A Framework for Interactive Geospatial Map Cleaning using GPS Trajectories

by

Nikhil Vementala

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved July 2017 by the
Graduate Supervisory Committee:

Paolo Papotti, Co-Chair
Mohamed Sarwat, Co-Chair
Kasim S. Candan

ARIZONA STATE UNIVERSITY

December 2017

ABSTRACT

A volunteered geographic information system, e.g., OpenStreetMap (OSM), collects data from volunteers to generate geospatial maps. To keep the map consistent, volunteers are expected to perform the tedious task of updating the underlying geospatial data at regular intervals. Such a map curation step takes time and considerable human effort. In this thesis, we propose a framework that improves the process of updating geospatial maps by automatically identifying road changes from user generated GPS traces. Since GPS traces can be sparse and noisy, the proposed framework validates the map changes with the users before propagating them to a publishable version of the map. The proposed framework achieves up to four times faster map matching performance than the state-of-the-art algorithms with only 0.1-0.3% accuracy loss.

This is dedicated to my loving parents.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my parents Vemmentalala Baswarajaiah and Vemmentalala Swarna who have made me who I am today. Without your love and moral support, this would have been never possible.

I would like to express my deep sense of gratitude to Dr. Paolo Papotti, for his continued support, encouragement, and allowing me to carry out my thesis and research under him. He was always available whenever I ran into a trouble spot or had any question about my thesis. I would also like to thank Dr. Mohamed Sarwat and Dr. K. Selcuk Candan for showing interest in my work and sharing their valuable comments about my work and closely following up with the progress of my thesis.

I would like to thank Dr. Dragan Boscovic for attending my thesis defense and for taking time in reviewing my thesis and providing feedback on my work.

I would like to thank all my friends Venkatesh Burli, Akshay Nayaknur and colleagues at Dr. Paolo Papotti's lab for making my journey of research exciting and making it a great learning experience. I am grateful to my family members Prashanth, Niveditha, Vikas Teja and Niharika for their unfailing support and continuous encouragement. I would like to acknowledge my academic advisor Arzuhan Kavak, for her valuable suggestions and help throughout my Masters.

TABLE OF CONTENTS

	Page
LIST OF TABLES	v
LIST OF FIGURES	vi
CHAPTER	
1 INTRODUCTION	1
2 RELATED WORK	4
2.1 Map Matching	4
2.2 Map Generation	8
2.3 Data Cleaning	10
3 METHODOLOGY	11
3.1 Definitions	11
3.2 Map Matching Algorithm	14
3.3 Trace Analyser	16
3.4 Decision Maker	18
4 ANALYSIS AND RESULTS	19
4.1 Map Matching Algorithm	19
4.2 Analysis of Parameters	24
4.3 Comparison of Clustering Algorithms	27
4.4 End-to-End Framework	30
5 CONCLUSION AND FUTURE WORK	34
REFERENCES	37
APPENDIX	
A DATASETS	41
B OPENSTREETMAPS	43
C DEMONSTRATION OF MAP MATCHING ALGORITHM	45

LIST OF TABLES

Table	Page
3.1 Definition of Functions	16
4.1 Similarity Measurement of the Missing Roads	32

LIST OF FIGURES

Figure		Page
1.1	Screenshots from Neis (2017) on 22-June-2017.	2
3.1	Architecture of Proposed Framework	12
3.2	LineString	12
3.3	Block Diagram of Trace Analyzer	17
4.1	Runtime of MM Algorithms on Seattle Dataset(in secs)	20
4.2	Reported Error of MM Algorithms on Seattle Dataset	21
4.3	Precision (a) and Recall (b) of MM Algorithms on Seattle Dataset..	21
4.4	Runtime (in Secs) of Algorithms on Uic Dataset.....	22
4.5	Reported Error of Algorithms on UIC Dataset	22
4.6	Precision of Algorithms on UIC Dataset	22
4.7	Recall of Algorithms on UIC Dataset	23
4.8a	Analysis of Route Numbered 563 of Chicago Dataset	23
4.8b	Analysis of Route Numbered 731 of Chicago Dataset	24
4.9	New Proposed Roads for Different Values of ‘min_cluster_size’	24
4.10	New Proposed Roads for Different Values of ‘min_cluster_size’	25
4.11	Length of Road Correctly Fetched for various Threshold ε	26
4.12	Length of Road Incorrectly Fetched for various Threshold ε	26
4.13	Comparison of Traclus and Kde Algo for Chicago Data(Correct Length) with ‘min_cluster_size’ set to 12	28
4.14	Comparison of Traclus and Kde Algo for Chicago Data(Incorrect Length) with ‘min_cluster_size’ set to 12	28
4.15	Comparison of Traclus and KDE Algo for Chicago Data(Correct Length) with ‘min_cluster_size’ set to 8	29

Figure	Page
4.16 Comparison of Traclus and KDE Algo for Chicago Data(Error Length) with ‘min_cluster_size’ set to 8	29
4.17 Comparison of Traclus and KDE Algo for Tempe City (Correct Length)	30
4.18 Deleted Roads from Chicago Map	31
4.19 GPS Traces From UIC Dataset (Day 1).....	31
4.20 Future Roads (in Blue)	32
4.21 Potential New Roads (Pink)	32
5.1 Limitations of the Framework.....	35

Chapter 1

INTRODUCTION

Maps are graphical representations of geographical areas. The representation is always an abstraction of the reality. The accuracy of maps has become more desirable. Information has been widely collaborated among different sources constantly to keep maps up to date. Maps on papers had many limitations like accuracy, scale, projection information and the amount of information to be shown on the maps. In the past decade, digital maps have gained popularity. Digitalization of maps has made it easier to distribute and update them to a larger community. Due to their wide usage, accuracy and real-time information of maps have become paramount. Different techniques were employed to make the digital maps up-to-date. Satellite imagery is one of the most widely-used technique to keep maps up-to-date. Updating maps involves a lot of computing power accompanied with skilled human supervision and due to the non-commercial demand of maps of the remote places, they are not regularly updated.

In recent times, Volunteer geographic information (VGI) systems e.g., OpenStreetMap(OSM) (Wiki, 2014) have gained a lot of attention due to its nature of building and collection information about maps. OpenStreetMap is an open source project by OSM Foundation which provides maps of the globe for free. Information is collected from various techniques like High-resolution Satellite/Aerial Imagery, tracing GPS points uploaded by users, hand-drawn maps of cities and field papers which are provided by a huge volunteer base. Using GPS traces to map the roads or entities around it, is one of the popular technique employed by OSM volunteers. To keep a map consistent volunteers are expected to iterate the process of updating maps at regular intervals which takes time and requires considerable human effort. OSM aims to map the entire planet and are actively

improving the coverage and accuracy of the maps. There are few third-party services that attempt to clean the OSM data in an interactive fashion. For instance, MapRoulette (MapRoulette, 2017) and Kort Game (KortGame, 2017) gamify the process of OSM bug fixing. MapCraft (MapCraft, 2017) divides the map into small pieces and enables real-time collaboration to edit maps. AddressHunter (AddressHunter, 2017) improves the address coverage of the places on the map. KeepRight (KeepRight, 2017) detects errors like non-closed areas, layer conflicts, and missing name tags.

Despite these efforts, the problem of automatic detection of new roads is still to be addressed and has not received much attention. In tackle this problem, we may encourage users to update the map manually or ask them to submit GPS traces of a place and we do the work of detecting and forming new roads and only query them with simple questions to know if the new roads are correct. The latter method is efficient and causes less burden to the user.

Figure 1.1: Screenshots from Neis (2017) on 22-June-2017.

Figure 1.1 shows statistics about the contribution of OSM volunteers during a period of a week (14 Jun 2017 - 21 Jun 2017) and is obtained from Neis (2017). The OSM database is growing with volunteers adding and modifying a lot of data. Although the number of roads modified is not as significant as a number of new roads. It shows that updating maps too requires a significant effort from the volunteers. Updating road information may include updating properties related

to a road like a name, maximum speed and modifying the geometric properties of the road. This thesis proposes a viable solution which will help in keep maps up to date and distribute the load of updating these maps into smaller tasks which can be easily managed by volunteers.

In this thesis, we present a framework which automates the process of recognizing new roads from GPS traces collected from the users. It matches GPS traces against the existing maps and identifies the outliers in the trajectories. It queries the user only if any parts of GPS traces is not found on the existing road network frequently. We have proposed a new algorithm for Map Matching and compared its performance with the existing state-of-the-art map matching algorithms. The map matching algorithm returns the segments of GPS trace known as outlier points which do not match with any of the roads in the existing network. We cluster the similar outlier points to form a new road. In ideal cases, a GPS trace will have the coordinate values depicting the actual path traveled by the vehicle. But in reality, a GPS trace is not very accurate for reasons like weak GPS signal strength and to support low power consumption. So we need an efficient way to match and recognize new roads. These new roads which are formed are maybe because of the noise present in the GPS trace. So with that being said, we introduce a person in this process, who can help in classifying a new road as legitimate or not. A new road classified as legitimate is added to the database or else it will be discarded. We also evaluate the effectiveness of the framework. We are also glad to say that our work has been accepted by IWCTS workshop, ACM SigSpatial 2017 (Vementala *et al.*, 2017).

The outline of the thesis is as follows. In Chapter 2, we will discuss the existing methods for map matching and map generation using GPS traces. In chapter 3, we will present the framework and discuss each component in details. The implementation details, the detailed analysis of the framework are presented in Chapter 4 and the conclusion and future work are given in Chapter 5.

Chapter 2

RELATED WORK

As discussed in the Introduction, identifying the segments of GPS which are not part of the existing road network, is one of the crucial tasks. It is followed by grouping similar unmapped GPS traces to form a new path and then query the users to know if the proposed road be a new extension to the existing road network. In this chapter, we will discuss prerequisites and different state-of-the-art methods to solve the above problems.

2.1 Map Matching

Map Matching algorithms are used to map a given location data to a spatial road network. In this section, we will discuss different existing map matching algorithms and their applicability to our current idea. Map-matching algorithms can be of two types: Online, and Offline algorithms (Pereira *et al.*, 2009). Online algorithms work on a real-time basis where the responsiveness of the algorithm is more important than correctness. Real-time GPS navigation is a good example for an online algorithm. Offline algorithms work in a post-processing fashion. Basically, they work on offline data, i.e. the information about entire trajectory is available. The offline algorithms stress on the accuracy of map matching as they have time and complete information about the route. This kind of processing is useful in analyzing traffic patterns which will help in better planning of public transportation or to predict traffic in a city. Most of the map matching algorithms assume that the road network is completely available, this leads to unexpected behavior when the trajectory goes off road or when the map is incomplete.

In Pereira *et al.* (2009) the author classifies map matching algorithms into four different groups depending on the technique used to perform the match. They are

geometric, topological, probabilistic and other advanced algorithms.

Geometric algorithms are based only on the match of the closest road segment for a given point. Since data are prone to noise, this method may lead to erroneous results, as it does not consider the continuity of road network. In Bernstein and Kornhauser (1998) and White *et al.* (2000), the authors discuss different types of geometric algorithms. In Geometric point-to-point matching, the given GPS point is matched with the nearest points that represent a road on the map. This algorithm is highly sensitive to the way the spatial network is implemented on the map. The roads which are represented with a high number of intermediate points are likely to match with a given point compared to the roads represented with a less count of GPS points. In point-to-curve matching, the given GPS point is matched to the road whose distance from the projection of a given point P is minimum. Using this method alone may not provide us the correct road, as it does not consider historical information about the other points in the trajectory and the closest point does not always match to the correct road. In curve-to-curve matching, the segment of road network which is closest to a part of GPS trajectory are considered as final links, however this method is sensitive to the noise in the data.

Topological algorithms consider additional properties like underlying structure and continuity of the road network in order to match a GPS point to the road. Yu (2006) and Quddus *et al.* (2003) have developed topological map matching algorithms. “Probabilistic algorithms use a region, an ‘error region’ which is usually an ellipse or a rectangle to match a given point” (Pereira *et al.*, 2009). Croyle *et al.* (1999) used this technique to map positions from a DR sensor to a map, and Zhao (1997) introduced this technique to map a GPS point and suggested the error variance can be obtained from regular GPS error rate. Other advanced algorithms use different techniques like Kalman Filters (Krakiwsky *et al.*, 1988; Obradovic *et al.*, 2006), Hidden Markov Models (Newson and Krumm, 2009), Multiple Hypothesis

techniques (Pyo *et al.*, 2001), Bayesian Inference and other advanced techniques.

Two map matching algorithms are discussed in detail which are used in my thesis. In Marchal *et al.* (2004), the author presents a topological Map Matching Algorithm called ‘Marchal’ which considers the nearest roads as a match and topology of the road network. The algorithm begins by identifying ‘N’ nearest roads for a given GPS point and assigns a score to it based on the distance between the road and the respective point. A road which is nearer will have a low score compared to the road which is farther away. The problem of map matching presented as finding a path along the GPS trajectory with the lowest score possible. This method only considers the next links in the continuity of the network topology. Since the mapping criteria is solely based on the nearest road on the network of a point, on denser road networks and when the noise of GPS points is high, the final path may not be accurate. This method is vulnerable to the noise present in the data. It always includes N nearest paths rather than the actual nearest path which is within a certain radius of a point; it is possible that we might include roads which are remotely related.

Map matching algorithms based on Hidden Markov Models (HMMs) by Liang *et al.* (2016); Newson and Krumm (2009) have set a benchmark for high accuracy and are more robust to noisy data compared to other methods. This is one of the reasons for us to choose HMM algorithm to compare our proposed algorithm. The problem of map matching also suits the HMM model well. The GPS points of the trajectory are Observations and the probable roads on the network are States. Emission probability provides us the probability of the road to be matched to the given point. The continuity of the road network determines the state transition probability. Road segments which share an endpoint have higher transition probabilities compared to the non-adjacent roads. The Viterbi algorithm returns the Viterbi path which is the most likely sequence of the hidden states. The Viterbi path is the final sequence road segment on the map. Only roads within a certain

distance ‘D’ of a given point (Observations) are considered as States, and emission probability is calculated for each of them. This will eliminate the unnecessary states which will have a very low emission probability and does not contribute to the final path. The roads which are nearer have a higher probability which indicates it is highly likely to be part of the final path. The second important factor is “transition probability,” which emphasizes on the continuity of the road. The Viterbi algorithm makes use of these two probabilities and discovers the best path satisfying these two properties.

However, one of the few drawbacks of this method is that calculating transition probability involves calculating the shortest path between two GPS coordinates, which is a resource intensive task. The number of Observations increases with the increase of radius around a given GPS point. Calculating transition probability between the GPS points is a tedious task and requires much computational power. Different techniques were proposed to increase the speed of the process. Song *et al.* (2012) suggest building a global index to index road segments and execute the HMM algorithm in a multi-core CPU to take advantage of multi-threading. This method will effectively reduce the time taken when processing batch GPS traces, but it does not address the issue of calculating distances between points. Liang *et al.* (2016) suggests considering the n-connection adjacency matrix which will store the number of connections away in place of distance. This is ambiguous as the length of the road is not constant in openstreetmap usecase. Sunderrajan *et al.* (2014) attempt to decrease the latency of HMM by building a Quad Tree to quickly find the edge and use the arc-flag approach to minimize the number of edges evaluated by the Dijkstra’s Algorithm. Since, we assume that the topology of the map is not constant, pre-calculating the shortest distance may not be a viable solution to increase the speed of the process if the rate of adding new roads is high.

Different techniques are used to evaluate map matching algorithms in previous

literature. In this thesis, we compare the map matching algorithms using some existing methods like Runtime, Reported Error (Newson and Krumm, 2009), Correct road match ratio or Precision (Jagadeesh *et al.*, 2004; White *et al.*, 2000), and Recall as discussed in the previous work. The HMM algorithm discussed in Newson and Krumm (2009), quantifies the correctness of the map matching algorithm based on the route mismatch fraction. Route mismatch fraction or Reported error is the ration of length of road erroneously added and subtracted to the resultant path to length of the correct route.

$$ReportedError = \frac{d_- + d_+}{d_0}$$

where d_- and d_+ denotes length of the road erroneously subtracted and added respectively when the path is returned by the map matching algorithm. d_0 is the length of the correct route also known as Ground Truth.

Other existing map matching algorithms like Jagadeesh *et al.* (2004) and White *et al.* (2000) use precision or correct-road-matching ratio as a metric to evaluate. The precision of the algorithm measures the degree of accuracy i.e., it shows how much of the path returned by the algorithm is correct. The recall of an algorithm measures the degree of correctness meaning that it shows how much of the path returned by the algorithm is correct compared to the ground truth.

2.2 Map Generation

As discussed in Chapter 1, the digitization of Maps has received significant attention over the years. The two means of representing a digitized map using software include Vector approach and Scalar approach(*). In the Vector approach, the LineString or coordinates on the map are line or points respectively in the 2D plane space. Geometric principles and rules are applied to perform any operations. In Raster format(*), we assume the map as a digital image plane space of two-dimensional array of high resolution. The resolution of the maps will decide the

accuracy and fineness. Image processing algorithms are useful in performing any operations like smoothing or clustering.

In Ahmed *et al.* (2014), the authors present different approaches of map construction from GPS data and also provide a comparison between them. The different approaches for map construction can be classified into three major groups: point clustering, incremental track insertion, and intersection linking. One of the two major approaches in the point clustering includes clustering of points in the dataset to identify the intersections. Then, the intersections are connected with each other according to the existing route information like vehicle heading (Edelkamp and Schrödl, 2003). In the second approach, the entire trajectory is considered to retrieve the road network. Initially the GPS traces are processed to find the skeleton of the road using KDE methods, then the skeleton is used to extract the actual road information (Schroedl *et al.*, 2004).

In Incremental track insertion, the map is built incrementally as the new traces appear. It may start from scratch or can be an update for the existing topology. In Edelkamp *et al.* (2008), the author proposes a fully autonomous map generation method, which tries to build a map from scratch. It accepts the GPS traces from the user and filters them for noise, then runs a Map Matching algorithm to check if the current trace is a new road or matches with any existing routes. Aggregating the newly formed segments helps in recognizing new paths, and roads are updated if any segment of GPS trajectory matches with the present road network. Aggregation of traces can be performed in two ways, incremental and batch-based. In incremental aggregation, the map is built as new traces appear. “We can see a sequence of snapshots in time: the maps grow as new traces come” (Edelkamp *et al.*, 2008). In Batch-based approach, all the traces are collected and then similar GPS traces are clustered and aggregated to form a new road in a batch. Authors Bruntrup *et al.* (2005), also present a framework for map generation with the help of GPS traces which make use of AI algorithms to infer

the road geometry. All the work above assumes that the GPS data are correctly filtered and the maps are complete, which is not necessarily true. The generated map is influenced by the noise present in the GPS. There is a need of bringing in an expert or an oracle who has knowledge of roads in a locality so that roads are edited, or adjusted to represent the actual roads.

2.3 Data Cleaning

Work on data cleaning has focused on relational data by exploiting different types of dependencies, such as functional dependencies (Cong *et al.*, 2007) and denial constraints (Chu *et al.*, 2013). These methods rely on user defined rules, that model patterns and constraints that must apply on the data. If a set of values violate a rule, it determines that there must be an errors in the data, and repair algorithms try to automatically identify it and fix it. However, such approaches based on rules do not directly apply for spatial data. This is especially true for the problem tackled in this thesis, since new roads cannot be inferred from existing ones, and road closure cannot be determined by an existing topology.

Chapter 3

METHODOLOGY

As discussed in Chapter 1, updating maps is a crucial step and requires a lot of effort, time, and skill of humans. In this thesis, we aim to automate this process of updating maps with minimal human involvement. Specifically, updating maps by exploiting a stream of GPS trajectories which are submitted by the user. In this chapter, we will discuss the workflow and key components of this framework. Figure 3.1 shows the three major components of the model. First, the map matching algorithm will take the GPS trajectory as an input and returns the mapped and unmapped parts of the trajectory. Second, the trace analyzer will analyze all the unmapped traces of different trajectories and cluster them based on their similarities. It will take the unmapped parts from the map matching algorithm as input, and when similar unmapped parts from a multiple trajectory reach a significant number, it creates a representational line which is an average of all the paths clustered. This new representational line is the new road that the framework is going to flag, and its correctness and accuracy completely depend on the trajectory data. Third, the Decision Maker will identify the new representational lines created and will query a volunteer about it. Based on the responses obtained from the users, it will decide to add this new path to the existing road network or it will discard the proposed road.

3.1 Definitions

Definition 3.1.1 (*GPS co-ordinate*) A GPS coordinate ‘ P ’ is a point on the earth space, which can be used to uniquely identify the location. It is in (x, y) format where x is a latitude and y is longitude.

Figure 3.1: Architecture of Proposed Framework

Definition 3.1.2 (*Road*) A road ‘ R ’ can be defined as a set of coordinates ‘ P ’ represented as a single *LineString* which depicts the roads, paths in real life. It can be represented as $R \in \{P_1, P_2, \dots, P_n\}$.

Figure 3.2: LineString

Above figure 3.2 shows LineString. Each road is identified using unique identifier ‘ I ’ and also has properties like road/street name, cost i.e., length of the road, reverse-cost (attribute used to enforce one-ways on the map). For a bidirectional road, cost and reverse-cost are same, whereas the unidirectional road will have a very high reverse_cost.

Definition 3.1.3 (*Road Network*) A road network ‘ N ’ is a directed graph with a set of roads ‘ R ’. It can be represented as $N \in \{R_1, R_2, \dots, R_n\}$.

Definition 3.1.4 (*GPS Trajectory*) A GPS trajectory ‘ T ’ is the set of GPS coordinates ‘ P ’ obtained from a GPS device of a moving vehicle. It is represented as $T \in \{P_1, P_2, \dots, P_m\}$

Definition 3.1.5 (*Ground Truth*) Ground Truth ‘ GT ’ for a trajectory ‘ T ’ is the actual path traveled on the road network ‘ N ’ by the vehicle from which the trajectory is collected. Say, if each road on the network is identified uniquely by ‘ I'_x ’, then it is denoted as $GT \in \{I_1, I_2, \dots, I_n\}$

‘ I ’ is the unique identifier for a road. Usually, ground truth is calculated by visually mapping the trajectory to the road network or rarely the trajectory uploader may provide it.

Definition 3.1.6 (*Match*) A GPS coordinate ‘ P ’ of a Trajectory ‘ T ’ is said to be matched to a road ‘ R ’ of the road network ‘ N ’ if the Euclidean distance between the ‘ P ’ and the projection of ‘ P ’ i.e., ‘ p ’ on the nearest segment of ‘ R ’ is less than a given threshold ε .

Definition 3.1.7 (*Map Matching*) It is the process of matching a sequence of GPS co-ordinates to the road network such that every point is matched to at most one nearest road.

A formal definition of map matching is clearly described in Bernstein and Kornhauser (1998). An ideal map matching method will return the path which is identical to the Ground Truth.

3.2 Map Matching Algorithm

In this section, a new topological Map Matching algorithm is proposed. It considers the continuity in the matching to avoid erroneous paths. In Marchal *et al.* (2004), the authors have introduced a new map matching algorithm which focuses on speed rather than accuracy. They have achieved this by reducing the number of computations required to decide if a particular co-ordinate belongs to the trajectory or not. The proposed new Matching Algorithm requires less computational power and also considers the continuity of the path to decide the complete resultant path along the trajectory.

The algorithm takes the trajectory ‘G’ and road network ‘R’ as input. $|G|$ represents the number of points present in the trajectory ‘G’. It starts with finding the nearest roads within the radius of ε for the first point in the trajectory. The closest one will be considered as the resultant path. Then it checks if the next point in the trajectory is inside the same radius of the current nearest road. It will continue until it finds a point P_{i+k} ($i+k < |G|$) which does not match to the current road. It will again try to find out nearest paths for point P_{i+k} ($i+k < |G|$). The one which is adjacent i.e., share an edge with the previous resultant path, will be considered as a match. In some cases, when the density of points in the trajectory is low, we might match GPS points to two different non-adjacent roads. In such cases, we can try to find if the roads are separated by one or two links. If this is true, we may include those paths else we can consider P_{i+k} ($i+k < |G|$) and its previous point as outliers. These outliers are monitored in the next step to check if there is any other similar outliers are returned for any other trajectories.

The algorithm 1 shows the new proposed map matching algorithm. The definition of functions used in the algorithm discussed below. The working of the algorithm is clearly explained in appendix C.

Algorithm 1 Map Matching Algorithm

INPUT : trajectory data
OUTPUT : (segments of road along trajectory, outlier points)

```
1: function MAPMATCHINGALGO(route_data) ▷ list of 'lat, long'
2:   finalPath =  $\emptyset$ 
3:   outlierIds =  $\emptyset$ 
4:   nearByRoads = getNearestRoadLinks(co_or)
5:   if nearByRoads  $\neq$  null then
6:     add best one from nearByRoads to the finalPath
7:   for co_or in route_data[1 : len(co_or)] do
8:     if (isPointOnLine(finalPath[-1], co_or) then
9:       continue;
10:    else
11:      nearByRoads = getNearestRoadLinks(co_or)
12:      if nearByRoads = null then
13:        addco_ortotheoutlierIds
14:      else
15:        nearestRoad = nearByRoads[0]
16:        if finalPath =  $\emptyset$  then
17:          add nearestRoad to finalPath
18:        else
19:          if areAdjacent(finalPath[-1], nearestRoad) then
20:            if areAdjacent(finalPath[-2], nearestRoad) then
21:              Replace finalPath[-1] with nearestRoad
22:            else
23:              append nearestRoad to the finalPath
24:          else
25:            found = Check if any of nearByRoads match with
26:            if found = True then
27:              append matched road to the finalPath
28:            else
29:              for road in nearByRoads do
30:                parted = checkIfParted(finalPath[-1], road)
31:                if parted  $\neq$   $\emptyset$  then
32:                  add the intermediate path to the finalPath
33:                  found = True
34:                if found = False then
35:                  gap detected, flag for examination
36:  return(finalPath, outlierIds)
```

getNearestRoadLinks()	<i>Takes</i> co-ordinate as input <i>Returns</i> all the roads around the given point within radius.
areAdjacent()	<i>Takes</i> two roads as input <i>Returns</i> if the two road share a common vertices.
checkIfParted()	<i>Takes</i> two roads as input <i>Returns</i> if the two road are seperated by one or two roads.

Table 3.1: Definition of Functions

3.3 Trace Analyser

Trace analyzer is responsible for clustering trajectories and build representational lines if they reach certain threshold and proposes it to the user to validate if it is a new road. Map Matching Module will return segments of road traveled by GPS trajectory and outlier co-ordinates. The Trace Analyser will form a LineString out of outlier points. The unmapped LineStrings are compared with each other for similarity and examined for any pattern or evidence required to infer a new road. To detect a new path, we need to find a particular GPS segment which is not part of the actual road network and appears frequently from many GPS trajectories. Once we cluster all the similar unmapped LineStrings, there should be an effective way to cluster similar trajectories into a group and build a representative trajectory to be put forward as a new road. First, we initialize a cluster in cluster manager and build a Maximum Bound Region (MBR) with a width of 20m on each side of the outlier LineString. When a Map Matching algorithm returns a new outlier LineString, we check if it matches with any of the existing clusters in cluster manager. In that case, it is added to that cluster or new cluster is initialized. If many trajectories are in the cluster, this serves as evidence for a new road on the map. Thus, to decide if a specific cluster should be considered to build a representational line, we need to have a parameter

‘min_cluster_size’. If the number of unmapped trajectories in the cluster reaches ‘min_cluster_size’, it is considered to be a potential new road which is marked as ‘future_roads’ by the framework.

Figure 3.3: Block Diagram of Trace Analyzer

One of the effective means of building a representational line is discussed in Lee *et al.* (2007). The authors have proposed *TRACCLUS*, a trajectory clustering algorithm. It is a *partition-and-group* framework. First, the partition framework partitions trajectory into a set of line segments and then groups similar line segments using a density-based clustering method to build a representational line. We have implemented this algorithm for clustering without any changes. All the parameters of the paper are set to the optimal values as suggested by the paper.

In the next step, the representational lines which are marked as ‘future_road’ are checked if they intersect with any junction or a nearby road. If any end of the representational line is within a 10 meter radius of an existing road network, then they are joined to complete the road network. Such are marked as ‘potential_roads’ which are forwarded to the Decision Making module to get user opinion.

3.4 Decision Maker

The decision maker accepts aggregated trajectories from the Trace Analyzer and will make a decision to add it to the road network by querying the user group. In order to make the framework robust to errors made by a single person, we chose to take a decision based on group opinion instead of one. The end-users can be the general crowd who have knowledge of that locality/city or in ideal situations the users who have uploaded the GPS trace. A user is allowed to either accept or reject the proposed changes. Decision to add or reject a change will be taken by considering the majority opinion in the group. If the newly proposed road is accepted by the user group, it is verified if it intersects with any of the existing roads in the network. If it intersects with any existing road, then the new road is split into two parts at the point of intersection and then added to the database. If the newly proposed road is rejected by the user, then the road will be saved to suppress future errors. We also extend the functionality of the decision maker to obtain more information from the user about the accepted or rejected change. For example, we can ask the user additional information like road name, number of lanes, and any restrictions upon accepting the road.

ANALYSIS AND RESULTS

In this chapter, we analyze the performance of the framework proposed in this thesis. To test the accuracy of the framework we compare its performance with the state-of-the-art HMM algorithm discussed in Newson and Krumm (2009). To test the efficiency of the framework, existing roads on the map are deleted and they are reconstructed using input GPS trajectories. The similarity of the proposed roads and deleted roads is calculated. We also demonstrate the effect of different parameters of the framework. We have performed all the experiments on real-world datasets. All the algorithms are implemented in Python using PostGIS, Shapely libraries for spatial operations. The tests are executed on system with Intel 64-bit i5 processor and 8 GB Ram.

4.1 Map Matching Algorithm

In this section, we compare the proposed map matching algorithms with the Hidden Markov Models (HMM) based algorithm. HMM algorithm (Liang *et al.*, 2016; Newson and Krumm, 2009) have set a benchmark for high accuracy and are more robust to noisy data compared to other methods. As discussed in Section 2.1, calculating transition probability is a resource intensive task. In order to make map matching faster, we have built a quad tree index on the data as suggested in Sunderrajan *et al.* (2014) and also pre-calculated and cached the transitional probabilities required to process the trajectories. Time to pre-calculate the transition probabilities is not included in the following statistics.

Below, we present a comparison between both algorithms in terms of Reported Error, Precision, Recall and Runtime. For the experiments, we have considered

Figure 4.1: Runtime of MM Algorithms on Seattle Dataset(in secs)

two datasets: Seattle dataset [A.1] and Chicago Dataset[A.2]. Seattle dataset provided by Newson and Krumm (2009) and is used to compare the robustness of both the algorithms. The dataset is modified to simulate different sampling rates. Four different datasets are created with different sampling rates of 10, 20, 30, 40 meters.

Figure 4.1 shows the runtime of both the algorithms. At high sampling rates like 10m, HMM algorithm takes the highest time to find the complete path on the map. But, the execution time decreases as the sampling rate increases, this is because as the sampling rate decreases the number of points decreases i.e., the number of states in HMM decreases. So the time taken to calculate the transition probability between different node is less. However, this is opposite in the case of proposed algorithm. High sampling rate between 10 to 30 has minimal effect on the execution time. This is because, if there are many points which are closer to a single road segment then they are ignored until the point that is not in range of the threshold distance of the current road segment. At a sampling rate of 40 meters, the points are distant from each other. In some situations, when the roads matching two consecutive points are not adjacent, the proposed algorithm will explicitly query the road network for the road segments which are in between them, and this is an expensive operation.

Figure 4.2: Reported Error of MM Algorithms on Seattle Dataset

Figure 4.3: Precision (a) and Recall (b) of MM Algorithms on Seattle Dataset

Figure 4.2 shows the reported error of both the algorithms. Figure 4.3 (a) shows the precision of both the algorithms. Figure 4.3 (b) shows the recall of the algorithms. The high recall and a considerably low precision values say that the path returned by the proposed algorithm may have extra roads than required.

The Map matching algorithms are also tested on the UIC dataset [A.2]. Ten different GPS traces are chosen randomly for the experiment. From figure 4.4, 4.6, 4.7, we can say that the proposed map matching algorithm is faster in identifying correct roads compared to HMM algorithm with minimal loss of quality. One of the reasons for low precision values for the proposed framework is, it classifies a given point as an outlier if it's distance from the underlying road network is more

Figure 4.4: Runtime (in Secs) of Algorithms on Uic Dataset

Figure 4.5: Reported Error of Algorithms on UIC Dataset

Figure 4.6: Precision of Algorithms on UIC Dataset

Figure 4.7: Recall of Algorithms on UIC Dataset

Figure 4.8a: Analysis of Route Numbered 563 of Chicago Dataset

than the given threshold ε . But, high recall value indicates that the proposed map matching algorithm is good at identifying most of the ground truth. However, the low precision of the map matching algorithm will not negative affect on the framework as the framework depends on the users to validate the changes. Proposing false positives is given priority than missing the changes in the road. In figure 4.5, most of the value for reported error are comparable except for route number 724, 563 which are drastically different. Upon, visualizing the road network and GPS traces, we have observed that this error is because the road network has 2 lanes on the road and both the algorithms made mistake in two different instances to map to the correct lane. In figure 4.8a, the road network is depicted using two lanes, the GPS trace is spread across two lanes. Since, a good number of points are also present on the other lane the HMM algorithm includes the opposite path into the resultant path. In figure 4.8b, due to initial error the erroneous path (in red) is added to the resultant path.

Figure 4.8b: Analysis of Route Numbered 731 of Chicago Dataset

Figure 4.9: New Proposed Roads for Different Values of ‘min_cluster_size’

4.2 Analysis of Parameters

As discussed previously, the performance of the framework depends on effectively identifying the outlier GPS traces and ‘min_cluster_size’ that helps in deciding if a cluster of outliers can be considered to form a representational line. We analyze the working of framework by varying the value of ϵ for map matching algorithm and ‘min_cluster_size’ for trace analyzer. For higher values of ϵ , the outliers are matched to the nearest, which leaves less evidence about the outliers and low values of ‘min_cluster_size’, framework tends to propose even a small group of outliers as a new road. When this value is high, it delays until the count of each cluster reaches that value and then aggregates the outliers as a new road.

Figure 4.10: New Proposed Roads for Different Values of ‘min_cluster_size’

To run this framework, we have deleted few roads from Chicago map and have supplied GPS trajectories of three days as input. Then we have analyzed the roads proposed by the framework at different values of ‘min_cluster_size’ by fixing the value of ε to 20 meters. This experiment is repeated on the UIC data set with values of ‘min_cluster_size’ set to 4,8,12,16,20. Figure 4.9 and 4.10 shows the length of roads correctly and incorrectly fetched by the framework. We can observe that smaller values of ‘min_cluster_size’ are able to fetch the correct results earlier than other but the length of the roads incorrectly fetched for the same value is much higher later. The higher values of ‘min_cluster_size’ suppresses the noise effectively. By examining the figures 4.9 and 4.10, we can say that the optimal value for ‘min_cluster_size’ can be set to 12. This value should be adjusted according to the quality and frequency of the GPS trajectories. For example, if user frequently rejects proposed roads we can increase the value of ‘min_cluster_size’.

Now, we study the effect of ε on the framework. The value of ε is crucial to decide if a given GPS point should be considered as an outlier or not. So, we run the framework for different values of ε say 10, 20, and 30 (in meters).

Figure 4.11: Length of Road Correctly Fetched for various Threshold ε

Figure 4.12: Length of Road Incorrectly Fetched for various Threshold ε

Figure 4.11 and 4.12 show the length of road correctly and incorrectly fetched for different values of threshold ε respectively. The lower values of threshold can be used to get accurate results but they will classify even slightly noisy data as outliers and forward it to the Trace Analyzer, which will result in false positives. On the other hand, higher values of threshold can effectively suppress the false positives as they try to map them to the nearest road. By analyzing both the figures, we can say that optimal value of threshold ε can be set to 20 for the UIC dataset.

4.3 Comparison of Clustering Algorithms

In Trace analyzer module, we have used an existing clustering algorithms to form a representational line after we believe that there is enough evidence to infer presence of a new road. In order to identify the effectiveness of Traclus algorithm by Lee *et al.* (2007), we have compared it with another procedure to cluster trajectories i.e., by image processing method kernel density estimator with a Gaussian kernel Biagioni and Eriksson (2012). In this method, a group of outliers is transformed into a 2D image. The geometry plane is transformed into 2D pixels of 1x1 meters. A 2D histogram is then produced for the image considering the number of times a GPS trajectory has passed through that cell. A Kernel density estimator is applied to the image which smooths the noise present in the GPS. It is then converted back into the geometry plane and placed on the existing road network.

We have conducted the experiment by considering GPS trajectories of 4 consecutive days and divided each day into shifts of 3 hours. In figure 4.13, X-axis denotes the each shift for 4 consecutive days followed by number of trajectories available up to that shift and Y-axis indicates the length of the road correctly fetched by the respective algorithm. We have considered optimal smoothing parameter for KDE algorithm which is 32, the KDE algorithm performs best at this value and the minimum cluster size of our proposed framework was set to 12, which is an optimal value to get good results. From the figure 4.14, we can observe that Traclus performs slowly when compared to the KDE algorithm in the context of length of relevant road fetched. At shift S12, we can observe that path returned by the Traclus algorithm and KDE are same and constant. Even though the Traclus algorithm is slower than KDE at value 12, it suppresses the false positives effectively when compared to the KDE algorithm. Figure 4.14 explains the how sensitive is each algorithm to noise in the GPS data.

Figure 4.13: Comparison of Traculus and Kde Algo for Chicago Data(Correct Length) with ‘min_cluster_size’ set to 12

Figure 4.14: Comparison of Traculus and Kde Algo for Chicago Data(Incorrect Length) with ‘min_cluster_size’ set to 12

The experiment is conducted by setting the value of ‘min_cluster_size’ to 8. Lowering the threshold will help the Traculus framework to recognize new roads quickly but it allows the framework to propose some false positives as a new road. Figures 4.15 and 4.16 shows the results of the experiment. At value 8, The framework may propose new roads quickly, but also it increases the chance of false positives.

The same experiment is repeated with the dataset of the City of Tempe. In this experiment 9 roads are deleted from the existing road network. The Tempe

Figure 4.15: Comparison of Traclus and KDE Algo for Chicago Data(Correct Length) with ‘min_cluster_size’ set to 8

Figure 4.16: Comparison of Traclus and KDE Algo for Chicago Data(Error Length) with ‘min_cluster_size’ set to 8

dataset doesn’t have timestamps in the data, so we have considered 3 GPS trajectories of every shift up to 4 days. Out of 9 roads, both the algorithms were able to retrieve 6 roads correctly, after which due to noise in the data, irrelevant data was proposed as new roads. From the figure 4.17, we can observe that the for the given parameters, the Traclus algorithm returns the correct path much faster than the KDE algorithm. But KDE algorithm is better at handling noise. From this, we can say that when the expected noise in the data is low, the Traclus algorithm performs better than KDE. On the other part, KDE algorithm is more tolerant

Figure 4.17: Comparison of TracAlgo and KDE Algo for Tempe City (Correct Length)

to the noise and can be useful when dealing with high noise data.

4.4 End-to-End Framework

In this section, we present the end-to-end analysis of the framework. The performance of framework is obtained by analyzing how quickly and correctly the framework is able to return the new paths. In order to conduct the experiment, we have deleted some roads randomly from the map of Chicago and used UIC dataset. The UIC dataset is a collection of traces taken over a period of a month. We ran the framework and examined the new roads created every day. The quality of the newly created road can be best assessed by visual verification or by using quantitative methods like Hausdorff distance and Frechet distance.

Figure 4.18 shows the two roads (in blue color) that are deleted from the existing road network. Figure 4.19 shows the GPS traces from UIC dataset. These traces are passed as input to the framework. 26 trips are recorded on Day 1.

For this experiment we set values of ‘min_cluster_size’ to 8 and threshold to 20 meters. If this value is one, framework starts to consider the outlier line as the final path and marks it as a ‘future_road’ which may not be very accurate. On

Figure 4.18: Deleted Roads from Chicago Map

Figure 4.19: GPS Traces From UIC Dataset (Day 1)

Figure 4.20: Future Roads (in Blue)

Figure 4.21: Potential New Roads (Pink)

Road ID	Frechet Distance	Hausdroff Distance
2443	0.00245	0.00004
2625	0.00024	0.00024

Table 4.1: Similarity Measurement of the Missing Roads

the other hand, we can set to the higher value when we want to a more accurate new road. As you can see the potential new roads shown in Figure 4.21, correctly aligned to the existing road.

We use Hausdroff distance and Frechet distance to find out the similarity between the proposed road and actual missing road.

Definition 4.4.1 (*Hausdroff distance*) “The Hausdroff distance measures the distance between two sets of metric spaces. Besse et al. (2016) defines it as “For every point of set 1, the infimum distance from this point to any other point in set 2 is computed. The supremum of all these distances defines the Hausdorff distance.”

$$d_H(A, B) = \max \left\{ \sup_{a \in A} \inf_{b \in B} d(a, b), \sup_{b \in B} \inf_{a \in A} d(a, b) \right\}$$

It is the maximum of all the minimum distances that are calculated from every point in set 1 to any point in set 2. Hausdroff distance can be computed in $O(n^2)$ time.

Definition 4.4.2 (*Frechet distance*) *The Frechet distance is often described as “walking dog distance”. Imagine a dog and its owner walking on two separate paths and are connected by a leash. The frechet distance between those two paths is the shortest length of leash required for traversing the paths without backtracking.*

The Hausdorff and Frechet distance of the newly proposed roads are presented in table 4.1. The low values of both Hausdorff and Frechet distance indicate that the roads which are built are very close to the original road segments. This observation is supported by the figure 4.21

CONCLUSION AND FUTURE WORK

A novel approach to maintain the spatial maps is proposed in this thesis after thoroughly studying existing approaches in the field of map generation. Most of the approaches till now, build map from scratch by assuming that most of the GPS data is correct which emphasize less on maintaining a partially built map to serve accurate information. Thus, we propose a new approach to address this problem of updating the maps by exploiting the GPS trajectories to recognize new changes, with minimum human intervention.

In order to identify a new road from the GPS trajectories, we need to first identify if a segments of the GPS trace which are not part of any existing road network. For this, we needed to have an effective map matching algorithm which can map the given GPS trace to the existing network with minimum errors. There is some notable work done in the field of map matching. Most of the map matching algorithms try to assign a given GPS trace to the existing road network based on different methods like the continuity of road network, first nearest road, probability matching. In this thesis, we proposed a map matching algorithm which maps the GPS trace to the road network based on the information available of the previously matched road network. This algorithm returns the points which are not part of the existing road network which will be monitored by the proposed framework.

When a set of outliers is returned by the map matching algorithm, the proposed framework tries to identify if they can be aggregated to form a new road. It is very similar to the problem of building a map from scratch, where we start with a bunch of GPS points to build a road map. But in this scenario, we needed to find a new road which aligns properly with the existing road network than new stand-alone roads. Trajectory clustering is also an crucial step in order to obtain

high quality roads. We have also presented comparison of two existing trajectory clustering algorithms, viz. the first one is to identify the similar lines based on the distance and angle between them in a geometrical plane and, the second is to transform existing geometric plane into a two-dimensional image and to apply image processing methods to filter out the noise and extract patterns. We have presented both of them to understand their applicability to the current context. From the experiments, it can be inferred that if the noise in the GPS data is low, geometric algorithms are effective and can be used to detect the new changes quickly, Image processing algorithms are effective, but slow in recognizing new roads but they are effective in filtering out noisy data.

One of the limitations of this framework is, when the two roads are within the threshold distance radius of each other, it is not possible for the proposed algorithm to find a new road. Figure 5.1 illustrates this scenario.

Figure 5.1: Limitations of the Framework

In future, we plan to extend the framework to detect lanes on the proposed road. A preliminary solution can obtain road information from the end user who asserts the newly proposed road. Also, we can extend it to detect roundabouts. Currently, all roundabouts are detected as just intersections using this framework.

This framework is apt for the OpenStreetMap use case. OpenStreetMap is an open source maps portal, where a huge number of volunteers map the existing

world and upload their GPS traces. Volunteers usually make all changes to the map manually. In this thesis, we have shown that the workload of updating maps can easily be delegated into an interactive process which results in minimum human intervention and provides a secure way to maintain maps.

BIBLIOGRAPHY

- AddressHunter, URL <http://wiki.openstreetmap.org/wiki/AddressHunter> (2017).
- Ahmed, M., S. Karagiorgou, D. Pfoser and C. Wenk, “A comparison and evaluation of map construction algorithms”, CoRR **abs/1402.5138**, URL <http://arxiv.org/abs/1402.5138> (2014).
- Atev, S., G. Miller and N. P. Papanikolopoulos, “Clustering of vehicle trajectories”, IEEE Transactions on Intelligent Transportation Systems **11**, 3, 647–657, iD: 1 (2010).
- Bernstein, D. and A. Kornhauser, “An introduction to map matching for personal navigation assistants”, (1998).
- Besse, P. C., B. Guillouet, J.-M. Loubes and F. Royer, “Review and perspective for distance-based clustering of vehicle trajectories”, IEEE Transactions on Intelligent Transportation Systems **17**, 11, 3306–3317 (2016).
- Biagioni, J. and J. Eriksson, “Map inference in the face of noise and disparity”, in “Proceedings of the 20th International Conference on Advances in Geographic Information Systems”, pp. 79–88 (ACM, 2012).
- Boundless, “Boundless: Introduction to postgis: Geometries”, URL <http://workshops.boundlessgeo.com/postgis-intro/geometries.html#polygons> (2017).
- Bruntrup, R., S. Edelkamp, S. Jabbar and B. Scholz, “Incremental map generation with gps traces”, in “Intelligent Transportation Systems, 2005. Proceedings. 2005 IEEE”, pp. 574–579 (IEEE, 2005).
- Chu, X., I. F. Ilyas and P. Papotti, “Holistic data cleaning: Putting violations into context”, in “ICDE”, (2013).
- Chu, X., J. Morcos, I. F. Ilyas, M. Ouzzani, P. Papotti, N. Tang and Y. Ye, “Katara: A data cleaning system powered by knowledge bases and crowdsourcing”, in “Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data”, pp. 1247–1261 (ACM, 2015).
- Cong, G., W. Fan, F. Geerts, X. Jia and S. Ma, “Improving data quality: Consistency and accuracy”, in “VLDB”, (2007).
- Croyle, S., E. Larry and E. Sittaro, “Vehicle navigation system and method”, URL <https://www.google.com/patents/US5862511>, uS Patent 5,862,511 (1999).
- Edelkamp, S. and S. Schrödl, “Route planning and map inference with global positioning traces”, Computer Science in Perspective pp. 128–151 (2003).
- Edelkamp, S., D. Sulewski, F. C. Pereira and H. Costa, “Collaborative map generation—survey and architecture proposal”, Urbanism on track: application of tracking technologies in urbanism **1**, 161 (2008).

- Jagadeesh, G., T. Srikanthan and X. Zhang, “A map matching method for gps based real-time vehicle location”, *The Journal of Navigation* **57**, 3, 429–440 (2004).
- KeepRight, URL http://wiki.openstreetmap.org/wiki/Keep_Right (2017).
- KortGame, URL <http://www.kort.ch> (2017).
- Krakiwsky, E. J., C. B. Harris and R. V. Wong, “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”, pp. 39–46 (IEEE, 1988).
- Laboratory, B. N. S., “Bits - software”, URL <https://www.cs.uic.edu/bin/view/Bits/Software>, accessed on 30-June-2017 (2017).
- Lee, J.-G., J. Han and K.-Y. Whang, “Trajectory clustering: a partition-and-group framework”, in “Proceedings of the 2007 ACM SIGMOD international conference on Management of data”, pp. 593–604 (ACM, 2007).
- Liang, B., T. Wang, S. Li, W. Chen, H. Li and K. Lei, “Online learning for accurate real-time map matching”, in “Proceedings, Part II, of the 20th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining - Volume 9652”, PAKDD 2016, pp. 67–78 (Springer-Verlag New York, Inc., New York, NY, USA, 2016).
- Liu, X., J. Biagioni, J. Eriksson, Y. Wang, G. Forman and Y. Zhu, “Mining large-scale, sparse gps traces for map inference: comparison of approaches”, in “Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining”, pp. 669–677 (ACM, 2012).
- MapCraft, URL <http://wiki.openstreetmap.org/wiki/MapCraft> (2017).
- MapRoulette, URL <http://wiki.openstreetmap.org/wiki/MapRoulette> (2017).
- Mapzen, “Mapzen · an open, sustainable, and accessible mapping platform · mapzen”, URL <https://mapzen.com/>, accessed on 24-June-2017 (2017).
- Marchal, F., J. Hackney and K. Axhausen, “Efficient map-matching of large gps data sets-tests on a speed monitoring experiment in zurich”, *Arbeitsbericht Verkehrs-und Raumplanung* **244** (2004).
- Meratnia, N. and R. A. de By, “Aggregation and comparison of trajectories”, in “Proceedings of the 10th ACM international symposium on Advances in geographic information systems”, pp. 49–54 (ACM, 2002).
- Neis, P., “Osmstats - statistics of the free wiki world map”, URL <https://osmstats.neis-one.org/>, (clone and improvement of the former OSMStats on altogetherlost.com, offline since December 2014). (2017).
- Newson, P. and J. Krumm, “Hidden markov map matching through noise and sparseness”, pp. 336–343 (2009), URL <https://www.microsoft.com/en-us/research/publication/hidden-markov-map-matching-noise-sparseness/>.

- Obradovic, D., H. Lenz and M. Schupfner, “Fusion of map and sensor data in a modern car navigation system”, *The Journal of VLSI Signal Processing* **45**, 1, 111–122 (2006).
- Pereira, F. C., H. Costa and N. M. Pereira, “An off-line map-matching algorithm for incomplete map databases”, *European Transport Research Review* **1**, 3, 107–124 (2009).
- Pyo, J.-S., D.-H. Shin and T.-K. Sung, “Development of a map matching method using the multiple hypothesis technique”, in “Intelligent Transportation Systems, 2001. Proceedings. 2001 IEEE”, pp. 23–27 (IEEE, 2001).
- Quddus, M. A., W. Y. Ochieng and R. B. Noland, “Integrity of map-matching algorithms”, *Transportation Research Part C: Emerging Technologies* **14**, 4, 283–302 (2006).
- Quddus, M. A., W. Y. Ochieng and R. B. Noland, “Current map-matching algorithms for transport applications: State-of-the art and future research directions”, *Transportation Research Part C: Emerging Technologies* **15**, 5, 312–328 (2007).
- Quddus, M. A., W. Y. Ochieng, L. Zhao and R. B. Noland, “A general map matching algorithm for transport telematics applications”, *GPS Solutions* **7**, 3, 157–167, URL <https://doi.org/10.1007/s10291-003-0069-z> (2003).
- Schroedl, S., K. Wagstaff, S. Rogers, P. Langley and C. Wilson, “Mining gps traces for map refinement”, *Data mining and knowledge Discovery* **9**, 1, 59–87 (2004).
- Song, R., W. Lu, W. Sun, Y. Huang and C. Chen, “Quick map matching using multi-core cpus”, in “Proceedings of the 20th International Conference on Advances in Geographic Information Systems”, *SIGSPATIAL ’12*, pp. 605–608 (ACM, New York, NY, USA, 2012).
- Sunderrajan, A., H. Aydt, W. Cai and A. Knoll, “Map stream: Initializing what-if analyses for real-time symbiotic traffic simulations”, in “Simulation Conference (WSC), 2014 Winter”, pp. 2896–2906 (IEEE, 2014).
- Vementala, N., P. Papotti and M. Sarwat, “A framework for interactive geospatial map cleaning using gps trajectories”, in “Proceedings of the 10th ACM SIGSPATIAL International Workshop on Computational Transportation Science”, (ACM, 2017).
- White, C. E., D. Bernstein and A. L. Kornhauser, “Some map matching algorithms for personal navigation assistants”, *Transportation research part c: emerging technologies* **8**, 1, 91–108 (2000).
- Wiki, O., “Main page — openstreetmap wiki{,}”, URL http://wiki.openstreetmap.org/w/index.php?title=Main_Page&oldid=1060762, [Online; accessed 22-June-2017] (2014).
- Wikipedia, “Crowdsourcing”, URL <https://en.wikipedia.org/w/index.php?title=Crowdsourcing&oldid=786753896> (2017).

Yu, M., *Improved positioning of land vehicle in ITS using digital map and other accessory information*, Ph.D. thesis, The Hong Kong Polytechnic University (2006).

Zhao, Y., *Vehicle Location and Navigation Systems*, Artech House intelligent transportation systems library (Artech House, 1997), URL <https://books.google.com/books?id=sCVSAAAAAAAJ>.

APPENDIX A

DATASETS

In order to evaluate the framework, we need to run the algorithms on the real-world dataset. We came across few dataset which was useful in accessing our framework. All the maps for road network are taken from Mapzen (2017), an open and accessible platform which hosts the OSM maps data.

A.1 Seattle Dataset

This dataset is obtained from Newson and Krumm (2009). This data is collected from Seattle, Washington. The dataset consists single trip information collected for 2 hours and is about 80 km. It has 7351 time-stamped latitude and longitude pairs.

A.2 UIC Dataset

This dataset is provided by BITS Networked Systems Laboratory at UIC collected from UIC shuttles which travel between different UIC campuses. There are 888 trips taken over a period of one month. Dataset can be located at Laboratory (2017)

A.3 Tempe Dataset

The dataset is extracted from ‘mapmyrun’ website. This website contains the GPS traces of user running or walking tracks. The data is presented in a gpx format and consists of only a series of latitude and longitude values.

APPENDIX B
OPENSTREETMAPS

Digitalization of maps is a complex task. The unstructured form of maps data has made it hard to store and represent as it is in the database. Different kinds of approaches were used to store this unstructured data into traditional and NoSQL databases. Here, we are going to discuss one of the methods followed by “OpenStreetMap (OSM),” a collaborative and open source project supported by OpenStreetMap foundation founded in 2004. It provides maps of the road network in the world for free. It is Wikipedia for geospatial information. Volunteers collect map data from scratch by performing systematic ground surveys using tools such as handheld GPS unit, a camera, a notebook. The data is stored in a semi-structured format (XML) and distributed in different formats through OpenStreetMap Wiki (2014). Following primary entities are used to represent any object in the real world: Point, Line String, and Polygon. A Point represents a particular GPS coordinate on the earth’s surface defined by latitude and longitude. A Line String is a list of points used to represent a road, boundary. A Polygon is used to represent city limits and boundaries of a region. In OSM, a Node is equivalent to Point. An ID and coordinates will uniquely define each node. A Way is an ordered list of nodes between 2 to 2000 that define a line string. A relation is a multi-purpose data structure which contains other basic elements like nodes, ways, and relations. All items will have a tag which is a key, value pair which stores metadata about map objects like a unique identifier, name, street name, direction restrictions, type of building. We use the information given in the above format by OSM and transform which will help us to identify the changes in the map quickly.

OpenStreetMap stores and distributes data in different formats. OSM stores the data in relational tables in PostgreSQL. It has one table for each data primitive, with individual objects stored as rows [W]. The complete data dump is called planet.osm. We can also download data of any arbitrary region by specifying the coordinates of the bounding box. It is provided in either XML or Protocol Buffer Binary Format (PBF). We can use tools like Osmosis and osm2pgsql to load data from XML and PBF into PostgreSQL database.

APPENDIX C

DEMONSTRATION OF MAP MATCHING ALGORITHM

A detailed working example of map matching algorithm proposed in the thesis is presented below.

1. In figure below, The lines in blue color are part of road network. The points are co-ordinates of GPS trajectory. We fix the value of ε to 20 meters.

2. The GPS points are matched to their nearest roads and the resultant path is depicted in brown.

3. The Distance from fourth point to road is more than given ε , So the algorithm will not be able to find any nearest road to that point. That point is marked as 'outlier'.

4. Since the 4th point is marked as 'outlier', path taken to reach that point and the path taken after that point are considered as 'outlier path'. We indicate that using a dotted box in the figure below.

5. Then we search for the nearest roads of 5th point. We select the closest road and add it to the resultant path.

6. The nearest road for 6th point is searched on the road network. But that nearest road is not adjacent to the last path in resultant path. So, it explicitly queries the road network for any connecting roads.

7. There exists a road which connects both the roads, so we add it to the resultant path along the current nearest road. If that path doesn't exist then we will add the path between current path and last point into outliers. This procedure is continued for rest of the GPS trajectory.

8. At the end of the trajectory, map matching algorithm returns resultant path and the outlier points.