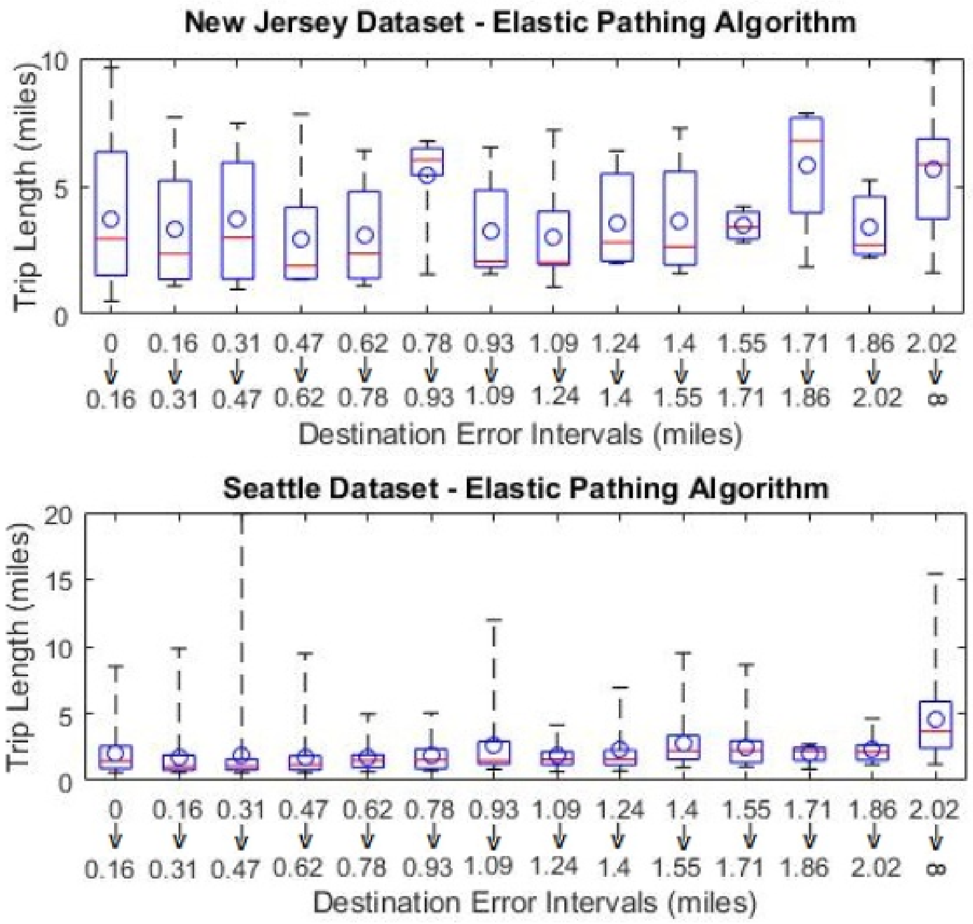

Figure 4.6: Box plots of estimation accuracy (or destination error) vs. driving trip length for the enhanced version of elastic pathing algorithm. The destination error is split into 14 intervals (e.g. interval 0-0.16 miles, interval 0.16-0.31 miles) with each interval equal 0.155 miles or (250 meters). Trip length distribution for each error interval is presented with a box-plot bar, showing: minimum, maximum, first quartile to third quartile range (blue box), median (red line), and mean (blue circle).

speeds to intersections or any scoring metric. From the starting location, the algorithm calculates the traveled distance from the speed data and selects a path randomly to advance. When encountering an intersection, the algorithm randomly selects a path to continue – each direction has an equal chance of selection.

We found that our elastic pathing algorithm achieves much higher estimation accuracy than naive guessing algorithm. Since naive guessing algorithm is based upon random selection, the solution path can be different each time the algorithm runs. We ran the guessing algorithm for ten times to obtain a rough range of its estimation accuracy. For error within 250 meters, naive guessing algorithm only estimates 0.87% (average value with min 0% and max 2.4%) of traces in New Jersey dataset and 4.6% (average value with min 4.1% and max 5.1%) of traces in Seattle dataset. This is much lower than our elastic pathing algorithm's accuracy: 17% for New Jersey dataset and 15% for Seattle dataset. We can see that naive guessing algorithm has higher accuracy in Seattle dataset than New Jersey dataset. This is due to Seattle dataset has much shorter average trip length than New Jersey dataset (2.6 miles vs. 4.65 miles).

4.4.3 Effect of Applying OSM Routing

We applied the routing (or navigating) method into our algorithm and compared its estimation accuracy with the version without it. The routing feature is provided by the OpenStreetMap (OSM) API. In order to obtain the combined score for each driving trace, we need to determine these constants: β which is the weight of the error accumulated by compressing and extending paths, and ω (equivalent to $1-\beta$) which is the weight of the routing error (see Equations 4.3 to 4.6). We enumerated β from 0 to 1 with 0.1 incremental steps.

One extreme case is when β equals 0. This means that only the routing error is considered to re-rank the top several candidate paths selected by our algorithm. Another extreme case is when β equals 1. This is when OSM routing error is not used at all. The error is contributed sorely by compressing and extending error.

We found that applying OSM routing drops the overall accuracy for both New Jersey and Seattle datasets. Varying the value of β from 0.1 to 0.9 does not have noticeable difference for the overall accuracy. Table 4.3 ("Add Routing") shows the estimation accuracy when both β and ω are set to 0.5. As shown in the table, our enhanced version without routing (happens when β=1) has better overall accuracy.

We found that OSM routing affects different drivers differently, especially for the New Jersey suburban dataset. There are six drivers in the New Jersey dataset. When β changes from 0.5 to 1 for P1 and P3, the portion of driving traces having error within 500 meters increases (P1: 10% to 20%, P3: 18% to 32%), meaning that our algorithm has higher estimation accuracy without using OSM routing. However, for P4 and P6, OSM routing helps increase the estimation accuracy for error less than 500 meters (P4: 22% to 28%, P6: 21% to 45%). For Seattle dataset, the accuracy of driving traces from 19 out of 21 drivers do not vary with β value. In other words, routing calculation mostly agrees with the best path selected by elastic pathing algorithm. From the reminding two drivers, OSM routing increases the estimation accuracy for P13 but decreases the accuracy for P15.

Applying OSM routing method also increases the algorithm execution time given that a corresponding navigated path needs to be found for each of the top ranked candidate paths. Our algorithm takes about 11 hours to process all the traces in the New Jersey dataset and 25 hours for Seattle dataset.

β=0.5, ω=0.5	New Jersey		Seattle	
Destination Error (meters)	Enhanced Version	Add Routing	Enhanced Version	Add Routing
<250	17%	15%	15%	15%
<500	24%	20%	28%	27%
<750	30%	29%	39%	39%
<1000	35%	33%	49%	49%
<1250	38%	36%	55%	55%

Table 4.3: Comparison of estimation accuracy when OSM routing is applied to our algorithm. Table shows percentages of traces that has destination error within certain range for both New Jersey and Seattle dataset.

4.4.4 Effect of Applying Road Speed Limits

Our algorithm applied road speed limits obtained from OSM to determine how well a path matches with a give speed trace. In this section, we explore how the algorithm performs when we relax this condition. This provides insights on how much the speed limit affects the estimation accuracy for different driving environments: urban and sub-urban.

We found that without using speed limits, the estimation accuracy for Seattle dataset does not drop a lot (e.g. only a few traces difference) while the accuracy drops noticeably by several percentage points for New Jersey dataset (see Figure 4.7). In New Jersey dataset, there are 15% of traces (37 out of 254) having destination error within 250 meters when speed limits are not applied. This is 7 traces fewer than the one using speed limits. For error within 500 meters, there are 19% of driving traces when speed limits are not applied (e.g. 13 traces fewer than the one with speed limits). Interestingly, for Seattle dataset, when we do not apply speed limits, the number of driving traces with destination error within 250 meters is only one trace fewer. The numbers of driving traces distributed in different error intervals are similar.

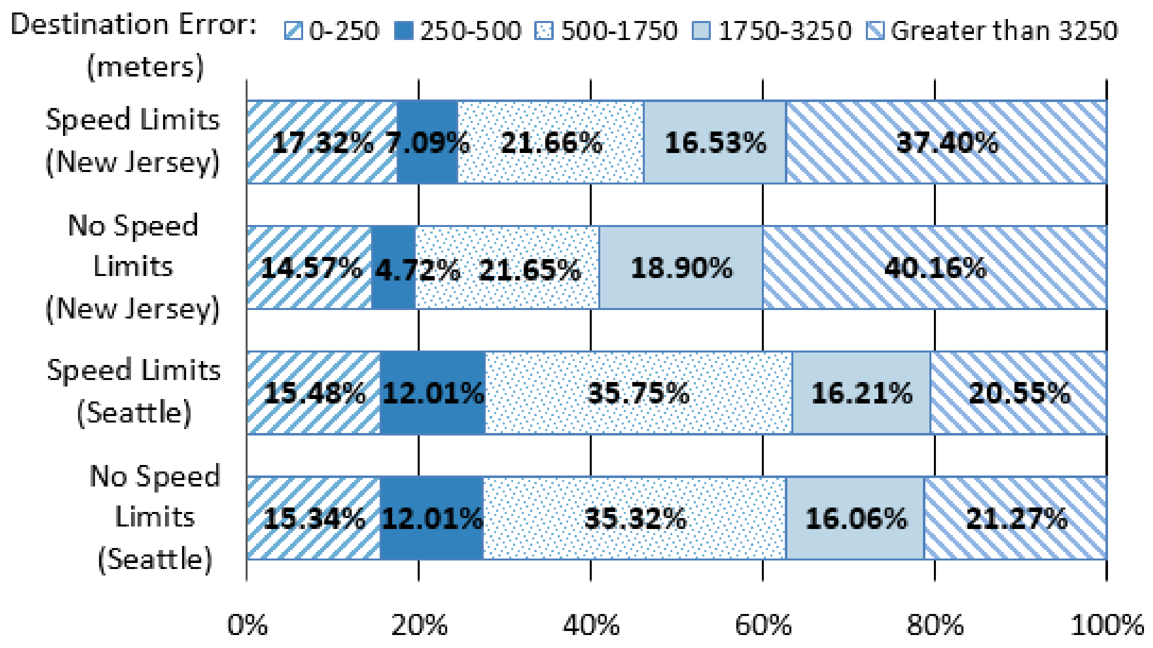

Figure 4.7: Effect of applying speed limits on the New Jersey and the Seattle dataset. Figure shows the percent of driving traces from the corresponding dataset with destination error within different intervals: from 0-250 meters (0.16 mile) to greater than 3250 meters (2 miles).

4.4.5 Results for Identical-Route Dataset

As different drivers can have very different speed patterns even for the same route, we present results for the identical-route dataset with 30 drivers to investigate how driving behaviors may affect our algorithm.

Overall Accuracy: We processed all the driving traces (59 traces for route A and 60 traces for route B) in this dataset to see the distribution of destination errors. It took about 2 hours in total. We used the New Jersey map during the processing.

During the result analysis, we discovered and fixed some issues of the New Jersey map provided by OSM. (1) The nodes in route A and route B were not connected in a region (about 0.5 miles starting from point a in Figure 4.3). Thus, the routes were disconnected from the map, resulting the ground truth path never reached by our algorithm. (2) There were other minor node connection issues: a short two-way road connecting to a local path in route A was mistakenly structured as one-way road in the map, a one-way road in route B was mistakenly structured as two-way road, and a node in one path was wrongly connected to a node in another path in route B. (3) There were many rail ways connecting together with driving ways in the OSM, misleading the algorithm to explore rail ways as well. The name of rail ways were sometimes not specified and sometimes not distinguishable from actual driving ways, making it difficult to separate with programming approaches. We fixed the nodes that were connected in route A and route B, making sure the ground truth path is reachable through the map. However, these issues may still happen in many regions of the map.

We compared the estimation accuracy of our algorithm using the original map and the fixed map (see Figure 4.8 for comparison). We found that fixing the map had a huge impact on the estimation accuracy for both route A and route B. We did not see any difference on the overall accuracy when we applied this fixed map for the New Jersey dataset with 254 traces. This is not surprising, since we only fixed a small region related to the two routes.

The overall estimation accuracy for the identical-route dataset is much lower than the New Jersey and Seattle datasets: 15%-33% traces with error less than 1 mile comparing to about 50% traces with error less than 1 mile for New Jersey and Seattle datasets. This is because the identical-route dataset only consists of two driving routes with fixed region for all drivers. We selected these two driving routes to pass areas with driving environments that were shown to be difficult for our algorithm based on our prior work (Gao et al., 2014). These routes consists of highways, local and dense streets where there are many turns and intersections. We selected two relatively complex routes so that they can cover

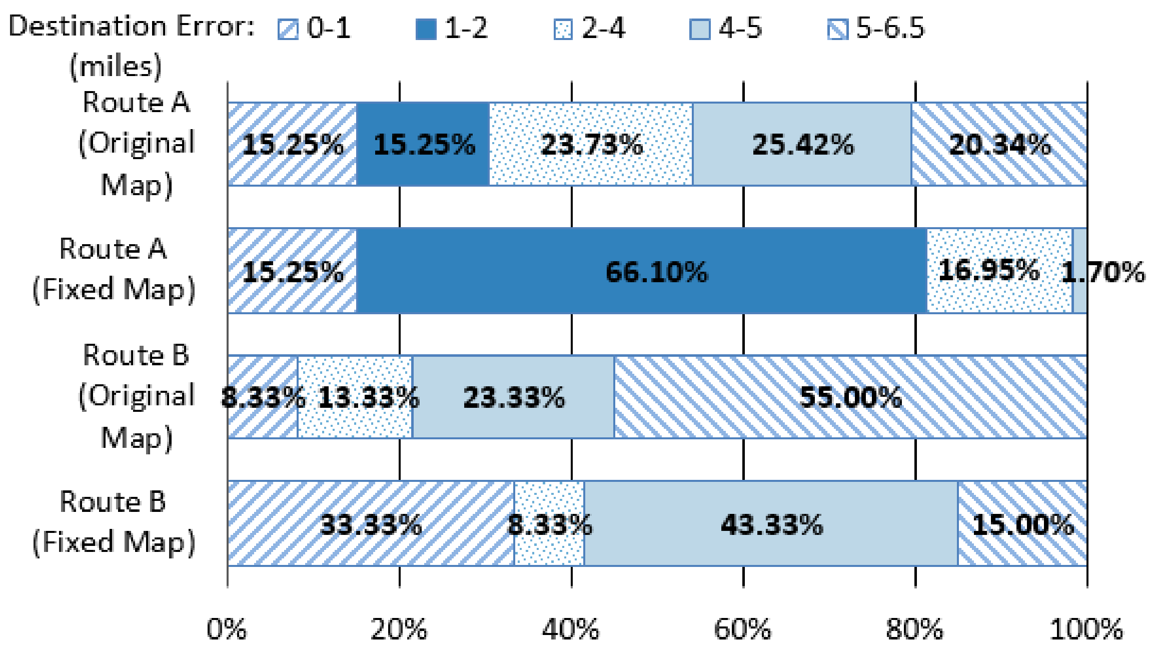

Figure 4.8: Comparison of estimation accuracy when using the original map and the fixed map. Figure shows the percent of traces with destination error within different intervals: from 0-1 mile to 6.5 miles. There is no driving trace with destination error greater than 6.5 miles.

a large variety of driving environments and the results would have reasonable variation for different drivers to analyze how different drivers and different driving behaviors may affect our algorithm.

Driving Behaviors vs. Estimation Accuracy: Different driving behaviors should result in different patterns of speed traces. For example, drivers who drive faster would usually have higher average speed than others given the same route and similar traffic condition. Drivers who brake harder than others would have higher average braking deceleration. For both route A and route B, we extracted some common features from speed traces:

- **Average Speed:** This is the mean of speed values within one driving trace.

- **Average Braking Deceleration:** We extracted all the monotonically decreasing speed intervals from a speed trace. For each speed interval with at least 10 mph decrement, we calculated the average deceleration. We further average these values from different intervals to obtain the overall average braking deceleration. Given only speed trace, it is not easy to extract only braking intervals without complicated analysis since the driving speed can also decrease when releasing the gas petal. Setting threshold for instant braking deceleration may bias some drivers. For our dataset, we found that 10 mph is a great threshold to filter out most non-breaking events for different drivers.

- **Number of Braking Events:** This is the number of monotonically decreasing speed intervals that have speed drop with at least 10 mph.

- **Number of Stops:** This is the number of stops in one driving trace. We are including the starting and ending stops, so there are at least two stops for each trace.

Figure 4.9 shows the percentage of driving traces that have destination error less than one mile for each interval of average speed and average braking deceleration. For driving traces corresponding to route A (e.g. 59 traces in total), there are 22 traces with average speed between 18 and 20 mph, and 32% of them (7 out of 22) have destination error within one mile. This percentage is relatively high comparing to other speed intervals. Thus, our algorithm has slightly better estimation accuracy when the average speed is around 18 to 20 for route A. When the average speed is too slow or too fast, the probability of our algorithm providing good estimates drops rapidly. For instance, there is no trace with error within one mile among the 10 traces with average speed between 14 to 16 mph. For route

B, the distribution of relatively accurate traces (e.g. error less than one mile) over the average speed is different from route A's. The average speed is generally higher than route A. This is due to route B having a long segment of highway path while route A does not pass highways. Relatively low or high average speeds help on our algorithm's estimation in this case (e.g. see figure 4.9). This result is also intuitive for route B: based on our trace analysis, slow average speed usually indicates more stops before or after the highway segment which then provides more information for our algorithm about intersections, and very fast average speed indicates high driving speed in highway segment which helps eliminate some unrelated local candidate paths during the algorithm's speed limit testing. When looking at average braking deceleration, our algorithm has higher probability of giving accurate estimation when the average deceleration is relatively low for both route A and route B.

Figure 4.10 shows how brakes and stops may affect algorithm's estimation accuracy. In general, drivers brake more often in route A than route B. For route A, majority of traces have a number of braking events between 17 to 19. These traces have the highest estimation accuracy. Similarly, for route B, the majority of traces have number of braking events between 8 to 11, and these traces also have the highest estimation accuracy. However, the distribution of accurate traces over the number of stops is different between route A and route B. 8 to 12 stops appear to be the best for our algorithm to estimate route A. While, more stops favors our algorithm's estimation in route B. We found that all 5 traces with 7 to 8 stops in route B have relatively low average speed (e.g. 23-27 mph in Figure 4.9). This aligns with the previous statement about why low average speed in route B may increase the estimation accuracy. We should, however, note that the data sets contained relatively few low and high average driving speeds, as these are extreme cases correspond to distinct driving styles.

We analyzed how brakes and stops correlate with the average speed. We found that stops and the average speed in route A have a high linear correlation ($R^2 = 0.7553$). It is also intuitive that the number of stops in route A, which does not consist of highways but passes through streets in grid path structure, has a large effect on the average speed. Others have relatively low correlation coefficient. For example, the second largest correlation (with $R^2 = 0.2984$) is from stops and the average speed for route B. Brakes and the average speed have very small linear correlation.

4.4.6 Analysis on Traces with Low Estimation Accuracy

We analyzed the traces with low estimation accuracy (destination error more than 2 miles). For each of these traces, we analyzed the main cause of estimation mistakes. The factors misleading the algorithm can be grouped into following categories:

- **Homogeneity of roads:** If an area consists of many similar roads (e.g. same speeds, similar intersection intervals, areas built in a grid), they cannot be distinguished.
- **Unpredictable stops:** If there are several stops (due to traffic or constructions) not near any intersection, the speed trace will mislead our algorithm to find a wrong path.
- **Very few stops:** A speed trace with very few stops lacks information about intersections for the ground truth, resulting many candidate paths cannot be effectively ranked.

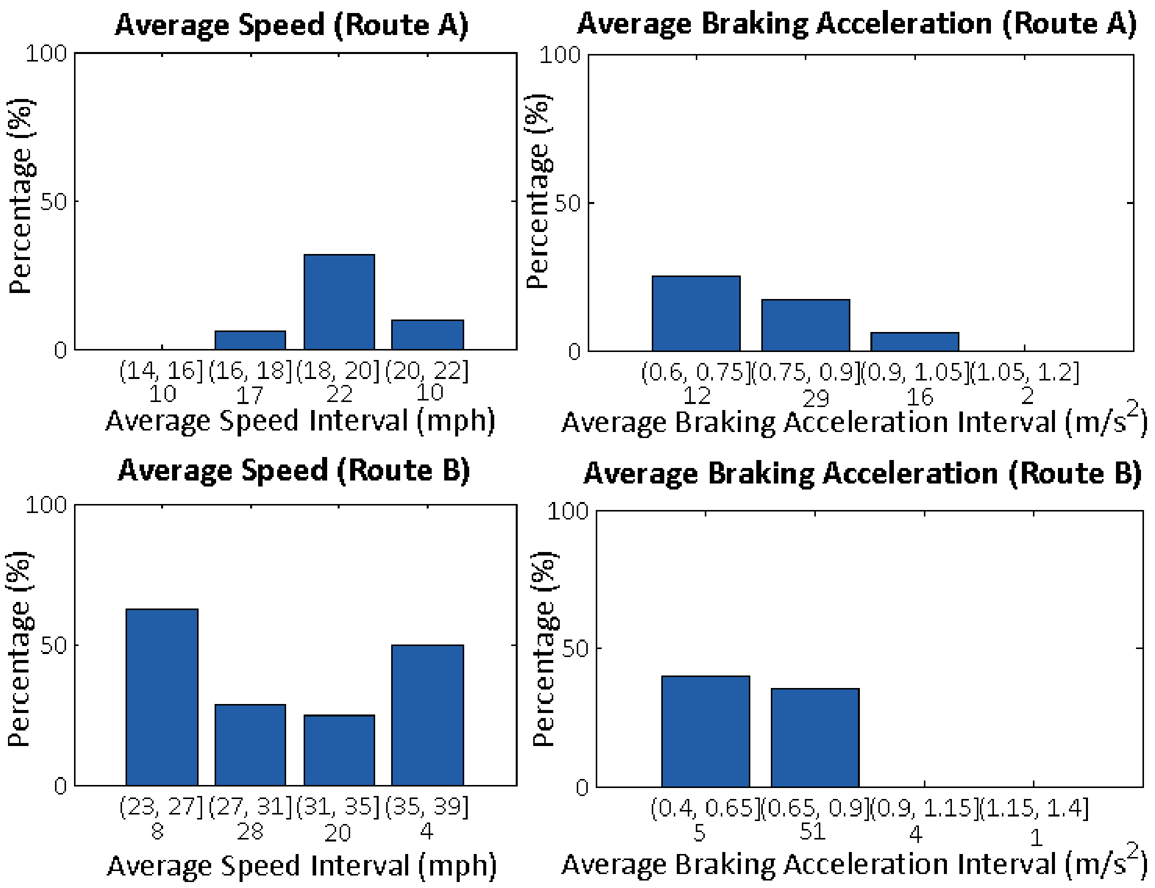

Figure 4.9: The effect of average speed and average braking deceleration to the estimation accuracy. The range of average speed or average braking deceleration is split into four intervals. Values in horizontal axis show both the interval and the number of traces within the interval. The percentage value in vertical axis shows the percent of driving traces within each interval having destination error within one mile.

- **Traces with mostly slow speeds:** Without high speeds to rule out turns and constrain paths to a few major roads, the correct path is indistinguishable from any other. Road speed limit testing also does not help in this case.

- **Limitations of OSM:** Nodes connected by the OSM are sometimes impossible to reach in real driving: connecting a road to a bridge over it, and entering non-driving paths (e.g. railways, non-existing paths, private regions).

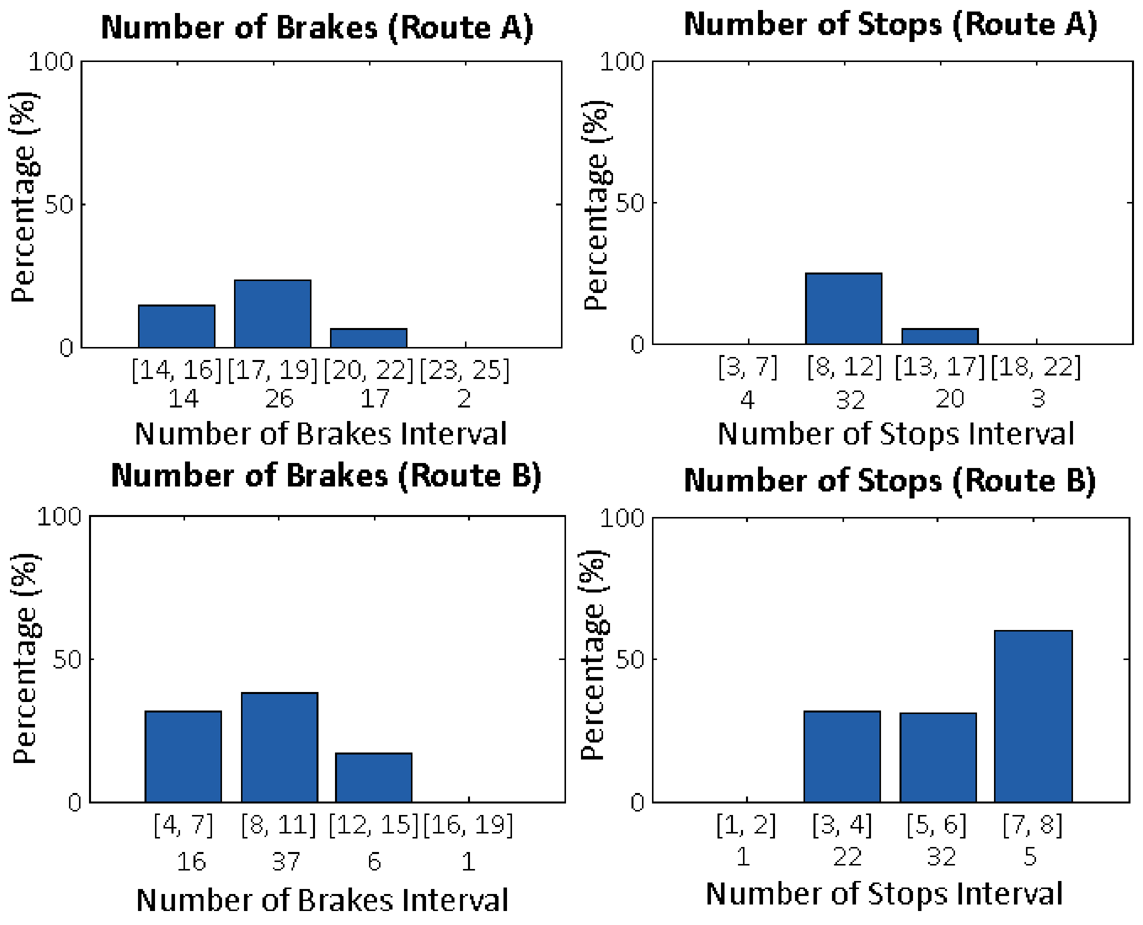

Figure 4.10: The effect of brakes and stops to the estimation accuracy. We use "brakes" in the figure to denote "braking events". Each range is split into four intervals with integer values. Values in horizontal axis show both the interval and the number of traces within the interval. The percentage value in vertical axis shows the percent of traces within each interval having destination error within one mile.

- **Unpredictable direction turnarounds:** Some drivers turned around their car in the midway of a path segment (e.g. illegal u-turn, or drive through a place and then reverse direction).

Based on our analysis of low-accuracy traces from all three datasets, Table 4.4 represents the number of traces that have estimation mistakes caused by each of the six factors. Although poor estimation of a trace may be due to several factors, we only labeled each trace based on the suspected root cause. For New Jersey dataset, limitations of OSM and

	New Jersey Dataset	Seattle Dataset	Identical-route Dataset	
			Route A	Route B
Homogeneity of roads	17	39	2	18
Unpredictable stops	17	28	5	6
Very few stops	13	21	1	10
Mostly slow speeds	22	25	1	3
OSM limitations	23	26	2	3
Direction turnarounds	3	3	0	0
Total	95	142	11	40

Table 4.4: Table shows number of traces with wrong estimation caused by each one of the six factors.

traces with mostly slow speeds appear to be the top factors. For Seattle dataset, homogeneity of roads causes the most number of traces to have poor accuracy. Identical-route dataset consists of two routes with (route B) and without (route A) highway segment. For route A, unpredictable stops appears to be the main factor. However, for route B, homogeneity of roads is the main factor due to several highways near the same region: most of these traces end up in a different direction or on a different highway.

CHAPTER 5

COMBINING MACHINE LEARNING WITH ELASTIC PATHING

5.1 Chapter Overview

In this chapter, we present a new approach by combining machine learning with our Elastic Pathing algorithm to estimate the driver's location based on speed trace. We start by explaining why machine learning is appropriate for this problem and what information this approach can add towards our Elastic Pathing algorithm. We explain the major challenge that we have encountered while using previous plain searching algorithm and show why machine learning approach can potentially further improve the driving route estimation. We then present our overall structure for the algorithm design that well applies the machine learning prediction. Meanwhile, we present our testing datasets for machine learning approach. Finally, we show the improved results compared to previous versions without applying machine learning.

5.2 Introduction of Machine Learning Approach

We first introduce the idea of applying machine learning model to help estimating driving direction when a driver reaches an intersection.

Challenges from Previous Algorithm: Although path reconstruction may seem simple in concept, it is actually very difficult. The first major difficulty is detecting turns with only speed data. When drivers break and then accelerate, did they make a turn or did they slow down because the person in front of them was turning? If drivers come to a complete stop and then accelerate, did they go straight or turn? The insufficient information about turns is the main reason that many candidate routes all appear to fit well with a speed trace. Our algorithm has difficulty determining which roads fit better, especially in areas having similar roads within symmetrical path network region.

Machine Learning Approach: Since the heart of this pathing problem is the lack of turn information, one approach is to treat this as a classification problem. We have previously tried examining the speed traces to extract features: stop, right turn, left turn, or straight road. Then, we could train the model to recognize different turns from the speed trace. However, road intersections and turning options are very diverse. Based on the testing with our datasets, we found various types of left turns and right turns with various turning angles. Different intersections have different number of connecting paths. We were unable to classify turn features with high accuracy due to large variation on turns and intersections.

After trying out various enhancements with our original Elastic Pathing algorithm, we revisited machine learning approach and figured that combining the information of the map with the speed data can potentially lead to feasible model for machine learning. Previous idea on classification only consider the pattern of the speed data and we attempted to label based only on the speed pattern. Indeed, we also have the information of intersections from the OpenStreetMap. If we train the model to learn whether the given speed segment matches with a turn, it can provide a good estimate on the turning direction. With the addition of intersection information from the map during training, the model can provide a good estimation on the quality of matching between speed trace and the turn direction from the map. This can largely help on finding the ground-truth driving route.

Applying Machine Learning Model: In order to train the machine learning model for speed and turn matching, we need to first extract the intersection locations and corresponding speed segments. This sample extraction algorithm needs to follow these three steps for each driving trace: (1) based on the ground-truth GPS coordinates of a driving trace, select the corresponding nodes from the OpenStreetMap that can represent the actual path; (2) extract the intersection nodes and other intersection information (e.g. number of connecting path, turn angle, turn direction); and (3) find the corresponding speed segments at these intersections. Then, we should combine the intersection data with the corresponding speed data to make samples for machine learning algorithm. The ground-truth label for each sample is either "0" the speed trace does not match with the selected direction or "1" the speed trace matches with the selected direction.

Secondly, we should select the best machine learning algorithm for this problem. We can apply Random Forest (Ho, 1995) to train our data and provide prediction. Random Forest estimates the importance of each feature through how much information gain the feature can provide while splitting data into different groups. This machine learning algorithm usually splits feature value into several intervals in order to classify data. This concept fits well with our matching problem because speed values can be intuitively grouped into different regions for different values of turn angle and different turn direction for matching.

Combining the model to help with our algorithm on route tracking is the final step. The model can provide a probability score for different driving directions in each intersection, representing how likely the driver would go for each road segment given the speed trace. Although simply picking the path with the highest probability may find the solution path, this approach is less likely to work for the following reasons: (1) machine learning predictions can make mistakes (e.g. having high prediction accuracy of 0.90 in each intersection can still result in low accuracy if the path has to go through 20 intersections – $0.90^{20} = 0.12$, which is pretty common in our datasets); (2) it may favor paths with few intersections as the probability depends on the number of intersections; and (3) it is still hard to determine solution path if probability values are close (e.g. is probability of 0.95 really much better than 0.90? Why not apply a secondary scoring/error metric to further compare?). Therefore, machine learning and probability metric should be combined with the matching error to better rank different candidate routes. One of the challenges on previous algorithm is that there are many paths all math well with the speed trace. After addition of this probability score, the algorithm can further distinguish the solution route from other similar candidate routes.

5.3 Algorithm Design: Machine learning with Elastic Pathing

In this section, we present our algorithm design that combines the machine learning model with our Elastic Pathing error metric to find the solution path. We first present the overall structure of the algorithm design. Then, we break down and describe details about how we implement each component: machine learning and training, path selection and probability estimation using trained model, and simplified Elastic Pathing to calculate error for final ranking.

5.3.1 Overall Structure of Our Algorithm

Figure 5.1 shows the overall structure of our algorithm for machine learning approach. For each driving trace, the time-stamped speed data and the OpenStreetMap are the inputs for the algorithm. The first step is to explore the map and find the most likely candidate paths ("Path Selection and Probability Estimation" block in the figure) though the use of machine learning prediction. We select this subset of candidate paths such that they have overall higher probability to be the solution path than others in the map. This step effectively filter out many unlikely routes, making later path ranking much easier.

If all the paths near this region do not seem to match well with the speed trace by using machine learning prediction (e.g. all directions have very low probability value), we can simply re-explore the map with our Elastic Pathing algorithm as described in the previous chapter. If we find a small subset of candidate paths, we can further rank these paths through the error metric that we used in our Elastic Pathing algorithm. We call this error calculation method as "Simplified Elastic Pathing" because it has been modified to only calculate the total compression and expansion error needed to math the speed trace to the candidate path. Please note that "Simplified Elastic Pathing" is a small method to calculate the error, which is different from "Elastic Pathing" (our main algorithm to explore and select solution path). Finally, we can obtain the solution path either through machine learning prediction combined with Simplified Elastic Pathing for ranking, or through running our complete Elastic Pathing algorithm when machine learning does not find any path.

In the following subsections, we explain how we implement each component of the design structure in Figure 5.1 in detail.

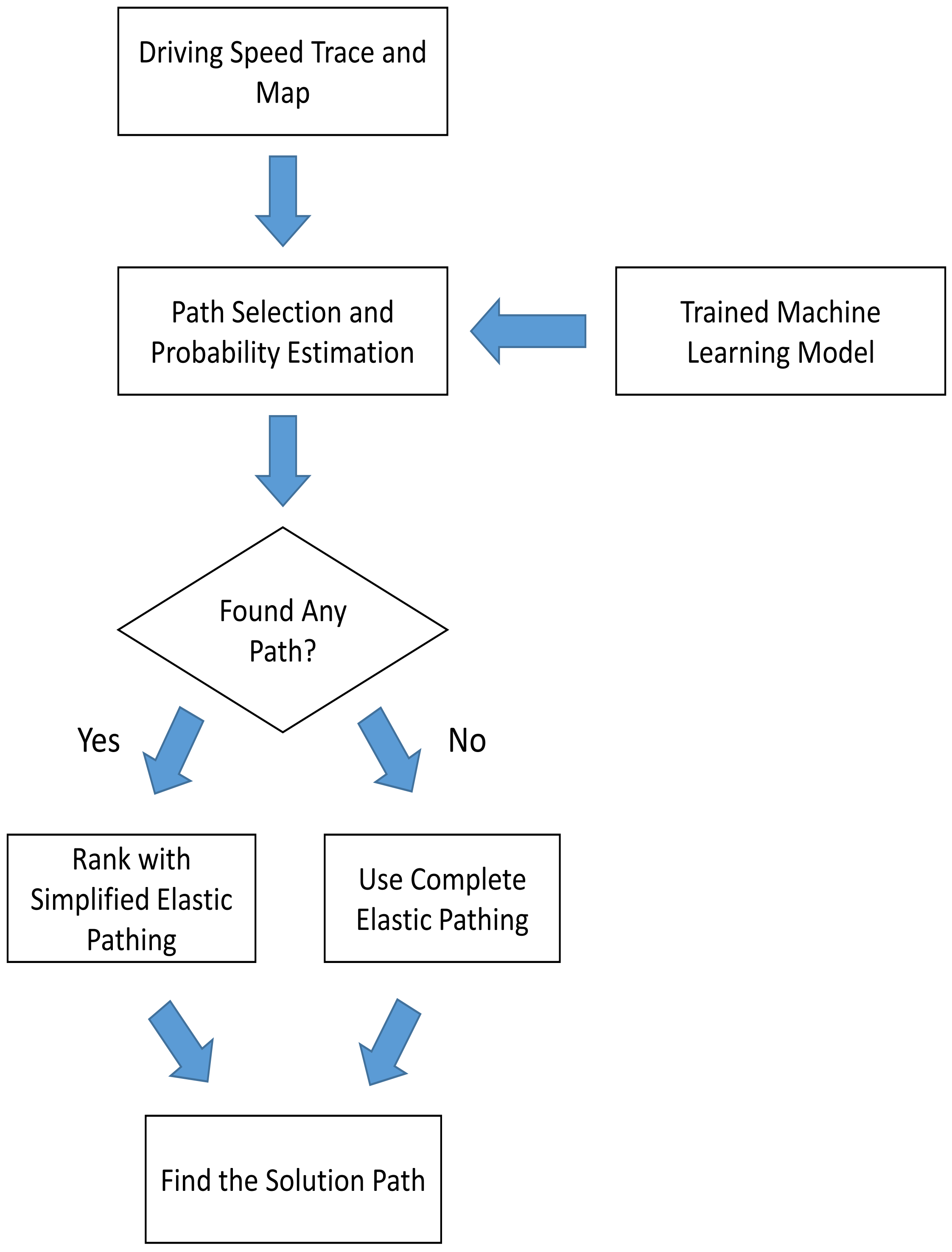
Driving Speed Trace and Map
Path Selection and Probability Estimation
Trained Machine Learning Model
Found Any Path?
Yes
No
Rank with Simplified Elastic Pathing
Use Complete Elastic Pathing
Find the Solution Path

Figure 5.1: Overall Structure of Our Algorithm. Figure shows how we apply and combine machine learning model to our Elastic Pathing algorithm.

5.3.2 Implementation of Our Machine Learning Model

Datasets: We used our Central New Jersey dataset (254 traces) and Identical-Route dataset (119 traces) for the evaluation of our machine learning approach. These two datasets have reliable, clean speed traces as well as good GPS ground-truth traces.

We did not use Seattle dataset for the testing of machine learning approach for the following reasons: (1) the dataset is much noisier (i.e. recall that this dataset was a GPS logger dataset containing walking, biking, flight, running, train, bus, taxi, and car driving trips) than our own datasets; (2) despite our previous effort of filtering for car driving routes, it is still hard to distinguish bus, taxi, and some of the motorcycle trips with the actual car driving trips (i.e. buses and taxis tend to circle around streets and stop very often on random places); (3) due to occasional GPS signal lost and pause of data collection, the Seattle traces after our filtering are much shorter than our own datasets; and (4) machine learning approach requires training and intersection data extracting, so it requires the dataset to have high qualify speed data with high sampling rate and good match with intersections for ground-truth data extracting and labeling.

Intersection Data Extraction: The first step is to extract and label data samples. Our machine learning model aims to predict how likely a driving is going for one direction when the driver reaches the intersection. Each direction (or turning option) in the intersection needs to have a probability value. For example, if a driver reaches a four-way intersection, the model needs to predict the three options (i.e. not considering the direction a driving came from) one by one and provide a probability value for each direction. Therefore, each option in an intersection is a data sample.

Each data sample includes a set of features and a ground-truth label. These features need to include a speed data segment near the intersection, information about the intersection, and information about the selected direction. The label is either “0” indicating the the selected direction is not the ground-truth direction or “1” indicating a match with ground-truth direction. A trained machine learning model can predict the label as “0” or “1” for each direction and provide a confidence score ranging from 0 to 1 indicating the probability of going for the selected direction.

Figure 5.2 shows an example intersection with selected direction going right with a 72 degree turn angle. Features about the intersection and selected direction can be obtained through OSM. With this intersection, we can obtain features: number of path options (2 in this case), turning direction (right, labeled as “-1”), and turn angle (72 degrees). Features about speed segment can be obtained from the speed samples. We cut the speed segment near the intersection: 50 meters before the intersection and 50 meters after the intersection. As different intersections may have different lengths of speed segment (i.e. number of speed samples for 100 meters depends on how fast a drive passes through the intersection, given our sampling rate is 1 sample per second), we re-sampled the speed segment into a fixed length of 10. Note that resampling still preserves the variation of speed values. If the speed segment is shorter than 10, we interpolate to 10 points (or upsampling). If the speed segment is longer than 10, we reduce to 10 points (or downsampling). This approach guarantees that we have fixed number of total features for each training and testing sample for our machine learning model.

Table 5.1 summarizes all the features that we extracted from our driving traces. For New Jersey dataset, we extracted 11,439 samples from 6 drivers. For Identical-Route dataset,

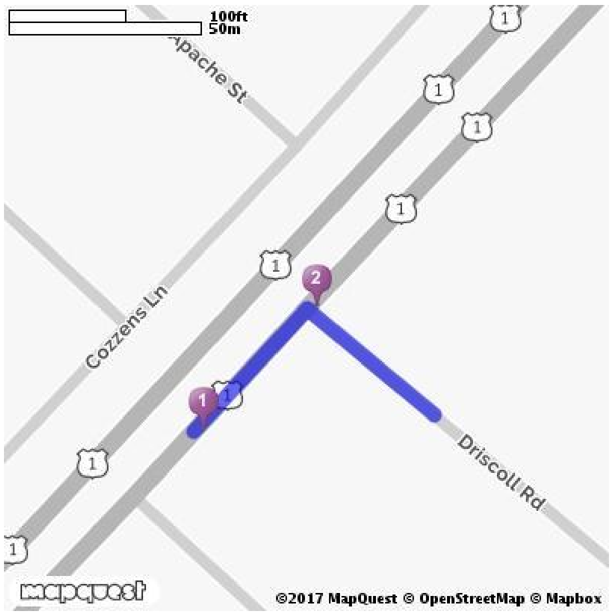

Figure 5.2: Intersection on the Map and Selected Direction. Figure shows the intersection that a driver is passing through. The marker 1 indicates the starting point of entering the intersection. The marker 2 indicates the intersection point. The blue line marked the selected direction. In this figure, we assume the driver is turning right with 72 degree of turning angle. The ground-truth label indicates whether this is an actual driving direction picked by the driver.

	From OSM			From Driving Speed Trace
Features	Number of Path Options	Driving/Turn Direction	Turn Angle	Speed Segment (length of 10)
Value Type	Integer	Right (-1), Straight (0), Left (1)	Double (in degrees)	Double (in mph)

Table 5.1: Summary of Features for Machine Learning. Table shows all the features and corresponding value types. Therefore, each training sample contains 13 features: 10 for speed segment, one for number of path options, one for selected driving direction, and one for turn angle.

we extracted 9,899 samples from 30 drivers. Please note that the number of samples for machine learning model is much larger than the number of driving traces because each trace contains many intersections and different turning options. Therefore, our datasets consist of very large sample size for the purpose of machine learning.

Model Training and Testing with Random Forest: We used Random Forest (Ho, 1995) to train our machine learning model. Considering the real-world attacking situation, it is unlikely to obtain any ground-truth driving direction data from the targeting driver. Therefore, we need to make sure that we do not use any of the targeting driver's ground-truth data for model training. We assume that attackers can obtain driving behavior data (such as speed) and some amount of driving direction data from other existing or hacked datasets. Alternatively, attackers can have volunteers driving around and collecting data about turn and intersections for model training.

We need to train and test both New Jersey dataset and Identical-Route dataset. For Central New Jersey dataset, we used the other 5 drivers data for training and validation. The targeting driver's samples were used for testing. For example, if we are testing for driver P6, we use data from driver P1, P2, P3, P4, and P5 for training and validation. We applied 10 fold cross-validation and used the best estimator for testing. For Identical-Route dataset, we still extract all the samples for testing, but we use only data from Central New Jersey dataset for training and validation. This is because Identical-Route dataset contains only two different routes with 30 drivers. Using data from Identical-Route dataset to train may result in machine learning simply memorizing correct driving directions for the same set of intersections. Instead, we want our machine learning model to learn how different speed patterns can indicate the chance of going to different directions.

5.3.3 Path Selection and Probability Estimation

Path selection and probability estimation step (see Figure 5.1) requires trace inputs (speed and map data) and a trained machine learning model for prediction on intersections. An efficient searching algorithm is needed to explore and select a subset of the most likely candidate paths. When reaching each intersection, we use the trained machine learning model to estimation the probability of going for each direction. At the end, there should be a probability map allowing us to select the top routes with the highest probability.

As the probability score for a candidate path is very important on path selection, we need to have an appropriate way to calculate the probability score for each candidate path. Theoretically, the probability that a driver can go to a candidate path is equal to the multiplication of probabilities along all the intersections within this path. Following shows the equation for calculating this probability for candidate path i:

$$p_i = \prod_{j=1}^{n} p_{ij} \tag{5.1}$$

where n is the number of intersections this candidate path passes through, and p_{ij} is the probability of going to the corresponding direction in intersection j. However, if we use this probability as the scoring metric, it is highly dependent on the number of intersections and it prefers selecting routes passing through fewer intersections. Therefore, we applied average probability as the scoring metric:

$$p_i = 1/n * \sum_{j=1}^{n} p_{ij} \tag{5.2}$$

Note that this average probability is simply the mean of all probabilities along intersections from a candidate route. It is not equal to the probability of the driver going for this route. For convenience, we refer this average probability as “probability score” in following.

We also need to set a threshold on the minimum probability for us to consider the driving direction. If the machine learning model predicts probability lower than this threshold, we need to discard this direction completely. In default, machine learning usually uses 0.5 as the threshold. In our case, we used slightly smaller value of 0.3 because we do not want to discard the actual solution path. In other words, we want the false negative error rate of the model to be as small as possible. Note that setting this threshold is very important given that we used average probability as the scoring metric to select paths (i.e. a path with very low probabilities on several intersections may have fine average probability overall). This also effectively reduce the computational cost of the algorithm while searching for best candidate paths. Our threshold value of 0.3 has been tested to work well with our machine learning model: setting this threshold too high will result in most paths (including the solution path) not selected, and setting it too low will result in too many paths to explore, making the solution path hard to distinguish from others.

Figure 5.3 shows an example of how our path selection algorithm makes decisions. We assume that a driver enters this region from point A. When the driver reaches the intersection (point B), we use our trained model to estimate the probability of going to each of the three directions (BK, BC, and BF). We assume that we obtain the probability values for different directions of each intersection as labeled in the figure (point B, G, and H). The direction HE has probability value of 0.1 smaller than 0.3, so it has been discarded. For this example, there are 4 active candidate path: ABKGI with probability score 0.9 (average of 0.9 and 0.9), ABF with probability score 0.5, ABCDE with probability score 0.8, and ABKGHJ with probability score 0.67 (average of 0.9, 0.5, and 0.6). The probability score needs to be updated as the algorithm further explores each path.

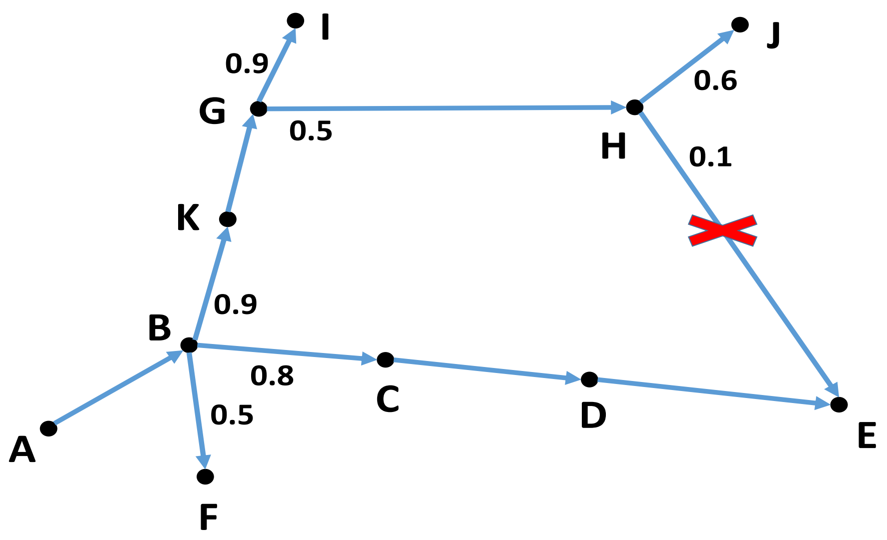

Figure 5.3: Example of how our path selection algorithm explore and select. We assume that driver enters this region from point A.

Algorithm 3 shows the pseudocode of our path selection and probability score calculation. The algorithm applies prioritized searching method to find the most likely candidate paths. Each time, the algorithm searches and advances the partial path having the highest probability score at the moment. In this way, the first completed set of candidate paths should be the best paths. Initially, a starting node needs to be assigned to the starting partial path as shown in Algorithm 3. Then, we need to initialize the probability score for this partial path as well. The while loop does the searching of candidate path. The partial path having the highest probability score (or the average probability) gets advanced first. "advanceUntilIntersection" is a function that calculates distance from the speed trace and matches the distance to the OSM. This function also does the prediction based on the trained machine learning model. The directions with probability less than the threshold are not included when finding new paths. Next, the probability score for each new path needs to be updated and the set of partial paths needs to be sorted by probability score. The while loop exits when we find the top 30 candidate paths or when all the partial paths

```
Input: The starting point-StartNode, a set of speed samples- Samples, a
       threshold for the probability value to accept- $\delta$ = 0.3, and a maximum
       number of top traces considered- MaxNumTrace = 30
Result: Complete = list of most likely candidate paths
1  begin
2      Partial ⟵ {[StartNode ]}
3      // Set the initial probability score as 1.0 for the
          current path, need to be replaced with actual
          average probability value later
4      Partial.setProbScore(1.0)
5      Complete ⟵ $\emptyset$
6      while Complete $<$ MaxNumTrace AND
7      Partial.$size > 0$ do
8          // Pop out the partial path with highest
              probability score to advance
9          P' ⟵ advanceUntilIntersection (Partial.pop, $\delta$)
10         // New paths are found after the intersection,
              then update the probability score for each new
              path
11         updateProbScore (P')
12         // Check for completed paths
13         join (Complete, {$x \in$ P' | $x$ complete })
14         join (Partial, {$x \in$ P' | $x$ incomplete })
15         // Sort by probability score
16         sort (Partial)
17     end
18     // The first completed path found has the highest
          probability score, but we will extract and keep
          the top paths for later processing.
19 end
```

Algorithm 3: Pseudocode for path selection and probability score estimation

have been explored. The maximum traces of 30 has been determined experimentally with our algorithm. Candidate paths that are not in top 30 usually have very small probability scores. For a few exceptions, if there are more than 30 candidate paths all having similarly high probability scores, we may as well cut off in 30 traces because our machine learning model has difficulty selecting routes for the given speed trace in any case (i.e. many routes all seem very good for the algorithm).

One key point to note is that this path selection algorithm does not always find the candidate path with the highest probability score, but it fits our problem the best. Recall that our probability score for one candidate path is the average probability across intersections and we always search for the path with current best average probability score. Different from error that can only increase, average probability score can increase or decrease depending on future prediction on intersections. This means that a path with low current average probability score may end up having high value if future prediction have high probabilities. Unless we actually search through all paths in the whole region, we cannot really find the path with the highest average. Our search that prioritizes the highest current average probability score actually matches well with our problem. This is because the beginning of the prediction is actually more important than the later predictions (e.g. if we predict the first intersection wrong, we may guess a completely different direction than the ground-truth). Therefore, for those paths that have low probabilities in the beginning of the search but having high average probability at the end, they are not as good as others that have high probabilities at the beginning. Our algorithm automatically considers this.

5.3.4 Simplified Elastic Pathing for Ranking of Selected Paths

After path selection with the trained machine learning model, we need to further pick the best path as the solution path. Simply selecting the candidate path does not work well because top selected paths often has similar probability scores. We decided to use the error metric that is similar to our Elastic Pathing algorithm. The purpose is to estimate how each of these candidate paths match with the speed trace. We called this modified error calculation method as “Simplified Elastic Pathing”.

Simplified Elastic Pathing follows similar idea of matching zero speeds to intersections of the candidate path and matching slow speeds to turns of the path. While we match the speed trace with a candidate path, error can occur if the driver stops in the mid-way (speed is zero but not in intersection) or the car is too fast to turn. Similarly, we can correct the path through advancing or rewinding speed samples, if the speed is too fast to turn. We can find the nearest intersection by advancing or rewinding the path if the driver stops in the mid-way.

The difference of this Simplified Elastic Pathing is that we only consider either rewinding or advancing instead of implementing both when error happens. Our main Elastic Pathing algorithm needs to explore both advancing and rewinding because it is searching for different path options while calculating error. When the candidate path and the speed samples are both known and we only want to estimate how well they matches with each other, only one option is actually needed (e.g. the one that gives smaller error) each time error happens. For instance, if the candidate path is the ground-truth path, the speed trace should have small or no error while matching. If the error happens, it is mostly because there are small offsets on distances (e.g. driver stops one or two cars behind the intersection, the actual turning point is just one or two speed samples away). Therefore, whichever error (by rewinding or advancing) that has smaller value should be the correct one. Although checking both rewinding and advancing each time will find similar results eventually, this greedy approach is much more efficient and works well in most cases. The computation cost for checking both rewinding and advancing is at least 2^n where n is the number of times error happens. On the other hand, the computational cost is only n using this greedy approach.

There are some specially cases that we need to consider. First, if there is no candidate path found through machine learning model, we need to use the our complete Elastic Pathing algorithm to re-search and find the solution path. Through our testing, this case can happen quite many times in our datasets because our machine learning approach is much more restrict than our Elastic Pathing algorithm while picking paths. However, if it finds candidate paths, it often finds a small subset of most likely paths with the actual solution path. On the other hand, Elastic Pathing algorithm can find one solution path even if the speed trace is bad (e.g. traffic, stops in random places). Therefore, machine learning approach and our previous Elastic Pathing algorithm actually complement with each other: machine learning approach predicts directions but does not do well on correcting error, and Elastic Pathing corrects and quantifies error but does not know about directions.

Another special case is when the machine learning selection already finds the best path, meaning that we are confident that the path with the highest probability score can be the solution path. This happens when it fits following condition: (1) the path with the highest probability score has value greater than 0.95, (2) this probability score is more than 0.5 larger than the second highest probability score, and (3) this candidate path passes through more than 10 intersections. When all the conditions are met, we can consider our machine learning model is very confident about this candidate path.

Algorithm 4 shows the pseudocode for our Simplified Elastic Pathing function. Except for the two special cases mentioned above, each of the selected candidate path from our machine learning approach needs to pass through this function to calculate the path error. The candidate path has the smallest error is the solution path.

```
Input: A path-P, speed samples-S, the current index into the speed samples-i, and the current node
       index in the candidate path-j.
while $i < S.length$ AND $j < P.length$ do
    // Match 0 speeds to intersections.
    if $S[i].speed \approx 0$ then
        while $S[i].speed \approx 0$ AND $i < S.length$ do
            $++i$
        end
        if P[j] at intersection then
            Pin (P)
        end
        else
            Fore ⟵ compressB (P)
            Back ⟵ stretchB (P)
            if $Fore.error < Back.error$ then
                moveTo (Fore) // Move to Fore and update path error
            end
            else
                moveTo (Back) // Move to Back and update path error
            end
        end
    end
    // Match turns at intersections to slower speeds.
    if P[j] at intersection then
        if $S[i].speed \leq$ maxSpeed ($P[j-1], P[j], P[j+1]$) then
            Pin (P)
        end
        else
            Fore ⟵ compressA (P)
            Back ⟵ stretchA (P)
            if $Fore.error < Back.error$ then
                moveTo (Fore) // Move to Fore and update path error
            end
            else
                moveTo (Back) // Move to Back and update path error
            end
        end
    end
    // Process normally
    // Update total traveled distance
    totalDis ⟵ updateDistance (S[i].speed, S[i].deltaTime)
    $++i$
    if totalDis reaches node P[j] then
        // Advance to next node in the path
        $++j$
    end
end
```

Algorithm 4: Pseudocode for the *Simplified Elastic Pathing* function used to calculate the error, indicating how well the speed trace matches with a path.

5.4 Results

In this section, we present results of this new machine learning approach. We first show the overall accuracy after testing it with our Central New Jersey dataset and Idential-Route dataset. Then, we present separately the results of our machine learning model on predicting driving directions alone.

5.4.1 Overall Accuracy

New Jersey Dataset: We tested our machine learning approach with our New Jersey dataset with 254 traces (mean trip length of 4.65 miles). Table 5.2 shows the overall estimation accuracy using our machine learning approach. We are mostly interested on the number of traces that can be estimated with destination error less than 250 meters and 500 meters. For error less than 250 meters, there are 63 traces from New Jersey dataset (or 25%) having this high estimating accuracy. There are 75 traces (or 30%) with destination error within 500 meters (i.e. 0-250m and 250-500m intervals in the table).

Table 5.2 also shows the results from both branches of our algorithm (i.e. see Figure 5.1 for the two branches based on whether a subset of candidate paths can be found through path selection and probability estimation). As shown in the table, our path selection and probability estimation with our machine learning model can find a set of candidate paths for 147 of 254 traces. For these 147 traces (column "ML+SEP" in Table 5.2), final path ranking through Simplified Elastic Pathing finds 55 traces with destination error within 250 meters. 55 is more than one third of the 147. This means that when our machine learning model can select out a subset of paths, there is a very high chance that the solution path is within these selected path and our Simplified Elastic Pathing has a high chance of

Destination Error		New Jersey			
Meters	Miles	ML+SEP	CEP	Total	Percent
0-250	0-0.16	55	8	63	25%
250-500	0.16-0.31	9	3	12	4.7%
500-750	0.31-0.47	5	7	12	4.7%
750-1000	0.47-0.62	8	5	13	5.1%
1000-1250	0.62-0.78	4	2	6	2.4%
1250-1500	0.78-0.93	5	8	13	5.1%
1500-1750	0.93-1.09	0	2	2	0.79%
1750-2000	1.09-1.24	5	9	14	5.5%
2000-2250	1.24-1.40	3	3	6	2.4%
2250-2500	1.40-1.55	4	6	10	3.9%
2500-2750	1.55-1.71	4	1	5	2.0%
2750-3000	1.71-1.86	1	1	2	0.79%
3000-3250	1.86-2.02	3	1	4	1.6%
>3250	>2.02	41	51	92	36%
Total traces		147	107	254	100%

Table 5.2: Overall results of our machine learning approach for New Jersey dataset. Table shows the number of traces that our algorithm can estimate with a destination error within different intervals. For example, the first error interval is 0-250 meters. Traces that has destination error within this interval are considered as the most accurate traces predicted by our algorithm. Column "ML+SEP" indicates the part of driving traces that have candidate paths selected through our machine learning model and then these candidate paths are passed to our Simplified Elastic Pathing to find the solution path. Column "CEP" indicates the part of driving traces that do not have any highly likely candidate path selected through our machine learning model. Then, they are re-run to find the solution path through our complete Elastic Pathing algorithm. "Total" column shows the total number of traces that are estimated with destination error within the corresponding error interval.

picking it out. For the part of traces (107 out of 254) that our machine learning model is not able to find good candidate paths, our complete Elastic Pathing algorithm complements it. There are 8 additional traces with error less than 250 meters found by our complete Elastic Pathing algorithm. However, most of the accurate traces (55 out of 63) are found by the "ML+SEP" branch.

Our machine learning approach shows an improvement on estimation accuracy compared with previous Elastic Pathing alogrithm. We compare the estimation accuracy of this

Destination Error (meters)	New Jersey	
	Enhanced Version	Machine Learning Version
<250	17%	25%
<500	24%	30%
<750	30%	34%
<1000	35%	39%
<1250	38%	42%

Table 5.3: Comparison of overall results for New Jersey dataset: the enhanced version of our Elastic Pathing algorithm vs. the machine learning version that we present in this chapter.

machine learning version to our previous enhanced version of Elastic Pathing algorithm in Table 5.3. There are three versions of our algorithm presented in this thesis: the initial version (presented in Chapter 3), the enhanced version (presented in Chapter 4), and the machine learning version (presented in Chapter 5). As our enhanced version is better than the initial version, we compare the enhanced version to the machine learning version in this section to show our new improvements on accuracy. The machine learning version has about 25% traces having destination error less than 250 meters, which is much better than our enhanced version of Elastic Pathing that has about 17% traces in this same error interval. Similarly, for error less than 500 meters, the machine learning version shows better estimation accuracy than the enhanced version (30% vs 24%). The detailed cumulative distribution function for New Jersey dataset and its comparison to enhanced version of Elastic Pathing are shown in Figure 5.4.

Identical-Route Dataset: In this subsection, we present the estimation results for the Identical-Route dataset (30 drivers with 159 traces, recall that we discarded one trace due to one participant went to the wrong route). Trip lengths for this dataset are all about 5 miles. In general, our estimation accuracy is much lower on this dataset compared with

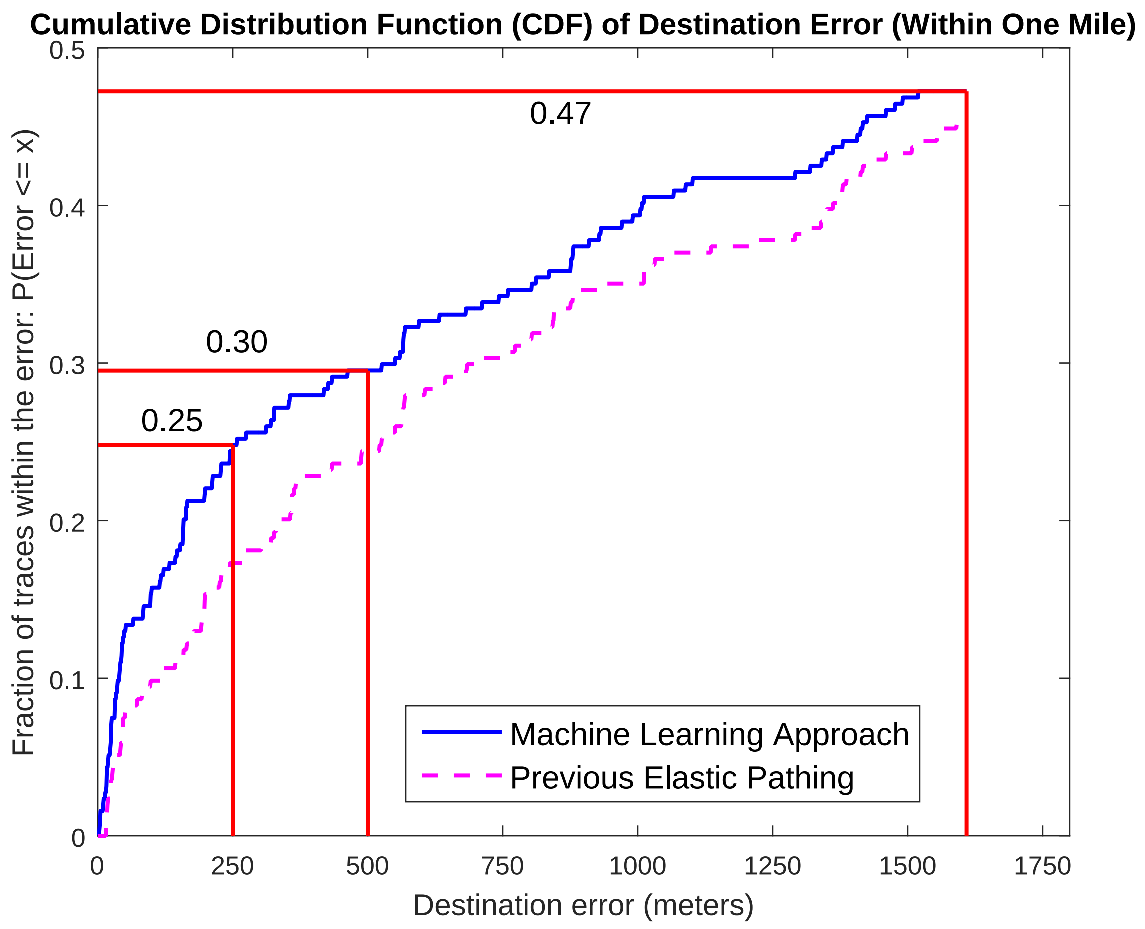

Figure 5.4: Figure shows the cumulative distribution function curves for destination error within one mile from our New Jersey dataset. The CDFs for both machine learning version and enhanced version of previous Elastic Pathing algorithm are shown in the figure for comparison.

other datasets. This is not surprising as we intentionally selected two routes (Route A and Route B) that have comprehensive driving environments to analyze how different drivers can affect our algorithm's estimation accuracy (see Chapter 4 for analysis). If the selected routes were to easy for our algorithm, we could not analyze individual driving behavior differences on estimation accuracy (because our algorithm may estimate all routes very well for all drivers).

Table 5.4 shows the overall estimation accuracy with our machine learning approach. For Route A, the "ML+SEP" branch of our algorithm processed 28 traces. The reminding 31 traces were processed through our complete Elastic Pathing algorithm. Most of the accurate traces (16 out of 19), meaning traces with error less than one mile, are found by our "ML+SEP" branch. However, for Route B, our machine learning path selection and probability estimation only find high candidate paths for 9 traces. 51 traces have solution path found by our complete Elastic Pathing algorithm. Our complete Elastic Pathing branch finds most of the accurate traces in this case (18 out of 20). Still, for the 9 traces that our "ML+SEP" found, there are 2 traces (22%) having error less than one mile. This shows that our machine learning model still finds good traces effectively. In other words, our path selection with the machine learning model is rather conservative: it does not select any candidate path if it is not sure they are good. On the other hand, our complete Elastic Pathing always finds a solution even if candidate paths are mostly bad. The overall distribution (CDF) of destination error within one mile for Identical-Route dataset and its comparison to previous Elastic Pathing algorithm are shown in Figure 5.5.

The difference of the results in Route A and Route B also reflects an interesting fact about the two branches (i.e. "ML+SEP" and "CEP") of our machine learning approach.

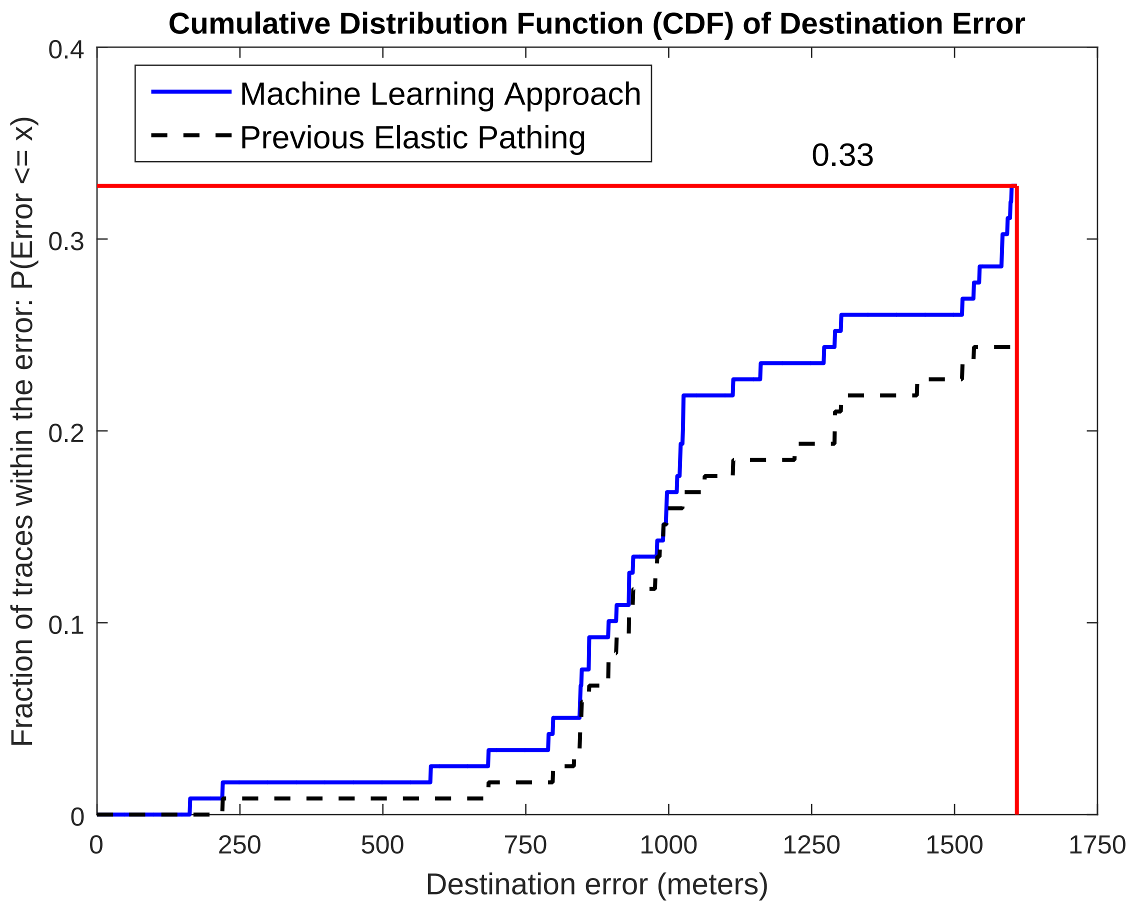

Figure 5.5: Figure shows the cumulative distribution function curves for destination error within one mile from our Identical-Route dataset. The CDFs for both machine learning version and enhanced version of previous Elastic Pathing algorithm are shown in the figure for comparison.

Recall that Route A passes through many intersections (and traffic lights) with commercial areas and grid type of residential areas, while Route B consists of only a few intersections with a long highway segment. Our path selection based on machine learning model is mostly focus on predicting driving directions in intersections. For Route A that has many intersections, “ML+SEP” branch has obvious advantage on estimating routes. For Route B that has very few intersections, the “CEP” branch does better than the “ML+SEP”. This is because our complete Elastic Pathing is good at estimating errors based on distance mismatch but weak on estimating directions especially on intersections. The only simple information our complete Elastic Pathing algorithm can use during the intersections is whether the speed is slow enough to make the turn. If a driver stops or drives slowly in intersections (which happens a lot), our complete Elastic Pathing algorithm simply considers all directions to be equally good.

Table 5.5 shows the improved accuracy (33% vs. 24% with error less than one mile) of our machine learning approach compared with enhanced version of our Elastic Pathing algorithm for the Identical-Route dataset. If we look at two routes separately, for Route A, our machine learning version estimated 32% of traces with error less than one mile. For Route B, our machine learning version performs the same as the enhanced version (both having 60 traces with error less than one mile).

Compared with accuracy for other datasets, the Identical-Route dataset still has lower accuracy overall even with our machine learning approach. This shows that the traces with comprehensive driving environments and complex intersections are also challenging for our machine learning approach. However, we are able to put the estimation accuracy one step further with our new machine learning approach.

Miles	Route A		Route B		Total	Percent
	ML+SEP	CEP	ML+SEP	CEP		
0-1	16	3	2	18	39	33%
1-2	0	20	1	0	21	18%
2-3	9	7	0	0	16	13%
3-4	0	1	1	4	6	5.0%
4-5	2	0	0	23	25	21%
>5	1	0	5	6	12	10%
Total	28	31	9	51	119	100%

Table 5.4: Overall accuracy of our machine learning approach for Identical-Route dataset. We use a different scale for destination error measurement because the estimation accuracy for this dataset is lower than other datasets. Intervals are: 0-1 mile, 1-2 miles, 2-3 miles, 3-4 miles, 4-5 miles, and greater than 5 miles. We mostly consider the 0-1 mile error interval as our algorithm predicts to the right direction when the error is within one mile (given that each trip length is about 5 miles). Again, column "ML+SEP" indicates the part of driving traces that have candidate paths selected through our machine learning model and then these candidate paths are passed to our Simplified Elastic Pathing to find the solution path. Column "CEP" indicates the part of driving traces that do not have any candidate path selected by our machine learning model and the solution path was found through rerunning our complete Elastic Pathing algorithm.

Identical-Route Dataset	Number of Traces	Destination Error <1 mile	
		Enhanced Version	Machine Learning Version
Route A	59	15%	32%
Route B	60	33%	33%
Overall	119	24%	33%

Table 5.5: Comparison of overall results for Identical-Rout dataset: the enhanced version of our Elastic Pathing algorithm vs. the machine learning version that we present in this chapter.

5.4.2 Evaluation of Machine Learning Model

In this subsection, we show how our machine learning model predicts driving direction. Machine learning model is the most important part of our path selection and probability estimation block in our overall algorithm structure (see Figure 5.1). Therefore, the good prediction accuracy from our machine learning model places an important role on the whole algorithm.

Overall, for New Jersey datasets, our prediction accuracy ranges from 0.86 to 0.92 for different drivers. The equal error rates from our machine learning models range from 0.08 to 0.14 for different drivers. Each driver's model was trained using other drivers' data, so different drivers have different trained models. For Identical-Route dataset, there is only one model (trained using New Jersey data and tested with Identical-Route data). Our prediction accuracy is 0.81 and the equal error rate (EER) is 0.19 for this model.

New Jersey Dataset: Figure 5.6 shows ROC curve for six driver in the New Jersey dataset about how well their trained machine learning model predicts the driving direction. Driver 2 has the smallest EER value of 0.08. Driver 1 and Driver 5 have the largest EER value of 0.14. On the point having equal error rate, the threshold probabilities for determining whether a direction is correct for six drivers are: 0.49 (Driver 1), 0.54 (Driver 2), 0.54 (Driver 3), 0.49 (Driver 4), 0.47 (Driver 5), and 0.46 (Driver 6). These are all close to 0.5, so their values are reasonable within practical cases. As mentioned in previous section, in our algorithm, we used probability threshold of 0.3 because we do not want to falsely exclude driving directions that are actually correct (i.e. we prefer very small false negative rate) and we do not want to include a lot of less likely directions (i.e. the threshold cannot be too low).

Identical-Route Dataset: Figure 5.7 shows ROC curve for the Identical-Route dataset. This whole dataset is the testing dataset, so we do not need to have models for different drivers. The EER value for this machine learning model is 0.19 with prediction accuracy of 0.81. The probability threshold for the point with equal error rate is 0.44. Again, we use much smaller probability threshold of 0.3 within our algorithm.

For both datasets, these testing accuracies from our machine learning model are actually pretty good given the large challenge of our problem. For example, each driver uses model that is trained using other drivers' data with different driving environments and driving routes. In addition, the predicting can be very challenging in many cases: different intersections (e.g. "Y" shape, "T" shape, "+" shape, and other irregular shapes), different drivers (e.g. speed variance), and the similarity of speed pattern for different directions (e.g. left vs. right).

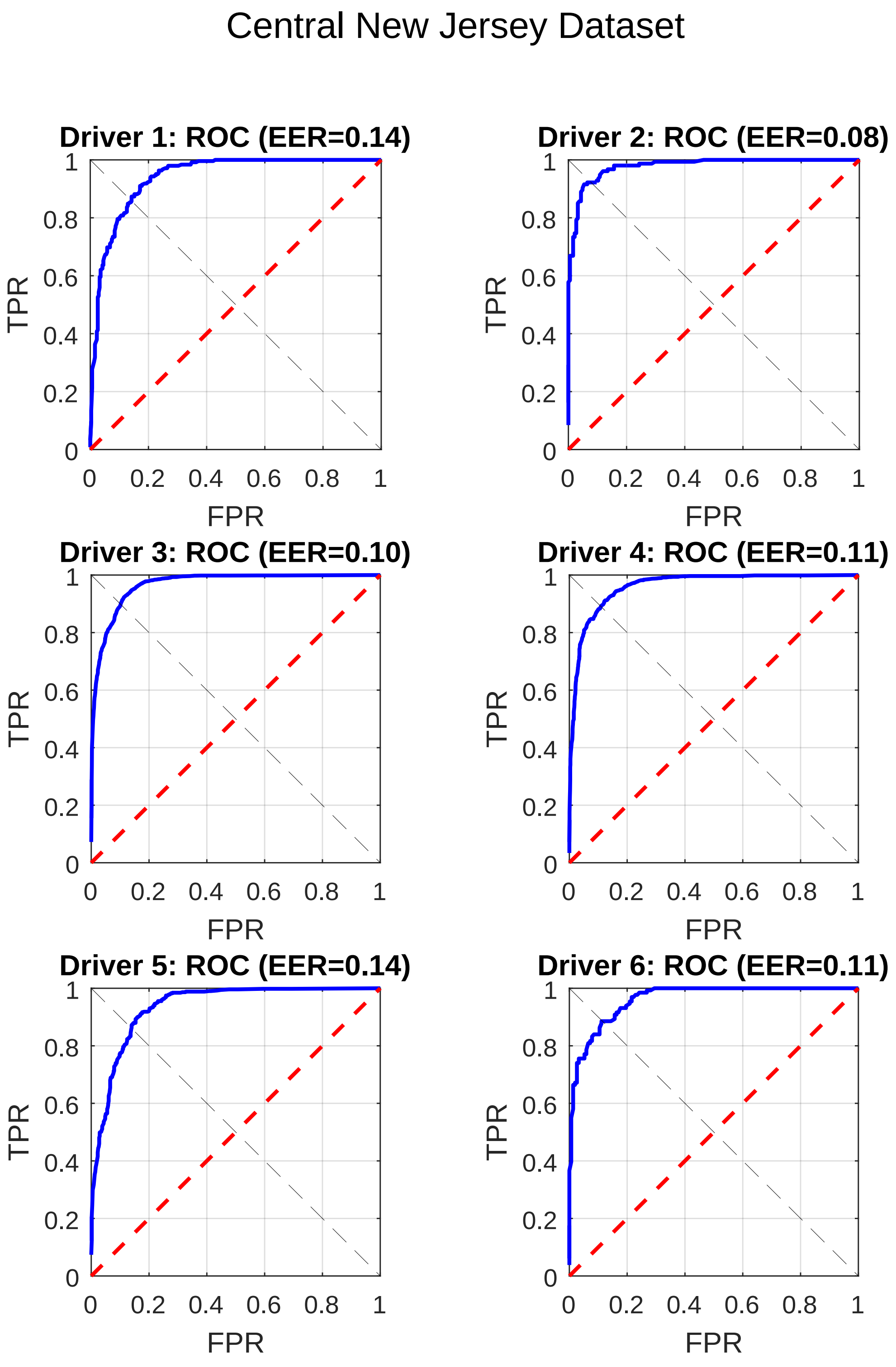

Figure 5.6: Receiver Operating Characteristic (ROC) curves and EER values from six drivers in the New Jersey dataset. The horizontal axis shows the false positive rate (FPR) and the vertical axis shows the true positive rate (TPR).

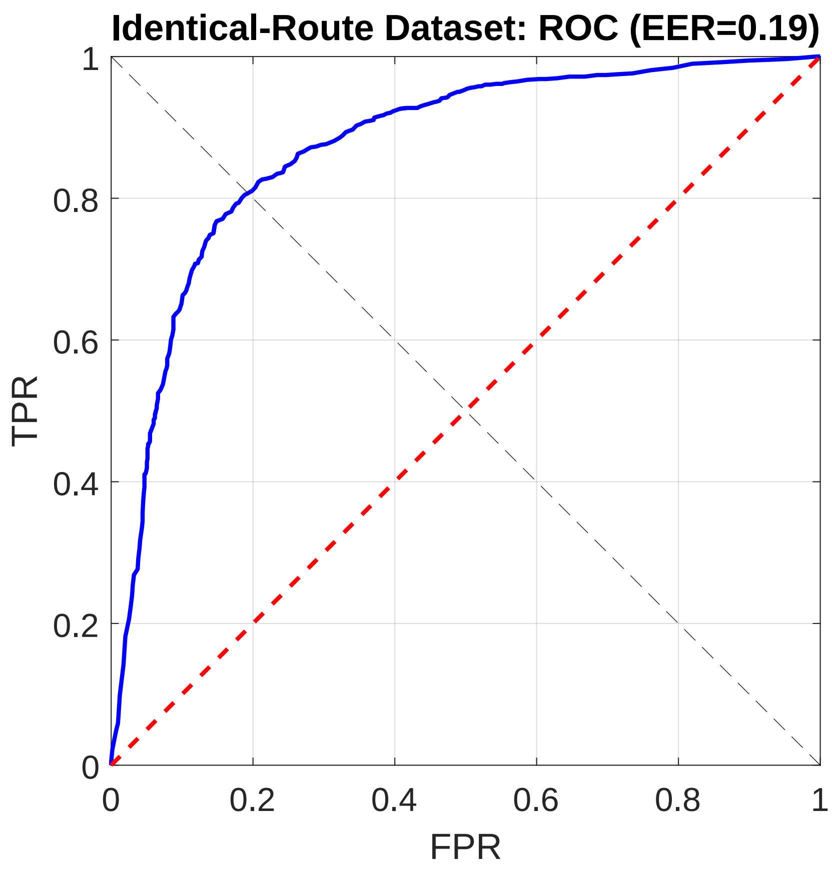

Figure 5.7: The Receiver Operating Characteristic (ROC) curve and the EER value for the machine learning model for the Identical-Route dataset. The horizontal axis shows the false positive rate (FPR) and the vertical axis shows the true positive rate (TPR).

CHAPTER 6

DISCUSSION

This chapter discusses different versions of our algorithm, improvements on estimation accuracy, limitations, and the importance of privacy-preserving data collection.

Algorithms and Improvements on Estimation Accuracy: This thesis presents three different versions (i.e. the initial version, the enhanced version, and the machine learning version) of our algorithm to estimate driving routes based on the starting location and the speed data.

Our initial version and the enhanced version of our Elastic Pathing algorithm share similar overall structure: using a prioritized search algorithm combined with an effective error metric for path selecting and ranking. The enhanced version presents several optimizations over the initial version, such as applying a new comprehensive mathematical model for turn radius determination, and it shows a noticeable improvement on estimation accuracy: it estimated 17% (was 14%) of traces from New Jersey dataset and 16% (was 13%) traces from Seattle dataset with error within 250 meters. Our Elastic Pathing algorithm provides reliable estimation results for traces having good speed patterns. Such traces have no heavy traffic, have very few random stops or maneuvers, and carry enough information (e.g. reasonable stops and speed values) to distinguish with other roads. This search algorithm can automatically prioritize those directions that seem to match well with the speed traces. Thus, it usually generates results instantly or within seconds for good speed traces. For ambiguous driving trips (e.g. either because of homogeneous driving environments or poor speed patterns), it may take much longer and the estimation is usually not good.

Based on our results, OSM routing with shortest path does not improve the overall accuracy of our existing datasets. However, it can be useful in estimating routes for a subset of drivers who mostly drive shortest routes in their daily commuting. Based on our results, routing method had different effects on suburban (New Jersey) and urban (Seattle) datasets. It also showed improvements on the accuracy for P4 and P6 in our New Jersey dataset and P13 in the Seattle dataset. In our case, the datasets are very comprehensive with various driving environments. The routing approach brings more penalty than benefits for our datasets.

On the other hand, using speed limits is important for our algorithm – there is a large accuracy drop when removing this constraint for New Jersey dataset. However, removing speed limit constraint has very small impact for the Seattle dataset. This may be because the traces in Seattle dataset have similar speed limits. The types of roads (mainly consists of streets built in a grid) in urban areas are not as diverse as those in suburban areas. In this case, speed limits does not help in eliminating candidate paths.

Our algorithm accuracy highly depends on different driving habits as shown in our analysis for identical-route dataset. For route A and route B, our algorithm favors different behaviors (e.g. different distribution patterns for stops and average speeds in Figure 4.10 and Figure 4.9) due to different driving environments. However, they also favor some common driving features: both routes have high estimation accuracy when average braking deceleration is low, and the effect of braking has similar patterns.

Our final approach using machine learning with our New Jersey dataset shows further improvements on estimation accuracy. As machine learning model provides information that complements with our Elastic Pathing algorithm, the overall combination works well when tested with both New Jersey dataset and the Identical-Route dataset. For the New Jersey dataset, our machine learning version can estimate 25% (was 17% with the enhanced version) traces with destination error less than 250 meters. This is a big improvement compared to previous accuracy improvement from the initial version to the enhanced version. Even with the Identical-Route dataset where the two driving routes are very comprehensive, the machine learning version shows improvements on the accuracy as well (e.g. 33% traces compared with previous 24% traces having destination error less than one mile).

Compared with previous approach with only search algorithm, machine learning version requires some driving data and training. However, once the machine learning model has been trained, the algorithm run time is very similar to our Elastic Pathing algorithm. The path selection and probability estimation step can automatically prioritize the search from the current best path. It finds a set of candidate paths very quickly if the speed trace is good. In the machine learning path selection, the speed trace appears to be good when speed segments for intersections can easily indicate driving direction with high confidence. If there is no candidate path looking good for the machine learning model, it usually makes decision fast as well. Then, these traces get passed into our complete Elastic Pathing algorithm to find a solution path. There is only one case that requires our path selection to run for a long time: many candidate paths appears to be good and path selection has to switch the searching direction all the time until the first set of candidate paths found. This case usually results in many candidate paths found from probability estimation and it is also less likely to find the ground-truth path at the end.

Limitations: One limitation is from the OSM routing. We applied OSM routing API currently available: finding the route with shortest distance. It does not have options to customize and find the fastest route to reach the destination for example. Intuitively, finding the fastest route is more challenging than finding the shortest route, as selection of fastest route depends on the traffic condition at the moment of driving. Drivers may not necessarily take the shortest route, so a better routing API allowing different customization and settings (e.g. finding the fastest route) may help improve our algorithm's estimation accuracy.

Notes on Privacy-Preserving Data Collection: Our work is not directed against any company or organization. However, prior experience has shown that even well-intentioned uses of data can result in losses of privacy, and thus, we wish to highlight potential dangers of this type of data collection. We do not claim that insurance companies are violating policy holder privacy in this way. However, the general principle of any privacy-preserving data collection is to collect only the data that is necessary for a particular application, and no more.

CHAPTER 7

CONCLUSION

In this thesis, we presented our elastic pathing algorithm which demonstrates an effective attack against user location privacy that uses only speed data from driving traces and an initial starting location. Over the three versions of our algorithms, we have shown improvements on the estimation accuracy. For New Jersey dataset, we can estimate around 25% of traces with destination error less than 250 meters with our machine learning approach. Although our Seattle dataset (e.g. relatively noisy traces with buses and taxes, having mostly short traces with few intersections to extract and train) does not match the requirement for the machine learning approach, our enhanced Elastic Pathing algorithm still estimates destinations with a good accuracy that effectively attacks location privacy: 16% traces with destination error less than 250 meters.

In addition, we performed a comprehensive analysis on our algorithm and data: we evaluated effect of applying speed limits, compared how different driving behaviors can affect our algorithm with a new identical-route dataset, built a tool that can visualize any driving trace data with animation in the Google map, and analyzed traces with low estimation accuracy and summarized the cases that can mislead our algorithm. To the end, we have effectively shown that our algorithm can solve the major challenge of estimating the driving route using very limited data.

BIBLIOGRAPHY

AASHTO. (2011). *A policy on geometric design of highways and streets* (6th ed.). Author.

Allstate. (2013). *drivewise.* (Retrieved September 27, 2013 from http://www.allstate.com/drive-wise.aspx)

Apuzzo, M. (2003, December). Police increasingly use electronic toll records to solve crime. *The Associated Press.*

Aviv, A. J., Sapp, B., Blaze, M., & Smith, J. M. (2012). Practicality of accelerometer side channels on smartphones. In *Proc. of acsac'12.*

Bellatti, J., Brunner, A., Lewis, J., Annadata, P., Eltarjaman, W., Dewri, R., & Thurimella, R. (2017, Jan). Driving habits data: Location privacy implications and solutions. *IEEE Security Privacy*, *15*(1), 12-20. doi: 10.1109/MSP.2017.6

Brush, A. B., Krumm, J., & Scott, J. (2010). Exploring end user preferences for location obfuscation, location-based services, and the value of location. In *Proc. of ubicomp '10.*

Davidoff, S., Ziebart, B. D., Zimmerman, J., & Dey, A. K. (2011). Learning patterns of pick-ups and drop-offs to support busy family coordination. In *Proc. of chi '11.*

Dewri, R., Annadata, P., Eltarjaman, W., & Thurimella, R. (2013). Inferring trip destinations from driving habits data. In *Proc. of 11th acm workshop* (pp. 267–272).

Eagle, N., & Pentland, A. S. (2009). Eigenbehaviors: identifying structure in routine. *Behavioral Ecology and Sociobiology*, *63*(7), 1057–1066. Retrieved from `http://www.springerlink.com/index/10.1007/s00265-009-0739-0`

Farrahi, K., & Gatica-Perez, D. (2011, January). Discovering routines from large-scale human locations using probabilistic topic models. *ACM Trans. Intell. Syst. Technol.*, *2*, 3:1–3:27. Retrieved from `http://doi.acm.org/10.1145/1889681.1889684` doi: http://doi.acm.org/10.1145/1889681.1889684

Gao, X., Firner, B., Sugrim, S., Kaiser-Pendergrast, V., Yang, Y., & Lindqvist, J. (2014). Elastic pathing: Your speed is enough to track you. In *Proceedings of the 2014 acm international joint conference on pervasive and ubiquitous computing* (pp. 975–986). New York, NY, USA: ACM. Retrieved from `http://doi.acm.org/10.1145/2632048.2632077` doi: 10.1145/2632048.2632077

GMAC. (2015). *National general insurance low-mileage discount.* (Retrieved November 7, 2015 from http://www.gmacinsurance.com/auto-insurance/smart-discounts/low-mileage-discount.asp)

Golfarelli, M., Maio, D., & Rizzi, S. (1998). Elastic correction of dead-reckoning errors in map building. In *Intelligent robots and systems, 1998. proceedings., 1998 ieee/rsj international conference on* (Vol. 2, p. 905-911 vol.2). doi: 10.1109/IROS.1998.727315

Golle, P., & Partridge, K. (2009). On the anonymity of home/work location pairs. In *Proc. of pervasive '09.*

Gonzle, M. C., Hidalgo, C. A., & Barabasi, A.-L. (2008, June). Understanding individual human mobility patterns. *Nature*, *453*, 779-782. doi: doi:10.1038/nature06958

Gruteser, M., & Grunwald, D. (2003). Anonymous usage of location-based services through spatial and temporal cloaking. In *Proc. of mobisys '03.*

Guha, S., Plarre, K., Lissner, D., Mitra, S., Krishna, B., Dutta, P., & Kumar, S. (2010). Autowitness: locating and tracking stolen property while tolerating gps and radio outages. In *Proc. of sensys '10.*

Han, J., Owusu, E., Nguyen, L. T., Perrig, A., & Zhang, J. (2012). Accomplice: Location inference using accelerometers on smartphones. In *Proc. of comsnets'12.*

Ho, T. K. (1995). Random decision forests. In *Proceedings of the third international conference on document analysis and recognition (volume 1) - volume 1* (pp. 278–). Washington, DC, USA: IEEE Computer Society. Retrieved from `http://dl.acm.org/citation.cfm?id=844379.844681`

Krakiwsky, E. J., Harris, C. B., & Wong, R. V. (1988). A kalman filter for integrating dead reckoning, map matching and gps positioning. In *Position location and navigation symposium, 1988. record. navigation into the 21st century. ieee plans '88., ieee* (p. 39-46). doi: 10.1109/PLANS.1988.195464

Krumm, J. (2007). Inference attacks on location tracks. In *Proc. of pervasive'07.*

Krumm, J. (2009a). Realistic driving trips for location privacy. In *Proc. of pervasive '09.*

Krumm, J. (2009b, August). A survey of computational location privacy. *Personal Ubiquitous Comput.*, *13*(6), 391–399. Retrieved from http://dx.doi.org/10.1007/s00779-008-0212-5 doi: 10.1007/s00779-008-0212-5

Krumm, J., & Brush, A. (2009). *Msr gps privacy dataset 2009.* (Retrieved from http://research.microsoft.com/ jckrumm/GPSData2009)

Krumm, J., & Brush, A. J. B. (2011). Learning time-based presence probabilities. In *Proc. of pervasive'11.*

Ludford, P. J., Priedhorsky, R., Reily, K., & Terveen, L. (2007). Capturing, sharing, and using local place information. In *Proc. of chi '07.*

General Motors. (2016). *Onstar.* (Retrieved April 20, 2016 from https://www.onstar.com/)

OpenStreetMap. (2016a). *Node.* (Retrieved June 2, 2016 from http://wiki.openstreetmap.org/wiki/Node)

OpenStreetMap. (2016b). *Way.* (Retrieved June 2, 2016 from http://wiki.openstreetmap.org/wiki/Way)

Mormon. (2016, December). *Osm routing in ruby based on pyroutelib2.* (https://github.com/geronimod/mormon)

OpenStreetMap. (2016). *Routing.* (Retrieved May 10, 2016 from http://wiki.openstreetmap.org/wiki/Routing)

OpenstreetmapWiki. (2015, January). *Pyroutelib2.* (http://wiki.openstreetmap.org/wiki/Pyroutelib2)

Patil, S., & Lai, J. (2005). Who gets to know what when: configuring privacy permissions in an awareness application. In *Proc. of chi'05.*

Popa, R. A., Blumberg, A. J., Balakrishnan, H., & Li, F. H. (2011). Privacy and accountability for location-based aggregate statistics. In *Proc. of ccs '11.*

Progressive. (2013). *Snapshot.* (Retrieved October 8, 2013 from http://www.progressive.com/auto/snapshot.aspx)

Roess, R. P., Prassas, E. S., & McShane, W. R. (2011). *Traffic engineering*. Pearson.

Sinnott, R. W. (1984). Virtues of the haversine. *Sky and telescope*, *68*(2), 159.

Song, C., Qu, Z., Blumm, N., & Barabsi, A.-L. (2010, February). Limits of predictability in human mobility. *Science*, *327*(5968), 1018-1021. doi: 10.1126/science.1177170

StateFarm. (2016). *Drive safe & save.* (Retrieved June 2, 2016 from https://www.statefarm.com/insurance/auto/discounts/drive-safe-save)

Troncoso, C., Danezis, G., Kosta, E., Balasch, J., & Preneel, B. (2011, September). Pripayd: Privacy-friendly pay-as-you-drive insurance. *IEEE Trans. Dependable Secur. Comput.*, *8*(5), 742–755. Retrieved from `http://dx.doi.org/10.1109/TDSC.2010.71` doi: 10.1109/TDSC.2010.71

Tsai, J. Y., Kelley, P., Drielsma, P., Cranor, L. F., Hong, J., & Sadeh, N. (2009). Who's viewed you?: the impact of feedback in a mobile location-sharing application. In *Proc. of chi'09.*

U.S. Department of Transportation. (2013). *Federal motor carrier safety administration.* (Retrieved October 2, 2013 from http://www.fmcsa.dot.gov/rules-regulations/administration/fmcsr/fmcsrruletext.aspx?reg=393.52)

Wahlstrm, J., Skog, I., Rodrigues, J. G. P., Hndel, P., & Aguiar, A. (2016, Sept). Map-aided dead-reckoning using only measurements of speed. *IEEE Transactions on Intelligent Vehicles*, *1*(3), 244-253. doi: 10.1109/TIV.2017.2657383

Wang, D., Pedreschi, D., Song, C., Giannotti, F., & Barabasi, A.-L. (2011). Human mobility, social ties, and link prediction. In *Proc. of kdd '11.*

Zang, H., & Bolot, J. (2011). Anonymization of location data does not work: a large-scale measurement study. In *Proc. of mobicom '11.*

Zavoli, W. B., & Honey, S. K. (1986, May). Map matching augmented dead reckoning. In *Vehicular technology conference, 1986. 36th ieee* (Vol. 36, p. 359-362).

Zhao, L., Ochieng, W. Y., Quddus, M. A., & Noland, R. B. (2003). An extended kalman filter algorithm for integrating gps and low cost dead reckoning system data for vehicle performance and emissions monitoring. *The journal of Navigation*, *56*(02), 257–275.

Zheng, Y., Chen, Y., Li, Q., Xie, X., & Ma, W.-Y. (2010, January). Understanding transportation modes based on GPS data for web applications. *ACM Trans. Web*, *4*, 1:1–1:36. Retrieved from `http://doi.acm.org/10.1145/1658373.1658374` doi: http://doi.acm.org/10.1145/1658373.1658374

Zheng, Y., Zhang, L., Ma, Z., Xie, X., & Ma, W.-Y. (2011, February). Recommending friends and locations based on individual location history. *ACM Trans. Web*, *5*, 5:1–5:44. Retrieved from `http://doi.acm.org/10.1145/1921591.1921596` doi: http://doi.acm.org/10.1145/1921591.1921596

Zhou, X., Demetriou, S., He, D., Naveed, M., Pan, X., Wang, X., . . . Nahrstedt, K. (2013). Identity, location, disease and more: Inferring your secrets from android public resources. In *Proc. of ccs '13* (pp. 1017–1028). New York, NY, USA: ACM.

Ziebart, B. D., Maas, A. L., Dey, A. K., & Bagnell, J. A. (2008). Navigate like a cabbie: probabilistic reasoning from observed context-aware behavior. In *Proc. of ubicomp '08.*