Road map generation and feature extraction algorithms from GPS trajectories and Trajectories Data warehousing

by

TARIQ ALSAHFI

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON
December 2020

Road map generation and feature extraction algorithms from GPS trajectories and Trajectories Data warehousing

The members of the Committee approve the doctoral dissertation of Tariq Alsahfi

Ramez Elamsri

Supervising Professor _____

Bahram Khalili _____

Leonidas Fegaras _____

David Levine _____

Dean of the Graduate School _____

To my mother, my father, my wife and my daughter Deem

ACKNOWLEDGEMENTS

In the name of Allah, the Most Gracious, the Most Merciful. Thanks to Allah above all for giving me the strength and time to complete this work. Thanks to my greatest teacher, prophet Mohammad, peace be upon him, who encouraged us to obtain knowledge from birth until death.

I would like to thank my supervisor, Professor Dr. Ramez Elmasri, who has been a great mentor to me since I started my program at UTA. I am honored to be one of his Ph.D. students. I really appreciated his continuous guidance, support, and motivation during my doctoral program. I would also like to thank my academic committee members: Dr. Bahram Khalili, Dr. Leonidas Fegaras, and Mr.David Levine, for being my teachers during my Master's and Ph.D. programs and for their insightful comments.

I would like to thank my parents, Mohammed Alsahfi and Nufue Alsahfi, for all they have provided me in life and are still providing, I would not have been able to complete this work without your continuous love and support. I would also like to thank my wife, Atheer Alsahafi, our daughter, Deem, for their support, patience, and encouragement during my program. Also, thanks to my brothers and sisters, Razina, Aliah, Shoker, Alia, Ali, and my older brother Shaker, may God have mercy on him, who left before seeing the completion of my degree, for their encouragement and support.

My sincere gratitude goes to the University of Jeddah and the government of Saudi Arabia for funding and supporting me during my ESL, Master's, and Ph.D. programs.

Last but not least, I would like to thank my friend Dr. Mousa Almotairi for his time and effort to evaluate and review my work. I would also to thank my friends Dr. Sakher Ghanem, Dr. Ali Alammari, Dr. Sami Alesawi, Bader Alshmaimri, and all members of the MAST lab.

December, 2020

ABSTRACT

Road map generation and feature extraction algorithms from GPS trajectories and Trajectories Data warehousing

Tariq Alsahfi, Ph.D.

The University of Texas at Arlington, 2020

Supervising Professor: Ramez Elamsri

Advanced technologies in location acquisition allow us to track the movement of moving objects (people, planes, vehicles, animals, ships, ..) in geographical space. These technologies generate a vast amount of trajectory data (TD). Several applications in different fields can utilize such trajectory data, for example, traffic control management, social behavior analysis, wildlife migrations and movements, ship trajectories, shoppers behavior in a mall, facial nerve trajectory, location-based services (LBS) and many others. Fortunately, there are now many trajectory data sets available that collected from moving objects such as cars with enabled GPS devices. Two main challenges arise when dealing with TD: 1) storing and analyzing TD data due to a large amount of data that arrives in a streaming and unpredictable rate. 2) inaccurate capturing of the exact location of moving objects due to the errors caused by GPS devices. In order to tackle these two problems and gain useful knowledge from TD, in this dissertation, we provide a framework called Trajectory Data Warehouse (TDW). This framework aims to review existing studies on storing, managing,

and analyzing TD using data warehouse technologies. Furthermore, we provide the requirements for building the TDW with different applications using the TDW.

Despite the second challenge, in this dissertation also, we utilize the vast amount of TD by building a digital road map. Because road maps are important in our personal lives and are widely used in many different applications; therefore, an up-to-date road map is essential. We propose a novel method to generate road maps using GPS trajectories that is accurate with good coverage area, has a minimum number of vertices and edges, and several details of the road network. Besides, our algorithm extracts road features such as turn restrictions, average speed, road length, road type, and the number of cars traveling in a specific portion of the road. To demonstrate the accuracy of our proposed algorithm, we conduct experiments using two real data sets and compare our results with two baseline methods. The comparisons indicate that our algorithm is able to achieve higher F-score in terms of accuracy and generates a detailed road map that is not overly complex.

Lastly, we present a data fusion framework for heterogeneous data Sources for Intelligent Transportation Systems (ITS). This framework aims to provide data fusion techniques to integrate and extract features from heterogeneous data sources to be ready for deep learning training approaches. We also generate preprocessed real-world traffic datasets that are publicly available to solve ITS-related problems. The traffic datasets have rich features such as traffic flow, average speed, vehicle occupancy, weather conditions, incidents information, congestion reports, point of interest locations, and temporal features. Furthermore, we provide two applications to show the importance of our data fusion techniques. (1) *Traffic datasets analysis and visualization*, where we build a data cube to provide in-depth analysis of the dataset. Also, a visual-interactive GIS tool that presents the results in different methods. (2) *Traffic flow forecasting using deep learning*, we performed a comprehensive study on

how different features can improve the traffic flow prediction models. The results show that deep learning approaches achieved better results when extra features are considered.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Figure Page

LIST OF TABLES

CHAPTER 1

INTRODUCTION

In this chapter, we start with definition and motivation of utilizing the availability of Trajectory data and other spatio-temporal data in section 1.1. Our research contributions come next in section 1.2. After that, in Section 1.3, we give an outline of the remaining chapters of the dissertation and how it is organized.

1.1 Definition and Motivation of Trajectory data

Advanced technologies in location acquisition allow us to track the movement of moving objects (people, planes, vehicles, animals, ships, ...) in geographical space. These technologies generate a vast amount of trajectory data (TD). Collections of objects movement data in term of position with respect to time is called Trajectory Data.

Fortunately, there are now many trajectory data sets available. These are collected from moving objects; e.g. cars with enabled GPS devices, people with smart devices, sensors on the road networks, and animals with GPS collars. Studying and analyzing trajectory data provide useful knowledge for several fields. For example, in traffic control management applications, the trajectory data collected from moving objects (cars, buses) moving along the road network can be used to build a road map, predict traffic flow, detect congestion areas, analyze the pattern of moving objects in different parts of a city, evaluate the traffic on working days versus the weekend, and locate the right space for placing advertising billboards on the road that has the highest number of vehicles. In ship movements applications, the trajectories of

ships can be used to determine the paths that are frequently used by different ships, locate the area that most of the ships use for fishing, and find the safest route to avoid pirates. Wildlife applications can use the trajectories of animals to study the animal's behavior, predict their future location, and how they migrate from one location to another. In air traffic management systems, the trajectories of airplanes can be used to choose the best route. Also, the trajectories collected from users with a smartphone can be used for social behavior analysis, obtain the most visited places by different users, and predict the users next location.

Hence, there are several challenges when dealing with trajectory data. First, trajectory data must be stored and analyzed effectively, which is considered challenges due to the large amount of data that arrive in a streaming and unpredictable rate [3, 5]. Second, inaccurate capturing of exact locations due to the errors caused by GPS devices. Thus, the purpose of this dissertation is to provide a method for effectively storing and analyzing trajectory data, which deals with the first challenge. Despite the second challenge, we utilize a huge amount of trajectory data, and we build a digital road map since many applications such as navigation systems, route planning, self-driving cars, and traffic control systems required the road map to be up-to-date. These road maps use only the trajectory data collected from moving objects. In addition, we provide a data fusion (DF) framework to integrate and extract features from heterogeneous sptio-temporal data sources to be used in Intelligent Transportation Systems (ITS) applications. This framework provides several DF techniques to deal with these data. We also present two applications to utilize the outputs of our DF framework: *traffic datasets analysis and visualization* and *Traffic flow forecasting using deep learning*.

1.2 Dissertation Contributions

The contributions of this dissertation can be divide into two parts. In the first part, we provide a survey of one method of storing and analyzing trajectory data that is Trajectory Data Warehouse (TDW). In this survey, we propose a framework that is influenced by the work in [3, 4, 5, 6], which aims to show the required steps to create a TDW to study the different characteristics of moving objects (specifically moving points as vehicles, people, animals, and ships). Additionally, the Moving Objects Database (MOD) is incorporated in the framework to provide more analysis results to the end users according to the application specifications.

In the second part, we utilize the availability of trajectory data and other spatio-temporal data. From these data, we build a digital road map using GPS trajectories collected from cars moving along the road network. We propose a novel method to generate accurate road maps with a good coverage area, a minimum number of vertices and edges, and several details of the road network. In addition, the algorithm extracts road features such as turn restrictions, average speed, road length, road type, and the number of cars traveling in a specific portion of the road. After that, we provide a machine learning framework for road network classification that aims to distinguish the roads based on their types, which either are main roads that expand over multiple counties or local roads that do not exceed the counties or cities boundaries.

Furthermore, we provide a data fusion framework with different techniques for ITS applications. This framework deals with heterogeneous data sources with different characteristics and generates preprocessed real-world traffic datasets with features. Then, we present two applications to show the importance of our DF techniques.

1.3 Dissertation Organization

In Chapter 2, we review existing studies on storing, managing, and analyzing TD using data warehouse technologies and we provide a framework to build Trajectory Data Warehouse. The goal of the framework is to show all the steps of building the TDW. The framework has five main components that are: 1) Data source, 2) Extract, Transform, and Load (ETL), 3) TDW, 4) Analysis Tools, and 5) Applications for TDW. These components are described in section 2.3. Additionally, we provide several applications that can benefit from the data that in the TDW as traffic control management, social behavior analysis, ships movements, TDW as a recommendation system, TDW for building and updating road map network, TDW for location-based services (LBS) applications, TDW for on-demand transportation services, and TDW for analyzing the moving objects on Toll Roads. Finally, we provide a discussion on several open issues and concerns that may become potential for future work in this field.

In Chapter 3, we describe our algorithm, called Road Map Generation (RMG), to generate digital road maps using GPS trajectories. The RMG consists of four main steps that are: 1) Analyzing the GPS data to provide more information such as direction, speed, time difference, road length, and so on. 2) Identifying the locations of turns and intersections by adjusting the Douglas-Peucker (DP) [8] algorithm. 3) Creating road segments between intersections. 4) Connecting road segments to the nearest valid turns and intersections. In addition, we demonstrate the accuracy of our algorithm using two real data sets and comparing it with two baseline methods. The comparisons indicate that our algorithm is able to achieve higher F-score in terms of accuracy and generates detailed road maps that are not overly complex.

In Chapter 4, we talk about our framework for classifying road maps using different machine learning algorithms. We applied Logistic Regression, K-Nearest

Neighbor, Naïve Bayes, and Support Vector Machine on road map data set to distinguish the roads based on their types, which either are main roads or local roads. This work can be used as pre-processing or post-processing to assist in constructing and updating the road map data set accurately.

In Chapter 5, we provide DF techniques for heterogeneous data sources for ITS applications. Our DF techniques aim to integrate and extract features from heterogeneous data sources with different characteristics (such as format and size) in a uniform representation to be ready for deep learning training approaches. In addition, we generate preprocessed real-world traffic datasets and we makes them publicly available for other researchers. These generated traffic datasets can be used for different ITS applications such as traffic analysis and visualization, traffic forecast prediction, incident prediction, travel time estimation. The preprocessed data can be used for benchmarking different prediction approaches that used deep learning. The traffic datasets provide rich features such as traffic flow, average speed, vehicle occupancy, weather conditions, incidents information, congestion reports, point of interest locations. It also provides temporal features such as a month of the year, day of the week, an hour of a day, weekend vs. weekdays, and other features for different road network locations. These features can allow researchers in various ITS fields to gain insight and apply their methods to solve ITS-related problems.

In Chapter 6, we present two applications that utilize the two outputs of our DF framework in chapter 5. The first application is Traffic datasets analysis and visualization. We build a data cube that provides an in-depth analysis of the traffic datasets. Furthermore, we design a visual-interactive Geographical Information Systems (GIS) tool to present the results in different ways such as maps, charts, and tables according to the user requirements. The second application is Traffic flow forecasting using deep learning approaches that are Deep Long-Short Term Memory

networks (LSTM), Deep Gated Recurrent Unit (GRU), and Bidirectional(BDLSTM). In this application, we check if incorporating traffic features with external factors will enhance the traffic flow forecasting result or not. Our experiments focus on short-term traffic flow (e.g., next hour) with multiple features. The experimental results show that better performance was achieved when extra features were included. Also, the overall result indicates our data fusion techniques' advantages.

Finally, Chapter 7 summarizes the contributions made in the dissertation.

1.4 Published Papers

As a result of my research, some articles were published during my Ph.D. study. The following are the published papers:

- **T. Alsahfi**, M. Almotairi, and R. Elmasri, "A survey on trajectory data warehouse," Spatial Information Research, vol. 28, no. 1, pp. 53–66, Feb 2020. Available: https://doi.org/10.1007/s41324-019-00269-x
- **Tariq Alsahfi**, Mousa Almotairi, Ramez Elmasri, and Bader Alshemaimri. 2019. Road Map Generation and Feature Extraction from GPS Trajectories Data. In Proceedings of the 12th ACM SIGSPATIAL International Workshop on Computational Transportation Science (IWCTS'19). ACM, New York, NY, USA, Article 2, 1–10.
- Mousa Almotairi, **Tariq Alsahfi**, and Ramez Elmasri. 2018. Using Local and Global Divergence Measures to Identify Road Similarity in Different Road Network Datasets.In Proceedings of the 11th ACM SIGSPATIAL International Workshop on Computational Transportation Science (IWCTS'18). ACM, New York, NY, USA, 21–28.
- Mousa Almotairi, **Tariq Alsahfi**, and Ramez Elmasri. 2019. Challenges of comparing and matching roads from different spatial datasets. In Proceedings

of the 12th ACM International Conference on PErvasive Technologies Related to Assistive Environments (PETRA '19). ACM, New York, NY, USA, 164–171.

CHAPTER 2

A Survey on Trajectory Data Warehouse (Drafted from [1])

2.1 Introduction

The movement of moving objects (people, planes, cars, animals, ships, hurricanes..) in geographical space can be captured nowadays using technologies such as Global Positioning Systems (GPS), smartphone sensors, Radio Frequency Identifications (RFID), and so on. These technologies generate a massive amount of spatio-temporal data (object id, time, position) for these moving objects. Collections of objects movement data in term of position with respect to time is called *trajectory data (TD)* and such data can be used in different domain applications as in traffic control management applications, ship movements applications, wildlife applications, social behavior analysis, and others.

In order to benefit from the collected trajectory data and gain knowledge, it must be stored and analyzed in an effective way, which is considered as challenges due to the large amount of data that arrive in a streaming and unpredictable rate [3, 5]. There are several existing methods proposed to deal with the trajectory data. One way is using moving objects databases (MOD) and another one is using data warehouses (DW) technologies. In this chapter, we propose a framework that aims to provide the requirements for building a Trajectory Data Warehouse (TDW) that is able to receive data from different sources and transform these data into valuable information. Our framework is based on the frameworks proposed in [3, 4, 5, 6] with additional methods, dimensions and measures to provide in-depth analysis that have

not been considered in the previous studies. However, there are many challenges that need to be considered when designing the TDW.

These challenges according to [5] are: the preprocessing phase that deals with a stream of observations (object id, time, location) and reconstructs trajectories out of these observations, storing the trajectories in MOD and apply an efficient extracting, transforming, and loading (ETL) process to pre-compute measures and load the TDW with appropriate values, providing an interface to allow multidimensional model analysis for the Online Analytics Processing (OLAP) operations.

To cope with these challenges, the proposed framework is influenced by the work in [3, 4, 5, 6], which aims to show the required steps to create a TDW to study the different characteristics of moving objects and provide a comprehensive overview of each step. In addition, we combine MOD and TDW in one framework so the same dataset can provide more analysis results to the end users according to the application specification. This chapter is focusing only on moving points (vehicles, people, animals, ships). The proposed Trajectory Data Warehouse framework consists of five layers, which will be discussed in more detail in section 3:

1- Data Sources: TDW data sources can be classified into two classes: one source is the data arriving from moving objects (spatio-temporal data), another source is the non-spatio-temporal data that can be used to add more semantic information to the first data source.

2- ETL:The ETL process stands for Extract, Transform, and Load. ETL is one of the critical components in the TDW. It performs extracting of heterogeneous data from different sources, transforming these data into the desired format, and loading these data to the TWD.

3- TDW: The TDW contains the data in a multidimensional model to allow the analysis tools to navigate and extract knowledge from the stored data.

4- Analysis Tools: The OLAP, Data mining (DM), and visualization techniques used to explore the TDW.

5- Applications for TDW: This layer can utilize the trajectory data stored in TDW and use it in different applications domain.

Furthermore, we provide discussion about different applications using the TDW such as traffic control management, social behavior analysis, ships movements, recommendation system and how these applications exploit the TDW. Also, we introduce new applications that can utilize the TDW such as building and updating road map networks, location-based services (LBS) applications, on-demand transportation services, and analyzing the moving objects on toll roads. We address some issues with existing TDWs and discuss future work in this field.

The rest of this chapter organized as follows: Section 2.2 provides an overview of the terminologies related to trajectory data and different types of data warehouses. Section 2.3 introduces the proposed framework with a survey of previous works. Section 2.4 provides a comparative study for the TDW and discusses several open issues. Section 2.5 includes the conclusions of this chapter.

2.2 Definitions and terminologies

2.2.1 Trajectory Data (TD)

In previous studies, the definition of trajectory data varies from one application to another. We now list some of the definitions of trajectory data from the literature: Definition 1:[9] (generic definition) "trajectory is the record of the movement of some objects i.e. the record of the positions of the object at a specific moment in time." Definition 2:[10] (generic definition)"trajectory is the user defined record of the evolution of the position (perceived as a point) of an object that is moving in space during

a given time interval in order to achieve a given goal."

Definition 3:[11] (persons, animals, vehicles movement applications) "A trajectory of a moving object is a discrete trace that the moving object travels in geographical space."

Definition 4:[12] (generic definition) "Trajectories *are* the segments of the object's movement track that are of interest for a given application."

The definitions above of trajectory data are also called *raw trajectory* since it contains only the object id, time and position information. From these definitions, we can see that the trajectory of moving objects can be observed as a sequence of tuples (object id, time, position) where *object id* refers to the moving object identifier, *time* refers to the time when a particular moving object is at a specific *position*.

Definition 5:[13] (semantic trajectory) "a trajectory that has been enhanced with annotations and/or one or several complementary segmentations." This definition of trajectory is called *semantic trajectory* since it has additional information than a raw trajectory.

2.2.2 Moving Object Database (MOD)

Moving Object Database (MOD) is an extension of database technology to support moving objects. Research in the field of MOD started in the late 1990s and until today the research in this field is ongoing. It receives significant attention from the database community. MOD deals with the moving objects that change their location over time, which is also called spatio-temporal database [14]. Research on MODs has two different approaches; one approach aims to answer queries on the current positions and the future location of the moving objects, the second one aims to show complete histories of the moving objects, which is called trajectory database in [15]. Raw trajectory data can be stored in MOD; however, it will cost

more in terms of processing and fetching the required data. Even though MOD uses optimized queries and indexing techniques, still these queries are expensive because of using many JOIN operators [3]. Due to this drawback in MOD, the alternative method that can deal with such challenges of handling trajectory data is using Data Warehouse technologies.

2.2.3 Data Warehouse (DW)

The author of [16] considers a data warehouse as "a subject-oriented, integrated, non-volatile, and time-variant collection of data in support of management's decisions." The data come from heterogonous independent sources and DW provides tools to transform raw data, clean, and integrate them into one common storage repository for further analysis [17]. Data warehouse uses the concept of a multidimensional model. This multidimensional model is used to organize data as a set of dimensions with different levels of hierarchies and fact tables [18]. A fact table includes foreign keys references to each dimension table and quantity information called measures. Dimensions tables provide details of the measurements in the fact table. For instance, Figure1 shows an example of the multidimensional model for a university data warehouse that stores information about students. It has a fact table, which contains the keys to each dimension and measures such as the students' major, GPA, honor status, and graduated status, and four dimensions tables, which store detailed information about students, departments, graduation, and time (semester). In addition, the multidimensional model can be represented in a data cube and each cell includes statistical data [19]. Different analytical tools are available to extract knowledge from the DW such as the Online Analytical Processing (OLAP) and Data Mining techniques (DM), which will be discussed in more detail in section 3.4.

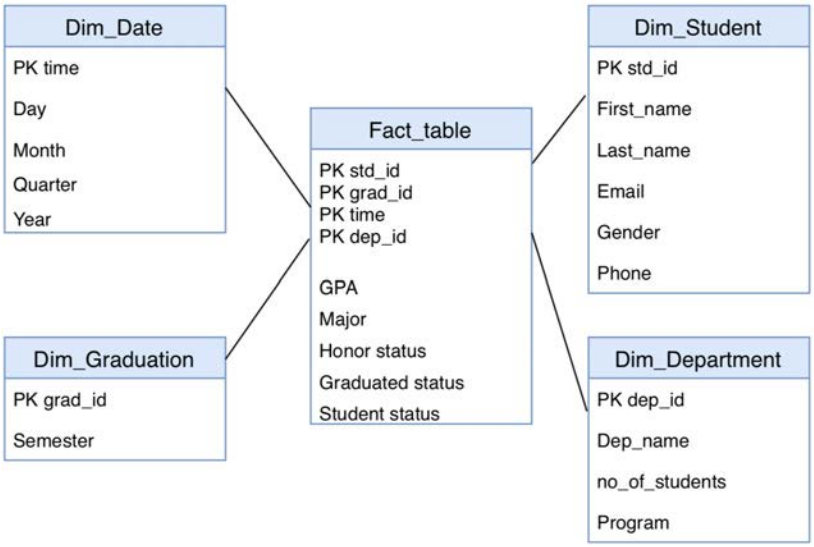


Figure 2.1: Multidimensional model

2.2.4 Spatial Data warehouse (SDW)

The authors of [20] introduce the concept of Spatial Data Warehouse (SDW). They consider the SDW as "subject-oriented, integrated, time-variant, and non-volatile collection of both spatial and non-spatial data in support of management's decision-making process." The data cube in SDW includes the spatial and non-spatial dimensions and spatial measures. SDW can manage spatial data and deal with its geometries, which are considered as a major difference from the traditional data warehouse. In addition, authors of [20] provide Spatial Online Analytical Processing (Spatial OLAP) operations to exploit the SDW. Another study [21] proposes different Spatial OLAP operations. However, the SDW does not consider moving objects in term of time, which leads to the Trajectory Data Warehouse [22].

2.2.5 Trajectory Data Warehouse (TDW)

Trajectory Data Warehouse (TDW) can be used to organize and analyze trajectory data collected from moving objects and can be exploited using the OLAP and

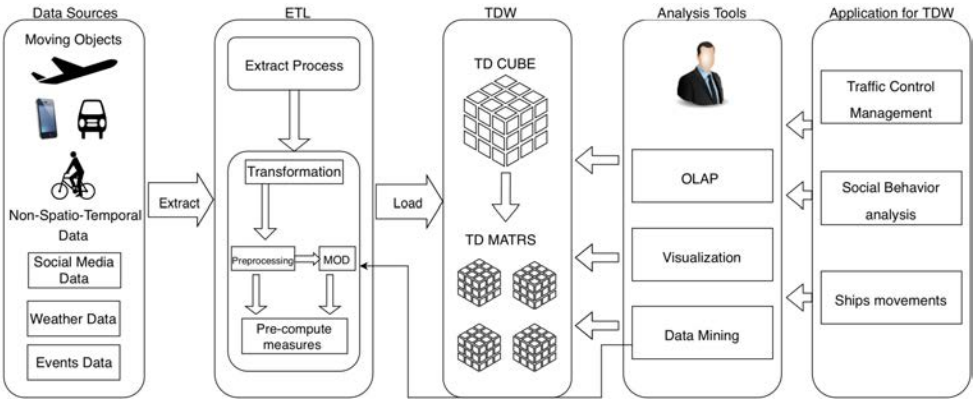


Figure 2.2: TDW framework influenced by the work in [3, 4, 5, 6]

data mining techniques [19]. The authors of [22] introduce the notion of TDW by using traditional DW to store aggregation information about the trajectories of moving objects and using the OLAP operations over the stored trajectory data. TDW is able to analyze measures of interest such as the number of moving objects in a specific area, average speed of vehicles, and acceleration of cars and other measures [6]. The goal of the TDW is to convert the raw trajectory data into valuable knowledge, which can be used in different application domains such as traffic manager applications, social behavior analysis, and recommendation system [4, 5, 23].

2.3 A Framework of Trajectory Data Warehouses

This section will introduce the framework of TDW that summarizes major steps required to build the TDW. Figure2 shows the overall structure of TDW. We focus on the most important layers. The framework starts by receiving data from different data sources (moving objects trajectories, non-spatio-temporal data sources), then reconstructs the trajectories and integrates them with other data sources. The reconstructed trajectories either are stored in a spatio-temporal database (i.e, MOD) or by using specialized software to perform pre-compute of measures before loading

them to the TDW. After loading the data to the TDW, OLAP, DM, and visualization techniques can be used to query and extract knowledge from the TDW. The following sections discuss each layer in more details.

2.3.1 Data sources

The primary data source for the TDW is the data collected from different moving objects (i.e. cars with enabled GPS devices, people with smartphones, and animals with tracking devices). These devices will provide the framework with data about the locations of the moving objects with respect to time. This type of data arrives in streaming form and at different rates. Other data sources are non-spatio-temporal data, which can be used to enrich the trajectory data semantically, such as social media data, weather data, crimes data, and events data. The study in [24] uses besides the data from moving objects other datasets like activities performed, and type of transportation to study the behavior of moving objects. Another study [25] uses seperately created trip log data with the data collected from moving objects to analyze the pattern of student's activites during trips. In paper [26], the authors use the city information that has the location of points of interest (POI) such as restaurant, company, park, and gas station in order to add semantic information to the trajectory data. Integration of non-spatio-temporal data sources along with trajectory data provide more meaning to the movement of the moving objects and enhances the decision taking by the decisions makers.

2.3.2 ETL

This process is considered as a backbone of the TDW framework since it converts the raw trajectory data into meaningful information and it also calculates the overall view of accumulated data sources. It begins with extract heterogeneous data from

multiple data sources, transforms the data into the desired format by applying a sequence of processes (cleaning, reconstructing, calculating, and integrating), and then load these data to the TDW.

2.3.2.1 Extraction of trajectory data

Data from moving objects arrive in a streaming and unpredictable manner. These data can be seen as a set of tuples (id,x,y,t) where ***id*** represents the moving object identifier, ***x,y*** represent the coordinate of moving object location and ***t*** represents the time where the moving object is located. The extraction process aims to extract the sets of observations to be ready for the transformation process. The data from non-spatio-temporal data sources as well can be extracted and sent to the next process.

2.3.2.2 Transformation process

This process will apply on the received observations of the moving objects from the previous step and integrate the trajectory data with the other data sources. This process depends on the application specification. Generally, there are two main functions in this process: Preprocessing in which trajectories are created from the aggregated received observations that are close to each other based on determined conditions, and Pre-compute of measures that divide the space into cells (grids) and compute the measures for each cell based on trajectories inside each cell.

Preprocessing phase: The goal of this phase is to convert the received observations (id,x,y,t) from moving objects into meaningful and enriched trajectory data. In order to reconstruct the trajectory from its received observations, one way is using the local linear interpolation method, which assumes the object is moving between two observations following specific rules [27, 3, 28, 22, 4, 5]. Figure3 shows an example of

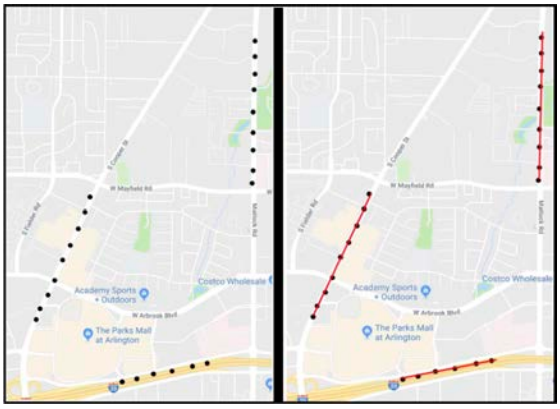


Figure 2.3: Raw trajectory - Reconstruct trajectory

a

```
Point_id, Latitude, Longitude, Time
624029,41.879176,-87.649627,1301806255.0
624042,41.879172,-87.649875,1301806259.0
624058,41.879171,-87.6502,1301806262.0
624077,41.879166,-87.650524,1301806265.0
624115,41.879159,-87.65077,1301806270.0
624150,41.879153,-87.651065,1301806274.0
624179,41.879151,-87.651426,1301806277.0
624199,41.879146,-87.651695,1301806279.0
624220,41.879138,-87.651968,1301806281.0
624242,41.879132,-87.652259,1301806283.0
624264,41.879126,-87.652578,1301806285.0
```

b

point_id	lat	long	distance_km	distance_m	bearing	direction	datetime	timedelta	velocity	acceleration	heading_main
624029	41.879176	-87.649627	0.020590606	20.53689927	-91.2409026	268.7590974	4/2/11 23:50	0 days 00:00:0	18.48320935	3.451355707	SW
624042	41.879172	-87.649875	0.026977605	26.90719348	-90.236669	269.763331	4/2/11 23:50	0 days 00:00:0	32.28863218	-0.03090321	SW
624058	41.879171	-87.6502	0.026900104	26.82993544	-91.18727	268.81273	4/2/11 23:51	0 days 00:00:0	32.19592253	-5.84044467	SW
624077	41.879166	-87.650524	0.020434598	20.38137293	-92.1885755	267.8114245	4/2/11 23:51	0 days 00:00:0	14.67458851	1.462907233	SW
624115	41.879159	-87.65077	0.02449623	24.43236075	-91.5646604	268.4353396	4/2/11 23:51	0 days 00:00:0	21.98912467	3.469198	SW
624150	41.879153	-87.651065	0.029966472	29.88826389	-90.4262046	269.5737954	4/2/11 23:51	0 days 00:00:0	35.86591667	1.411274612	SW
624179	41.879151	-87.651426	0.02233588	22.27763362	-91.4299682	268.5700318	4/2/11 23:51	0 days 00:00:0	40.09974051	0.307552606	SW
624199	41.879146	-87.651695	0.022678421	22.61935873	-92.2537812	267.7462188	4/2/11 23:51	0 days 00:00:0	40.71484572	1.333775164	SW
624220	41.879138	-87.651968	0.024164333	24.10133114	-91.586159	268.413841	4/2/11 23:51	0 days 00:00:0	43.38239605	2.085598622	SW
624242	41.879132	-87.652259	0.026487735	26.41866294	-91.446979	268.553021	4/2/11 23:51	0 days 00:00:0	47.55359329	2.168895401	SW
624264	41.879126	-87.652578	0.028903858	28.82854672	-91.9892496	268.0107504	4/2/11 23:51	0 days 00:00:0	51.89138409	1.028269367	SW
624288	41.879117	-87.652926	0.030049493	29.97106824	-89.5747319	270.4252681	4/2/11 23:51	0 days 00:00:0	53.94792283	-0.00055715	W
624312	41.879119	-87.653288	0.030048877	29.97044918	-90.2124557	269.7875443	4/2/11 23:51	0 days 00:00:0	53.94680853	-1.40885307	SW
624336	41.879118	-87.65365	0.028479331	28.40505689	-91.3457556	268.6542444	4/2/11 23:51	0 days 00:00:0	51.12910239	-3.72690016	SW
624362	41.879112	-87.653993	0.024327506	24.26405671	-91.3128644	268.6871356	4/2/11 23:51	0 days 00:00:0	43.67530208	-9.81347322	SW
624388	41.879107	-87.654286	0.020092703	20.04029638	-91.2716615	268.7283385	4/2/11 23:51	0 days 00:00:0	24.04835565	-2.76548924	SW
624428	41.879103	-87.654528	0.026321785	26.25314656	-91.456105	268.543895	4/2/11 23:51	0 days 00:00:0	15.75188794	2.971517146	SW
624512	41.879097	-87.654845	0.028057384	27.98415901	-90.455221	269.544779	4/2/11 23:51	0 days 00:00:0	33.58099081	2.370088347	SW
624557	41.879095	-87.655183	0.022665379	22.60625325	-91.1272803	268.8727197	4/2/11 23:51	0 days 00:00:0	40.69125585	2.084399354	SW
624587	41.879091	-87.655456	0.024987455	24.92225253	-90.7668274	269.2331726	4/2/11 23:51	0 days 00:00:0	44.86005456	1.712590075	SW
624617	41.879088	-87.655757	0.026895322	26.82513039	-90.4748996	269.5251004	4/2/11 23:51	0 days 00:00:0	48.28523471	-0.52156277	SW
624649	41.879086	-87.656081	0.026314291	26.2456162	-90.4853906	269.5146094	4/2/11 23:51	0 days 00:00:0	47.24210916	-1.56354938	SW
624680	41.879084	-87.656398	0.024572458	24.50833911	-90.7797823	269.2202177	4/2/11 23:51	0 days 00:00:0	44.11501039	-6.53095035	SW

Figure 2.4: This is an example showing the trajectory data: **a** original data received from GPS and **b** after applying different methods of preprocessing .The dataset obtained from [7]

received observations (left - as points) and the results after applying the preprocessing (right - as trajectories). Moreover, the authors of [4] define general parameters that can be used to decide if the new arrival observations belong to previously stored trajectory or to create a new trajectory. These parameters could be the temporal gap between two observations, the spatial gap between two observations, maximum speed, maximum noise duration, or tolerance distance. In the papers [3, 4, 5], this process is named as "the reconstructing process", which only required for the spatio-temporal data that is received from moving objects.

Another way of preprocessing in addition to the existing methods is to receive the whole trajectory for each moving object from start to end of each trip, which can be defined according to the application specification. For example, if the TDW is for analyzing trajectories of a Taxi, each trip will be considered as one trajectory. Also, when the TDW is used to study the behavior of people, each day can be considered as one trajectory in order to extract all the movement patterns. This helps to distinguish between the locations where the moving object is moving or staying.

Furthermore, the raw trajectory data received from each moving object can be semantically enhanced to extract more information using different methods as in [29, 30, 31], which can be adopted by the TDW framwork. The study in [29] provides a model to add more information to the raw trajectory data. Their method first finds the locations where the moving object is stoping (staying points) and use another dataset to annotate the stop point with the location information (e.g. home, work, school). The authors of [26] transform the trajectory data to semantic trajectories by using other data sources, for example the POI information of a city. Figure4 shows an example of raw trajectory data from a vehicle moving along the road and the results after different methods of preprocessing that add details to enrich each trajectory. After the preprocessing phase is finished, the result will be sets of trajectories

Figure 2.5: Roadmap divide to cells to compute measures for each cell

with additional information such as duration time at each location, heading/bearing, speed, and other additional information that will be used in the following step. The reconstructed and enriched trajectories combine with non-spatio-temporal data can be either stored in a spatio-temporal database (MOD) as in [3, 4, 5] or directly used in specialized software as in [27, 22, 28].

Pre-compute of measures: Measures are used to describe some of the properties for trajectory data of moving objects. As it is mentioned before, the fact table contains sets of aggregated measures. Most of the TDW and TD studies, such as in [27, 3, 28, 22, 4, 25], divide the space into cells or zone (Figure5 for example) and include measures that can be answered from the stored information in each base cell. The values of measures are stored in an aggregation without storing the identifier of individual moving objects due to the users' privacy, and individual user data may require large space [22]. Aggregation means that computing the measures such as average speed for all moving objects in a specific cell during a specific time and storing only the values for the measures without the moving objects details. Different applications have different sets of measures. In table 2.1 and 2.2, we provide some of the measures that can be answered from the TDW according to the applications

Table 2.1: Measures for TDW

Measure describe in	Measures	aggregation function	Moving Object
[28, 22]	No. of observation in the cell	D	Car
[28, 22]	No. of T's starting in the cell	D	Car, Ships
[27, 28, 22, 3]	No. of T's in the cell – presence	H	Car
[28, 22, 3]	Total distance covered by T's in cell	D	Car, Ships
[28]	Average speed of T's in the cell	A	Car, Ships
[28]	Maximum speed of T's in the cell	D	Car, Ships
[22, 3]	No. of T's entering,leaving, crossing cell	D	Car, Ships
[22]	No. of T's that covered a distance larger than a given value d in the cell	H	Car, Ships
[22]	No of T's that covered a distance larger than the average distance covered, by all the T's intersecting the cell	H	Car, Ships
[4]	No. of distinct users	H	Car, Ships
[22]	No. of T's that intersect,another T only in the cell	D	Car, Ships
[4]	Average traveled distance	A	Car
[32]	No. of Stops by trajectory	D	Car
[33]	NbPath:No. of paths constituting T	D	Herd
[33]	MaxSpeed:find maximum speed of the herd	D	Herd
[33]	MinSpeed: it corresponds to minimum speed of the herd	D	Herd
[33]	CoveredDistance:corresponds to the covered distance traveled by the animals during their displacement	D	Herd

Table 2.2: Measures for TDW

Measure describe in	Measures	aggregation function	Moving Object
[33]	During: duration of the trajectory	D	Herd
[34]	Frequent pattern of trajectories: To discover different pattern of MO	N/A	Car
[25]	Hot Zone: Return the location that visitors spend more than 3 min at specific location	N/A	People
[35]	Billboard location: This measure will return the best location to place ads	N/A	N/A
new measure	Turn location: This measure will return the point where moving objects changed location	N/A	Any MO
new measure	No. Visitor: Return the number of visitors for specific POI	D	People
new measure	Staying duration: This measure return the total amount of time user spent at specific POI.	D	Any MO
new measure	Road status: to return the road status (closed, open) for specific time and location.	N/A	N/A
new measure	Transportation mode: to find type type of transportation that used by the moving objects	N/A	People

The table describes the different measures stored in a fact table and can be answered from the TDW according to the applications type. T: refers to Trajectory. No. refers to number. D:Distributive, H:Holistic, A:Algebraic

type. Next, we will explain how the measures are computed before loading them to the fact table of the TDW.

1. *Pre-compute of measures using specialized software:* The studies in [27, 28, 22] use specialized software to deal with the reconstructed trajectories. Their goal is to use less buffer memory to store received observations and perform pre-compute of measures. Measures are categorized according to the amount of pre-computing requires before loading them into the TDW. For instance, measure *Number of trajectories starting in the cell* does not require pre-computation and can be uploaded into the TDW using single observation. Other measures such as *Number of trajectories that intersect another trajectory only in the cell* requires a pre-computation of all the observations for these trajectories.

2. *Pre-compute of measures using spatio-temporal database:* The authors in [4, 5] use an engine called HERMES MOD to compute measures such as average distance travel, average traveled duration, and average speed for each base cell of the TDW. The motivation behind using MOD is to utilize the set of operations available in the MOD to handle trajectory data. Before performing compute of measures, they proposed two methods to extract part of trajectories that fit in each base cell: *cell-oriented approach* (COA) and *trajectory-oriented approach* (TOA). The COA searches for the parts of trajectories that reside in each base cell, while the TOA goal is to exploit the spatio-temporal cells that trajectories lie in. Another study [3] follows the same approach for computing the measures.

   In addition to previous methods, we propose other methods to compute these measures, which can be adopted by the TDW to support new applications. Since the trajectory data has been enhanced in the preprocessing phase with other

semantic information, we can use this information to compute new measures such as *turn location* and *points of interest (POI)*. To compute the measure *turn location*, which returns the location where the moving objects change direction, we can check each trajectory and return the points where the direction has changed [36]. After that, for all points that have been checked, a clustering algorithm as in [37], DBSCAN [38], or K-means can be applied to find the location of each specific turn.

For measures such as *points of interest (POI)* or similar that try to identify staying points, we can check each moving object trajectory and find the points that the moving object stays without changing location for some time. Then by using any of the clustering algorithms as in [39, 40, 41, 42], we can find the areas where most of the moving objects stay and mark these areas as the points of interest for further analysis. The study in [25] computes the measure *hot zone* by finding the locations where users stayed more than 3 minutes and within spatial distance 50 meters. Another study [35] uses the TD of moving objects to determine the best location for placing billboard advertising.

Aggregation over measures for OLAP operations: Since the OLAP is one of the

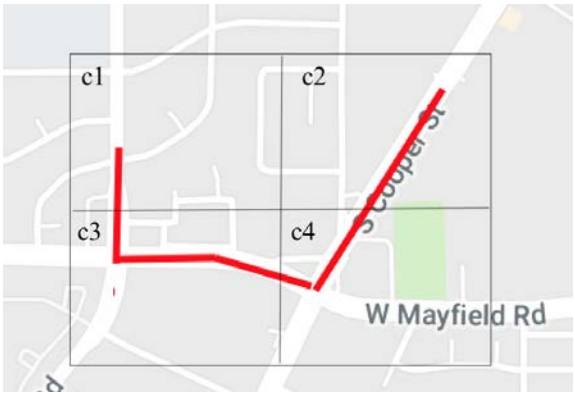


Figure 2.6: Trajectory will count 4 times during the roll-up

essential tools to extract knowledge from the TDW, the TDW must support different types of OLAP operations. For instance, to compute the roll-up operation, which requires information at the lowest level to determine the values in a higher level, aggregate functions are needed to be applied [3]. Table 1 shows the type of aggregation function needed to compute each measure. According to [43], the aggregate functions are classified into three categorize based on the space complexity to compute the higher values from the already stored pre-computed values in each base cell:

Distributive: the higher values are calculated using pre-computed values by applying a simple function such as SUM.

Algebraic: the higher values can be calculated using pre-computed values with other sets of measures.

Holistic: the higher values cannot be calculated using the pre-computed values in the base cell.

Holistic measures are difficult to compute in an exact way since no set of tuples have the pre-computed value that could be used to produce higher values [22, 4]. An example of a measure of trajectory data that needs a holistic aggregation function is the *presence* measure. It returns the number of distinct trajectories occurring in a given spatio-temporal area. A double count problem can occur when performing the OLAP roll-up operation since only the aggregation information is stored in the TDW without identification of each trajectory. This problem is defined according to [44]: "if an object remains in the query region for several timestamps during the query interval, it will be counted multiple times in the result." Figure6 shows one trajectory that spans multiple cells during a specific time and each cell store one part of the trajectory, during the roll-up if we sum up the value in each base cell we will end up with four distinct trajectories while it is only one trajectory. To solve this issue, authors of [27, 5] define two alternative aggregation functions to

compute the presence measure in an approximate way by defining distributive and algebraic aggregation functions. They develop an algorithm for each method. From their experiments, the algebraic aggregate function shows more accurate results with fewer errors, while the distributive function shows a large number of errors when the granularity increased. The study in [4] follows the same approach to solve the double count problem for measures such as *count users*. The study in [3] introduces a measure called *visited* which can be used to estimate the presence measure. On the contrary of these mentioned studied, the authors of [32] use another approach to tackle this problem by developing an algorithm to count the number of distinct trajectories. This is because the object identification is stored in their TDW, so the algorithm checks the regions in the lowest level that the object is in and returns the number of trajectory count. Their experiment shows a more accurate result than the presence measure.

2.3.2.3 Loading data to the TDW

Loading process aims to load the TDW dimensions and fact table with suitable aggregation information. The measures that are stored in the fact table can be loaded with the pre-computed values in the previous process. Studies in [4, 5] load the pre-computed values to the fact table after using the MOD. On the other hand, studies in [27, 28, 22] load the aggregation information after pre-computing the measures without using any spatio-temporal database. From our perspective, using the MOD provides some advantages over the specialized software. Since the TDW stores aggregation information, the MOD can store each object trajectories to be used later in case that some requirements are changed as a new measure is identified, or to provide data mining capabilities over the trajectory data as in the study of [6].

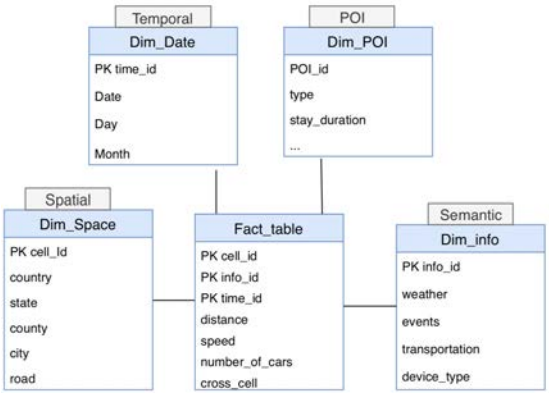


Figure 2.7: TDW model using star schema

2.3.3 Trajectory Data Warehouse

The TDW contains the Trajectory Date cube and collection of trajectory data marts that are generated form the TD cube. The TD cube consists of spatial (i.e. country, state, road, river,...), temporal (i.e.. minute, hour, day, month, year..) dimensions to describe where the objects are moving during a specific time, and non-spatio-temporal dimensions (i.e. type of transportations, events, type of smartphone, gender, age,...) in order to enrich the trajectory data, and a fact table contains measures of interest.

In addition, we add a new dimension called Point of Interest that includes information about the important locations in a specific region that have been visited by the moving objects. These POIs could be gas station, school, mall or any other important point according to the user requirements. With this dimension, we will be able to provide more analysis on the TD such as find the time spent by shoppers inside a mall, a restaurant that has most visited customers, the location where wildlife animals stay a long time during a day, and so on. POIs can be categorized as main

type (Food, Services, Health) and subtype (restaurant, gas station, hospital). Figure7 shows an example of the TDW model using star schema.

2.3.3.1 Trajectory Data Warehouse models

Most of the previous studies on the TDW consider the logical level with no interest in the conceptual level, by adapting the well-known models for designing the traditional data warehouse and adding the spatial and temporal dimensions. The study in [4, 5] use the *multidimensional model* for designing the TDW. The trajectory data cube consists of one fact table includes beside the keys to each dimension measures over the aggregation information and is three dimensions: Spatial dimension to describe the geography where the objects are moving; temporal dimension to define the time; and non-spatial dimension (semantic) which refers to the user information. The dimensions tables have different sets of attributes that can be used to define different levels of hierarchies to answer OLAP operations such as the roll-up,drill-down operations.

Similarly to [4, 5], the study in [3] uses the *dimensional fact model formalism*, which consists of spatial dimension that can be defined according to the environment of the moving objects, temporal dimension, and non-spatial dimension. These dimensions provide different levels of hierarchies. Besides, the TDW has two fact classes one has measures for a specific region and another for two adjacent regions. The studies in the [27, 28, 22] model the TDW using the *star schema model*, which consists of spatial, temporal dimensions with different level of hierarchies, and one fact table. Furthermore, the studies in [32, 33] use the same model with additional non-spatial dimensions to provide more analysis details. The study in [13] models a semantic TDW and aims to provide more knowledge of the moving object's behaviors. The

TDW contains six dimensions (Spatial, Time, Trajectory, Transportations, Pattern, and Trajectory) with different level of hierarchies and a fact table.

2.3.3.2 Trajectory Data Marts

In traditional DW, data marts usually contain one subject area such as data for a single department "Sale department, Employees Department." It is a small version of the whole DW. In the same manner, the Trajectory Data Mart (TDM) can be used to store information for specific use. It builds from the Trajectory Data Cube. For instance, one data mart can store the information about trajectories of moving objects in a specific city; another can be used to store the information for any specific month. However, in previous studies, there is no consideration of using the TDM.

2.3.4 Analysis Tools

To exploit the TDW and reveal knowledge to the end users, different tools are available for this purpose such as the OLAP, DM, and visualization techniques. The OLAP provides different types of operations to exploit the multidimensional data in an efficient way. [18] OLAP operations such as *roll-up*, which is used for data aggregations along dimensions (i.e. aggregate data from lower dimensional to higher dimensional, for example, from day to a week to month..). The *drill-down* operation is used to decrease the level of aggregation (i.e. from higher dimensional to lower dimensional, for example, from state-county-city..). The *Slice* and *Dice* operations are used to select part of the data cube. *Pivot* operation is used to change the view of the data cube. In this chapter, we will discuss the three techniques that have been widely used in the TDW, which are the visualization of TD and OLAP operations, data mining techniques.

2.3.4.1 Visualization of TD and OLAP operations for TDW

Traditional OLAP representation of the result such as tables, graphs, and text is inadequate for decision making for the TDW. If these representations are used for the TDW; the interpretation of the results is not an easy task [3]. To provide more in-depth analysis to the end users, the OLAP tools can be integrated with the Geographical Information Systems (GISs) for utilizing graphical display. Authors in [3, 5] develop an interface to visualize the OLAP operations on a map. The developed tools allow the user to investigate the data store in the TDW and visual the results on a map.

In addition, the trajectory data can be explored by using visualization techniques. This method is either applied to the cleaned trajectory data stored in the MOD within the TDW without requiring to precompute measures or it can be applied to several dimensions of the TDW. The authors of [25] study the pattern of moving objects (students on a field trip) by visualizing the log data using the space-time cube method.

2.3.4.2 Data Mining

DM techniques are an essential tool in TDW that allow to find hidden information and discover useful knowledge. The studies in [6, 34] use a pattern mining technique for the TDW, which aims to analyze mobility pattern of a moving object or multiple moving objects together. The authors of [6] focus on detecting the traffic relationships between different road segment in a road network. Their TDW framework provides DM over trajectory data store in the MOD. They model the road networks as directed graph G=(E, V) where *Edges*: represent the road segments, *Vertices*: indicate point of interset (POI) locations (school, gas stations,..). The authors use an

algorithm that defines the relationships between the network edges (traffic propagation, split, and merge) and provides a clustering algorithm to group each edge with similar values together.

The study in [34] introduces a new measure for the TDW called:*frequent pattern of trajectories*. This measure is obtained from the data mining process over the trajectory data and the patterns are exploited using the OLAP queries. The system works by receiving raw trajectory data from moving objects (t,x,y) and use the local linear interpolation to reconstruct the trajectories. Then the trajectory data transform to (t',r) where *t'* represents the time the moving object enters a new region *r*. These transformed data then loads to the TDW, to perform the OLAP operations. One application that can benefit from this measure is a tourist application, which can be used to find the most visited region by different visitors. The main difference between [6] and [34] is that in [34] a new measure is added to the TDW, while in [6] they rely on the MOD to apply the DM.

2.3.5 Applications for TDW

A variety of domains of applications can benefit from the TDW. Examples of applications using TDW are traffic control management, location-based services, animals' movements studies, ships movements, shoppers behavior in a mall, facial nerve trajectory, and others more. We will mention four applications with some queries that can be answered from the TDW. In addition to these applications that were considered in previous studies, we introduce new applications that can utilize the proposed TDW framework to be used for future applications.

2.3.5.1 Traffic Control management

One of the primary applications using the TDW is the road traffic application. The trajectories data collected from moving vehicles along the roads can be used to make decisions to improve the road services, control traffic, and discover patterns of moving vehicles. TDW that stores trajectories data of moving vehicles for road traffic are able to answer queries like: *"Which area has the most traffic in the morning?" "What is the number of vehicles leaving a specific area in the evening?" What is the road status at 3 pm in the highway I-20?" " Which region has the most visited vehicles during the weekdays?" "What is the average speed of vehicles inside the city?"*

The study in [3] discusses various properties of the TDW for analyzing the road traffic such as the number of visited vehicles at a different time for a specific region, numbers of cars leaving a particular area and entering another area, the difference in traffic between working and weekend days, and so on. Another study in [4] also utilize the TDW for traffic control and answers query like: find all the parking stops near specific areas (restaurant, mall, school,..) and retrieve the number of stops during a specific time within specific regions. In addition, the study in [5] shows how the traffic is diverse throughout the week and the movement of cars from one part of a city to another part.

2.3.5.2 Social Behavior Analysis

TDW can be employed to analyze the behavior of an individual or group (human, animals, ..) movement to predict their future locations, find patterns in the activities performed by a specific group, find the locations that have been visited by many moving objects, and so on. For instance, TDW for shoppers in a mall can answer queries like: *" what is the most visited store during the weekend?" " Which*

*store has the least shoppers between 12 pm and 3 pm?." " how long have the shoppers stayed at a specific store?"*The study in [25] analyzes the TD combined with log data and find the hot zones that users stayed for a specific amount of time. In addition, the TDW can be used for tracking the migration and the movement of animals. The study in [33] uses the TDW to store the trajectories of animals at a different time in different regions, and allows analysts to explore different behaviors of animals and predict their next location.

2.3.5.3 Ships Movements

In addition to the previous examples of TDW, it also can be used to analyze the movement of objects in a different spatial domain. The study in [3] shows how the TDW is used to analyze the movement of ships in a sea and provide some quires that can be answered from the TDW: *" Which is the most fishing areas during a specific week?" "Where is the area with the highest amount of fishing during the day?" "What is the number of ships in a specific region during the first week of each month?"*

2.3.5.4 TDW as a recommendation system

The study in [13] uses the TDW as a recommendation system to create a plan for visitors of a specific location. The user provides the time and budget to the TDW. Then the TDW returns a list of recommended places to be visited and the duration that has been extracted for the trajectories of previous tourists. This application utilizes knowledge base information, which is non-spatio-temporal dimensions, to provide more accurate results based on customers favorites and requirements.

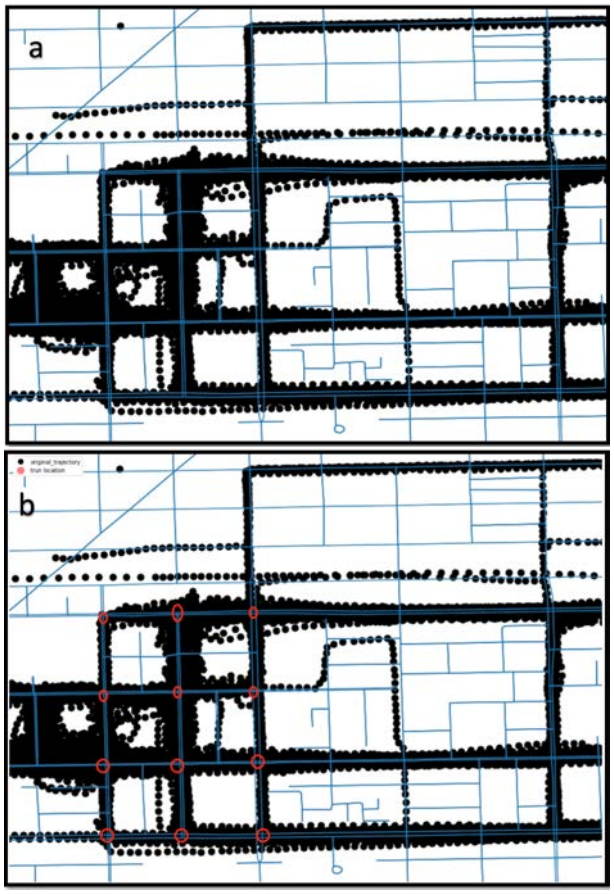


Figure 2.8: This is as example of using the TDW for bulidng and updating road map network: **a** shows the raw trajectories (Black dots) from moving objects - (blue line-represent the exisiting road map) and **b** shows the result (Red circle) of the turn location measure .The dataset obtained from [7]

2.3.5.5 New Applications for TDW

This section will discuss new applications that have not been considered in previous TDW studies and can be supported by the proposed framework.

1- TDW for building and updating road map network:

The measures stored in the fact table such as turn location and road status can be combined with other information that is stored in the MOD inside the TDW. They can be used to create a road map for the area that does not have a map before or keep an existing road map up-to-date. Figure8 shows the received trajectories from

vehicles moving along the road, then the result after querying the TDW for turn location. Furthermore, the measure road status can be used to find the roads that have been closed recently since no moving objects traveled by or roads that have been recently opened after construction is finished.

2- TDW for Location-based services (LBS) application:
The proposed TDW can be used as the data source or geodatabase component of any LBS application. The TDW dimensions and measures that store information such as the POI, the road status, and transportation mode can assist in locating the requested services. Examples of LBS can be find the nearest restaurant or mall, find the best route from my current location to a new location, and so on. The TDW with the other components of LBS can answer queries like: *"List the nearest ATMs from my current location", " Find the fastest route from my current location to the airport", and "how long does it take by bus to the nearest mall"*

If we take the following query: *"Provide a list of the five nearest restaurants that have been visited by at least a hundred customers last month"*, the TDW can assist easily in returning the number of visitors for a specific point since it is already stored in the fact table. The LBS application will utilize this information and give to the user a list of restaurants that satisfy the users' condition.

3- TDW for on-demand transportation services: Since TDW store the history information of the location and time that customers request service, then we can utilize the TDW to find the areas with many service request during specific day and time to predict the location of the next customers and assign several cars at each location.

4- TDW for analyzing the moving objects on Toll Roads:

The authorities for toll roads using the toll tag or plate number to record the number of moving objects (Cars, buses, trucks, and so on) passing each tollbooth, which is considered as POI. TDW framework can utilize this data and provide detailed information to the end users. Queries can be answered from the TDW for analyzing toll roads like: *"What is the average speed between two tollbooths?", "What is the number of vehicles passing specific tollbooth on Saturday between 1 pm and 3 pm?", " Which part of the toll road has the maximum speed?", " Which part of the toll road has the lowest speed?"*

2.4 Discussion

In this section, we provide a comparative study (see table 2.3) for the TDW that we have introduced and we discuss several open issues and concerns which may become potential for future work in this field. We take in consideration five factors in our comparative study, which are: TDW schema, List of Dimensions in TDW, Data sources (Real, or Synthetic data sets), Implementation tools and framework, and Analysis tools.

Even though there are many studies have been done on the TDW, there still are open issues to be considered as future research work in this field. Below is a list of these issues:

-Real-Time TDW: To the best of our knowledge, previous studies on the TDWs follow the traditional ETL process which depends on batch mode and there are no discussions on how to provide real-time analyses. However, these days, users require the data to be available and up-to-date in order to make their decisions spontaneously. For instance, in road traffic application, a police department (user) may want to predict the traffic (high or low) in a specific location for a given future time and,

Table 2.3: Comparison of TDW approaches

paper	TDW Schema	Dimensions	Data Sources	Tools	Analysis tool
[27]	Star Schema	Spatial, Temporal	Moving Objects (School buses) - R+S	Oracle	OLAP
[28]			Moving Objects-S	MS SQL Server	OLAP
[22]				N/A	OLAP
[32]		Spatial, Temporal, non-spatial	Moving Objects (Cars)-R	St-Toolkit	OLAP
[33]			Moving Objects (Herds)	N/A	N/A
[6]	Multidimensional model	Spatial, Temporal, non-spatia	Moving Objects (Cars)-R	N/A	DM
[4]				N/A	OLAP
[5]				N/A	Visual-OLAP
[3]	Dimensional Fact Model	Spatial, Temporal, non-spatial	Moving Objects (Cars, Ships), - R	Oracle and V-Analytics	Visual-OLAP

This table compares different frameworks on TDW.

based on this prediction, locate the proper number of police cars at that locations. Current TDW is not able to provide this information to make decisions on time; it becomes a necessity to introduce Real-Time Trajectory Data Warehouse that provides the capability of a real-time report.

- *Integration of non-spatio-temporal data*: There are some works on the integration of trajectory data with other data sources, still there is a potential for enhancement to integrate more non-spatio-temporal data such as social media data, health data, weather data, traffic accident data to support the decision makers and add more meaning to the trajectory data. Furthermore, methods for an efficient integration of trajectory data with non-spatio-temporal data need to be considered.

- *New measures*: Introduce new complex measures to the TDW that allow the users to gain more information about the moving objects and their behaviors. For instance, a new measure for finding the correlation between the average speed of cars and the weather condition (clear, rain, fog, cold), a new measure for predicting the traffic by combining the trajectory with events data in a specific location, a new measure to find the POI by integrating the social media and trajectory data. Some work on spatio-temporal prediction has been done [45, 46, 47], which can be used as a basis for these new measures.

- *New DM techniques*: Apply the well-studied techniques of DM to take as much as possible advantages of data provided by TDW. Some of the powerful techniques are association rules, clustering, and classification.

- *Map comstruction*: The trajectory data store in the MOD within the TDW can be used to generate a map for a road network, animals movement map, and ships path map.

2.5 Conclusion

Technologies such as Global Positioning Systems (GPS), smartphone sensors, and Radio Frequency Identifications (RFID), allow capturing the locations of moving objects in time, which leads to the creation of spatio-temporal datasets called trajectory data. Dealing with trajectory data raises challenges in many aspects such as storing, managing and analyzing these data. Many applications could utilize and benefit from the trajectory data such as traffic management, social behavior analysis, wildlife migrations and movements, ship trajectories and many others. The MOD and TDW are mainly used to handle the trajectory data.

In this chapter, we have reviewed the existing studies that have been conducted on TDW, and we proposed a framework that includes an extensive review of all the requirements needed to create a TDW. In a nutshell, we explained how the TDW framework starts working by receiving data from two different sources, which are: raw trajectory data acquired from moving objects, and non-spatio-temporal data. Then, the ETL process is used to reconstruct, transform, and integrate trajectory data with other data sources and pre-computing of different measures. Later, the pre-computed measures are loaded to the TDW. In addition, we discussed different analysis tools that have been used in order to exploit the TDW, which are the OLAP and DM technologies. We presented four types of applications that use the TDW. Finally, we provided a comparative study on different TDW approaches and we discussed some open issues related to TDW. We intend in our future work to build real-time TDW and add more complex measures.

CHAPTER 3

Road Map Generation and Feature Extraction from GPS Trajectories Data (Drafted from [2])

3.1 Introduction

Digital road maps are currently essential. A variety of applications depend on maps such as navigation systems, route planning, self-driving cars, and traffic control systems. All these applications require the road map to be up-to-date. Usually, the process of generating an up-to-date road map requires either specialized vehicles roaming the road network to generate the road data or using satellite imagery to extract the road maps. Using these methods is expensive in terms of cost and time since they require specific cars and skilled drivers [48] or maintaining the satellite.

Fortunately, there are now many tracking data sets available. These are collected from moving objects; e.g. cars with enabled GPS devices, and people with smart devices. Such data sets enables us to infer and generate up-to-date digital road maps automatically. These generated road maps from GPS trajectories can be used in areas that have no map or may update an existing road map by discovering new roads or detecting closed roads. In addition, the GPS trajectories allow us to extract road features (semantic attributes) such as speed information (average, limit), length of the road, road type (one-way, two-way, main road, local road), traffic volume at each portion of the road network, road direction, intersection connections, and restrictions at each intersection or turn of the road.

Using the GPS trajectories to generate a road map is a challenging task due to: (1) the errors caused by the GPS devices, (2) different sampling rates on which the

GPS data is collected, (3) differences in the number of trajectories per road segments where the main roads have more trajectories than the local roads [49, 50], (4) the need to determine the road intersection locations and directions.

Several studies aim to generate road maps from GPS trajectories [51, 52, 53, 54, 55]. However, the generated road maps of these studies are either too complex (in terms of the number of vertices and edges) with good coverage or too simple with poor coverage [56]. If a map is too complex, then route planning algorithms such as shortest path requires more time. On the other hand, if a map is too simple, then good route plans may be missed because of reduced map coverage. In addition, the focus of these studies were mostly on generating geometric road maps while ignoring the semantic information of roads, which are essential for many applications such as routing planning, traffic monitoring, GPS navigation systems, and location-based service (LBS) applications. In this chapter, we propose a novel method to generate road maps using GPS trajectories that is accurate with good coverage area, has a minimum number of vertices and edges, and includes several details of the road network.

Our contributions in this chapter can be summarized as follows:

- An effective and efficient method for identifying the locations of intersections and turns. Our method utilizes a line simplification algorithm, which overcomes the errors caused by (1) GPS noise, (2) points that have been repeatedly captured while the vehicle is stopped, and (3) different sampling rates.
- Our method generates the road map connectivity information to connect road segments and identify the shape of intersections and turns.
- Our algorithm extracts road map features such as road direction, road type (one-way, two-way), average speed, road length, and intersection restrictions.

- We demonstrate the accuracy of our algorithm using two real data sets and comparing with two baseline methods. The comparisons indicate that our algorithm is able to achieve higher F-score in terms of accuracy and generates detailed road maps that are not overly complex.

The remainder of this chapter is organized as follows: Section 3.2 describes related work for road map generation from GPS trajectories. In Section 3.3, we provide the terminology used in this chapter and the problem statement. Section 3.4 explains the steps of our method to generate the road map. Then, Section 3.5 describes the experimental evaluation that compares our algorithm with baseline algorithms. Finally, Section 3.6 concludes our work and discusses future work.

3.2 Related Work

Different methods to generate road maps form GPS trajectories have been introduced. Several surveys and comparisons of different approaches are provided in [56, 57, 58]. According to Ahmed et al. [56], approaches for road map generation from GPS trajectories can be classified into three categories: *Point clustering, Incremental track insertion, and Intersection linking.*

Point clustering methods consider the input as a set of points and use clustering methods to generate the road segments and connect them to generate the road map, as in [53, 59, 60].

Incremental track insertion methods, which are similar to map-matching methods, start with an initial empty map and insert one trajectory at a time until no more trajectories are found. The road map is built after all the trajectories are inserted. Examples of this method are the works in [51, 54, 61].

Intersection linking algorithms have two-steps to generate the road map. First, it detects the locations of intersection. Second, it connects intersections using tra-

jectories data to build the road segments. The algorithm by Karagiorgou and Pfoser [52] using the change in direction and speed between two consecutive GPS points to indicate turns. Then, it clusters these locations to infer intersections. After that, intersections are connected to build the road segments and create the road map. Using only the direction and speed for detecting turns may lead to detect locations that are not actual turns or miss actual turns locations. Fathi and Krumm's algorithm [62] finds the road intersections using a trained detector model on the ground truth data and connects the extracted intersections using the trajectories data. Ezzat et al. have an algorithm [63] that extracts locations of turn by using a line simplification algorithm and clusters these locations to find the intersections. Then, it builds the road segments by connecting the extracted intersections. Using only the line simplification algorithm to find turns may introduce locations that are not actual intersections or turns because the line simplification algorithm is not designed for trajectories data. Xie et al. introduce an algorithm [64] that detects the location of turns using a dynamic programming approach to find the longest common subsequences and divide it into sub-track to find the locations of intersections.

Our proposed algorithm in this chapter is based on the intersection linking approach. Our work is different from existing works in that we identify the locations of intersection by using a line simplification algorithm with *spatial-constraints*. Then, we use a *grid-based method* in order to find the actual locations of intersection despite the GPS noise, different sampling rates, and the varying numbers of trajectories traveled by road segments. In addition, we generate a directed road map with fewer number of vertices and edges that can be used in applications such as route planning, GPS navigation systems, or maps for devices with low memory.

3.3 Definitions and problem statement

Definition 1. *A Road Map is a representation of a road network. It is represented as a directed graph G = (V,E), where vertex V correspond to a geographical location point in the road network and edge E represents the road segment that connects two vertices.*

Definition 2. *A Road Network intersection is a location where the driver can change the vehicle direction in different ways and is shared by different roads.*

Definition 3. *A Road Network turn is a location where the driver only can make a change in one direction.*

Definition 4. *A Road Network segment is an edge of the road network that connects two adjacent turns or intersections to each other.*

Definition 5. *A Raw GPS trajectory T is a list of ordered points with location and time information for a specific traveling path.*

$$T = p_0, p_1, p_2, ..., p_n$$

where p_i=(x_i,y_i,t_i); (x_i,y_i) represents the geographical location (latitude,longitude), (t_i) represents the time, and n represents the number of GPS points in T.

3.3.1 Problem statement

Given a set of GPS trajectories $TS=(T_1, T_2, ..., T_i)$, we want to generate a directed road map that is similar to the road network in the real world based on the set of trajectories.

3.4 Road Map Generation Algorithm (RMG)

Our algorithm, called **RMG**, aims to generate a directed road map that is accurate with good coverage area, has a minimum number of vertices and edges,

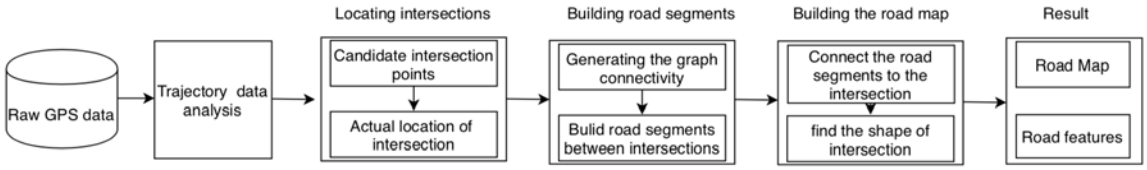


Figure 3.1: RMG Algorithm overview

and includes road features. It consists of four main steps, as shown in Figure 3.1. First, the algorithm analyzes the GPS data and provides more information such as direction, speed, time difference, road length, and so on. In the second step, it identifies the locations of turns and intersections by adjusting the Douglas-Peucker (DP)[8] algorithm. This is a line simplification algorithm. We enhance this method to incorporate spatial-constraints follow this by a grid-based method. After that, our method creates a road segment between intersections. Finally, it connects each road segment to the nearest valid turn or intersection using the graph connectivity table and finds the shape of each turn or intersection. The road features are extracted in different steps of generating the road map. In the remaining part of this section, we will explain each step of the algorithm in detail. Although there is a difference between intersections and turns, as mentioned above in Section 3, we are going to use the term intersections to indicate both intersections and turns. We will provide the method to distinguish between them.

3.4.1 Step 1: Trajectory data analysis

The raw GPS data received from vehicles include only the location (latitude, longitude) and time stamps. Important information (i.e. direction change, speed, road length,..) can be extracted by analyzing the raw GPS trajectory data, is the aim of this step.

To calculate the spatial distance d between two consecutive GPS points *(P1(lat1,lon1),*

P2(lat2,lon2)), we use the Haversine formula [65], which is an equation to calculate the great-circle distance between two points using latitudes and longitudes, as follows:

$$a = (sin^2(\frac{lat2 - lat1}{2}) + cos(lat1) * cos(lat2) * (\frac{lon2 - lon1}{2})$$

$$c = 2.atan2(\sqrt{a}, \sqrt{(1-a)})$$

$$d = R.c \tag{3.1}$$

Where R is the radius of the earth (mean radius = 6,371 km). From equation 3.1, we can calculate the speed s between two consecutive GPS points as follows:

$$s = d/\triangle T \tag{3.2}$$

Where $\triangle T$ represents the time difference between the two GPS points. For the change in direction, we measure the bearing rate [66] between two consecutive GPS point as follows:

$$x = cos(lat2) * sin(lat2 - lat1)$$

$$y = cos(lat1) * sin(lat2) - sin(lat1) * cos(lon2) * cos(lon2 - lon1)$$

$$bearing = arctan(x, y) \tag{3.3}$$

This analysis step will help in generating the road map, extracting road features, and adding semantic information to the output road map. For instance, the direction information will help in learning the intersection restrictions, road directions, and in building the road graph. Table 3.1 shows two consecutive GPS points of a trajectory and Table 3.2 shows the extracted attributes after the analysis step.

3.4.2 Step 2: Locating intersections

In this step, we aim to: (1) locate the road intersections, which are considered as one of the essential parts of a road network, and (2) identify if they are intersections

Table 3.1: Sample of raw GPS trajectory data

Pid	latitude	longitude	Time
624029	41.879176	-87.649627	1301806255
624042	41.879172	-87.649875	1301806259

Table 3.2: Extracted attributes after analysis step

Attribute	Values
Distance km	0.02059
Direction	268.759
Bearing	-91.25
Angle diff	1.004
Date time	4/2/11 23:50
Time diff	4 s
Velocity	18.48 km
Heading angle	SW 88.75

or turns. Distinguishing intersections and turns is a necessary step to know the road network connectivity and length. In order to achieve that, the RMG algorithm starts by generating the candidate points for an intersection and then uses the grid-based method to identify the actual location.

3.4.2.1 Candidate points of intersections:

In order to generate candidate points for the locations of an intersection, we use the DP algorithm that is a generalized technique for line simplification. It preserves the points that represent the shape of a line and removes all unnecessary points. The DP algorithm starts by creating a straight line that connecting the start point *(p_s)* and the end point *(p_e)* of a line. If all points of the line are within distance ε, then remove

all other points and use this new line. Otherwise, if any point of the line has distance large than ε, then the line will be split into two lines, which connect first point *(p_s)* to this point and from this point to the end point *(p_e)*. The algorithm repeats itself recursively until no more simplification is found. The parameter required for the DP algorithm is the maximum allowed distance ε, which can be estimated as the width of a road with three-lanes. The remaining points after the simplification process will be used as initial candidate points for intersections.

However, unlike the study in [63] were the DP algorithm is only used to find intersection points, we add spatial-constraints to filter the result of candidate points before considering them as final candidate points for intersections. The reason is that the DP algorithm is a generalized technique for line simplification and so it is not considering the special characteristics of trajectory data, which may lead to the removal of points that have essential information related to the behavior of vehicles [67]. Also, we distinguish between intersections and turns using the spatial-constraints, which are:

1. Intersecting with other trajectories in different directions. We check if the points of a trajectory returned from the DP algorithm are intersecting with other trajectories having different directions or not.
2. Changing in direction. We also check the difference in direction between the points returned from the DP algorithm and the successor and predecessor points of the same trajectory. In case of change in direction, we consider this point as a candidate point for intersections or turns.

Our RMG algorithm differentiates between intersections and turns as follows: If conditions (1) and (2) are both met, then the candidate point will be considered as intersection point, otherwise, if only condition (2) is met, then the candidate point will be considered as a turn point. Figure 3.2a shows the points of one GPS tra-

jectory and Figure 3.2b shows the candidate points after applying the DP algorithm with the spatial-constraints. Algorithm 1 shows the steps for generating the candidate points. In line 6, the DP algorithm is applied on a single trajectory; then the spatial-constraints are applied on the returned points (lines 7-15). Using the line simplification algorithm overcomes various GPS challenges, such as removing points that were captured while the car is stopped, and dealing with data sets with different sampling rates. For example, Figure 3.3a shows two trajectories with different sample rates where the first trajectory has high sample rates and the second 3.3b has low sample rates. The line simplification algorithm only keep the important points that are located at or near the road intersection in both cases.

Algorithm 1 Candidates points for intersections and turns

Input: Set of GPS trajectory Tr

Output: Candidates points for intersections and turns

$I \leftarrow \phi$ Intersection points

$T \leftarrow \phi$ Turn points

ε Thresholds for the DP

$\angle$ Thresholds for the change in direction

while Tr $\neq$ Null **do**

  $CP \leftarrow DP(Tr[i], \varepsilon)$ *Apply the DP on a single trajectory

  for P $\in$ CP($Tr[i]$) **do**

    $intersect \leftarrow Intersects((P(i), Tr(i)), Tr)$

    $dir \leftarrow IsChangeDirection(P(i), \angle)$

    if $intersect$ = True and dir = True **then**

      I.insert($P(i)$) //add this point as intersection point

    else if $intersect$ = False and dir = True **then**

      T.insert($P(i)$) //add this point as turn point

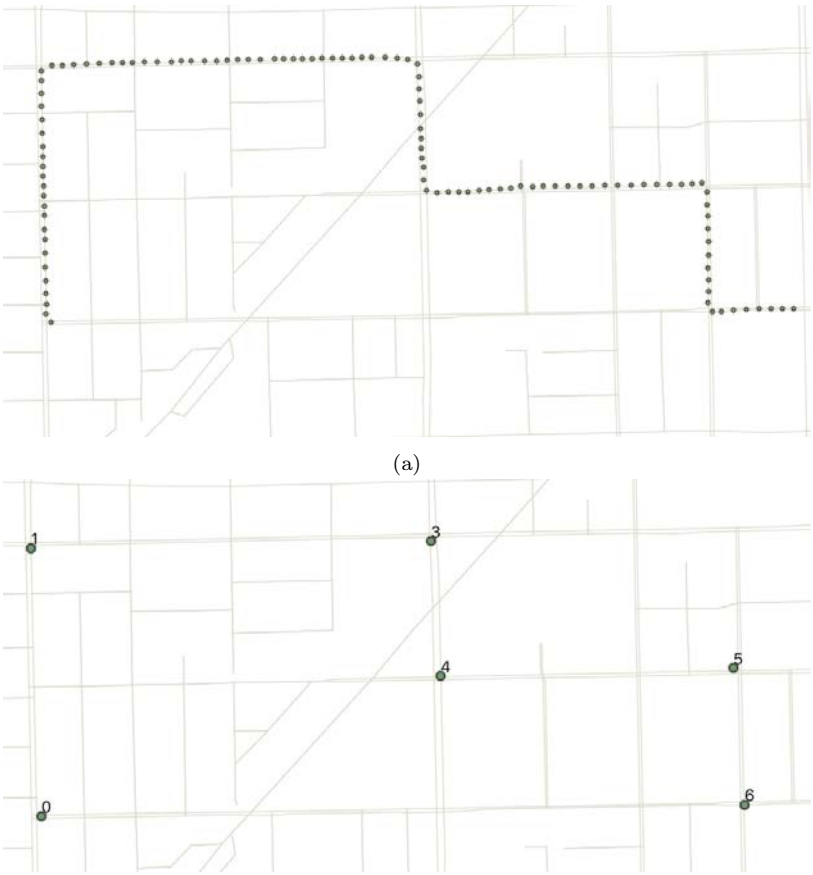
    end if

  end for

end while

3.4.2.2 Identifying the actual locations for intersection using the grid-based method:

After we identified the candidate points, we want to compute the actual locations for intersections. Many candidate points may not locate at the actual intersections because of missing GPS signals, different sampling rates, and some points have been captured before or after the intersection. For example, Figure 3.4a shows that



(a)

(b)

Figure 3.2: Extraction candidate points.a) GPS trajectory. b) Final candidate points

the cells with the *id* ($C_{i\text{-}2}$, $C_{i\text{-}6}$) include points that are not near the actual location of intersection. We introduce a grid-based method that aims to compute the actual location of intersections. It starts by dividing the area covered by the trajectories into a gird that consists of cells with identical size, Figure 3.4a, for example. Then, we count the number of candidate points in each cell. After that, we start with the cell C_i that has the highest number of points and we check the 8-neighborhood cells around C_i to count all the points inside these cells. The cells that have number of points above a certain threshold will be chosen to represent one intersection and a new *id* is assigned to these cells. Finally, we calculate the centriod of the points

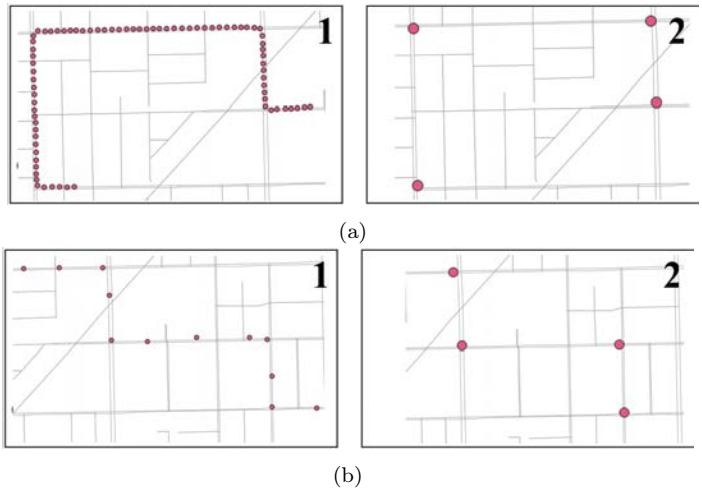


Figure 3.3: The result after applying the DP algorithm on two trajectories:(a-1) original trajectory with high sample rates (b-1) original trajectory with low sample rates (a,b-2) candidate points.

inside the chosen cells to represent the actual location of an intersection or turn in our generated road map. If all the points are marked as intersection points, it will be an intersection location, otherwise it will be turn location. Therefore, the result of this step represents the geographical locations of intersections and the candidate points that contributed to creating these intersections will have the same *id* as the intersections. Algorithm 2 and Figure 3.4 summarize the steps of finding the actual location of intersections. The minimum threshold for number of points in each cell that is going to be chosen was computed based on training sets and experimental evaluations.

Algorithm 2 Actual locations for intersection

Input: I Intersections and T turn points

Output: Actual locations of intersection and turn

1: G Grid of cells C
2: $s \leftarrow$ Cell size
3: NP Numbers of points in each cell
4: sort the cells in ascending order according to NP
5: $\varepsilon \leftarrow$ Thresholds for number of points in each cell to be chosen
6: **while** C $\neq$ Null **do**
7: // iterate over the cells in G
8: $CR \leftarrow$get the 8-neighborhood cells of $C[i]$
9: **for** Cells $\in$ CR **do**
10: **if** $Cells[i].NP > \varepsilon$ **then**
11: $Cells[i]$.id= $C[i]$.id
12: **end if**
13: **end for**
14: **end while**
15: FindCentriod(C) //compute the centriod for the cells with the same id to be the locations of intersection or turn

3.4.3 Step 3: Building road segments

So far we have identified the actual locations for all intersections in the generated road map. In this step, we build the road segments between intersections in order to connect them utilizing the trajectories that passed among these intersections. We start by creating the graph connectivity table (GCT) that lists the combinations of

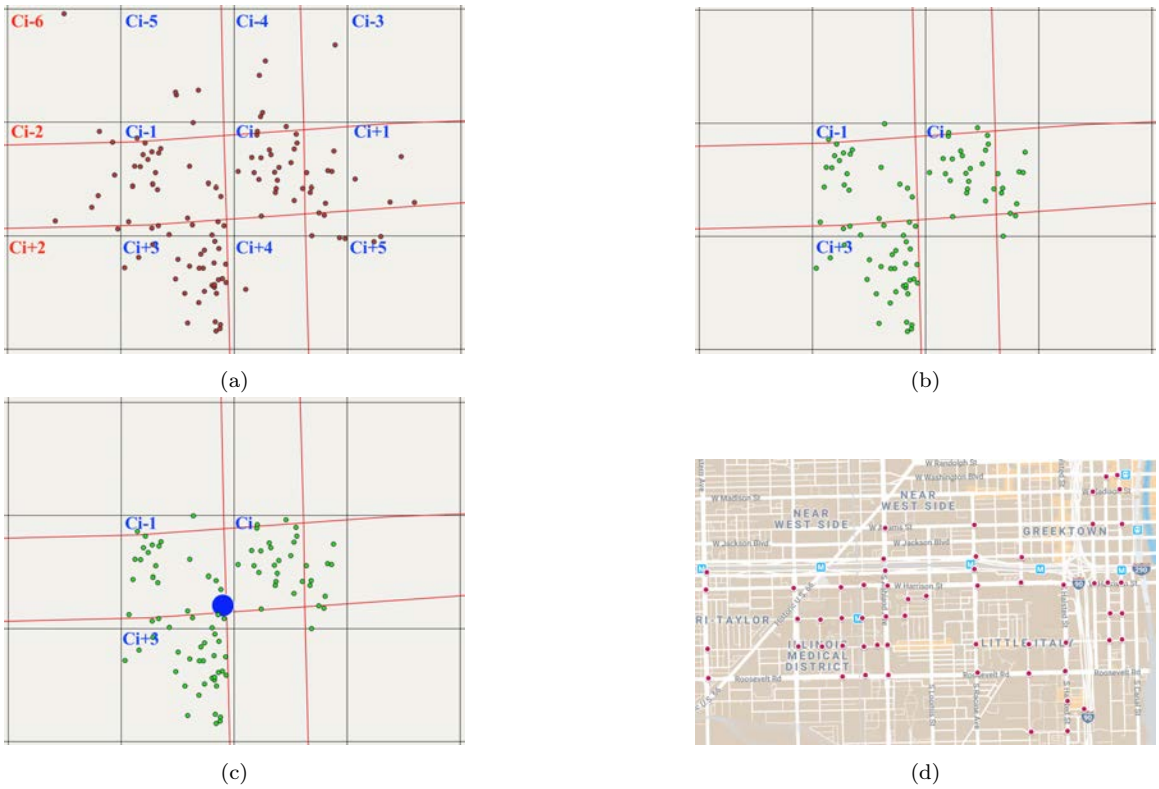


Figure 3.4: Steps to find the actual location of an intersection: a) Candidate points of an intersection, red lines represent aground truth map.b) Cells with large numbers of points. c) The Blue dot represents the extracted intersection location. d)locations of intersections (red dots) on Google Maps.

all intersections and counts the number of trajectories that pass through each pair of consecutive intersections. Based on GCT, we extract parts of the trajectories between each pair of intersections.

3.4.3.1 Generating the graph connectivity table (GCT)

In order to build the directed graph for the road map, we utilize the information of the cells in the grid from the previous step. That means for each trajectory; we know the cell *id* of each point within the grid. Also, we know the location of each generated intersection inside the grid and the candidate points from each trajectory

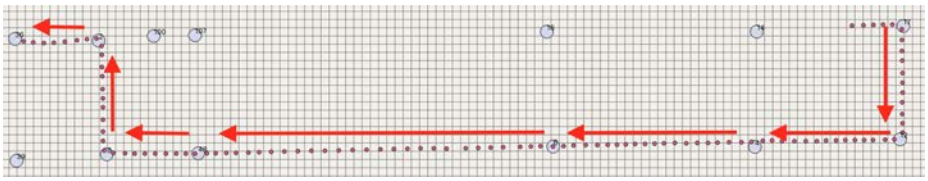


Figure 3.5: Extract the graph connectivity information

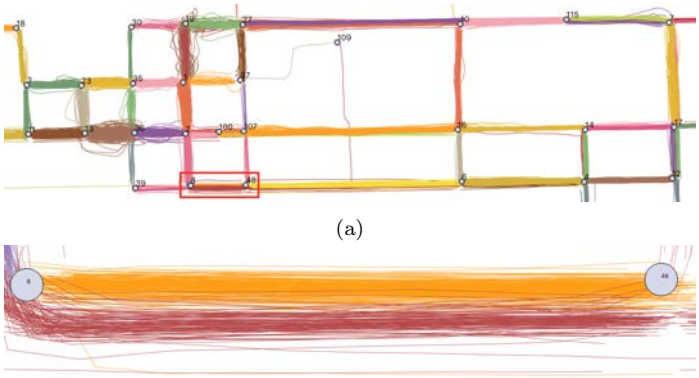
Table 3.3: Graph connectivity information example

vertex 1	vertex2	no of trajectories
1	2	286
2	8	129
2	100	80

that contributed to compute the intersections. With this information, we can generate the GCT. The algorithm checks for each trajectory, if it has points that helped in creating a specific intersection or passed by a specific intersection. It adds the intersection *id* to these points. After that, we go through each trajectory to know the graph connectivity. Algorithm 3 shows the steps of generating the GCT. For example, the trajectory, in Figure 3.5, has eight points associated with eight intersections (i.e. P_5:17,P_{15}:12, P_{20}:4,..) and the RMG algorithm produces the graph information as the example shown in Table 3.3.

3.4.3.2 Creating road segments between intersections:

Based on the GCT information, we extract a portion of each trajectory between two intersections. Figure 3.6 shows an example of the parts of trajectories between intersections. We create a cluster that has all portions of trajectories between two specific intersections in the same direction together. Then, we create a buffer around each cluster and we pick the middle trajectory to represent the road segments. After that, we apply the DP algorithm to simplify the line, which will produce a road map



(a)

(b)

Figure 3.6: Portions of trajectories between intersections. a)Road segment between intersections. b) The Road segments between two intersections in the red box

with fewer vertices at the end. Since the number of trajectories is different in each part of the road network, as shown in Figure 3.6a, our algorithm is able to build a road segment between intersections even if few trajectory segments exist. Figure 3.7 (1-3), shows the steps of creating the road segments between two intersections.

3.4.3.3 Extracting road features:

Using the information from the analysis step and the GCT table, we are able to extract some of the road features such as type of roads (one-way, two-way,..) directions of road segments, average speed in each portion of the road, intersections restrictions, and the number of trajectories passed between two intersections.

3.4.4 Step 4: Building the road map

The goal of this phase is to connect each road segment to the right intersections in order to build the final road map. From the previous steps, we extract each road segment between intersections using the GCT. However, most of the portions of trajectories have a length slightly different than the actual length between intersections because of approximations. Therefore, by using the GCT, we connect the portion

Algorithm 3 Generate graph connectivity information

Input: Set of GPS trajectory Tr, intersection location I

Output: GCT graph connectivity information

1: $GCT \leftarrow \phi$ Graph Connectivity table
2: CP Candidate points
3: **while** Tr $\neq$ Null **do**
4: **for** P $\in Tr[i]$ **do**
5: **if** $P[i]$ is CP or $P[i]$ has same cell id as I **then**
6: $P[i]$.insert($I.id$)
7: **end if**
8: $PI \leftarrow$ Get only P of $Tr[i]$ that has $I.id$
9: **for** p $\in PI$ **do**
10: GCT.insert($p[i].id$, $p[i+1].id$, $GCT.count + 1$)
11: **end for**
12: **end for**
13: **end while**

of each road segment with its actual intersections locations. For example, after we reach Figure 3.7 step number 4, we connect the portion of the trajectory segment to intersection 7 and 17 to create the road segments.

3.4.4.1 Identifying the shape of road map intersections:

The goal of this step is to update the generated road map and add detail to each intersection (shape and turn restrictions) to be similar to the actual intersections. We refine the topology of each intersection to find the shape without adding new segments. By using the direction information of each trajectory that passed through

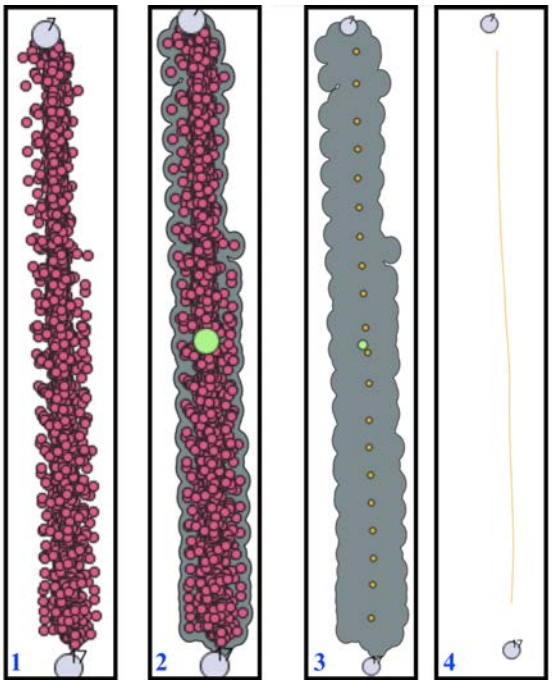
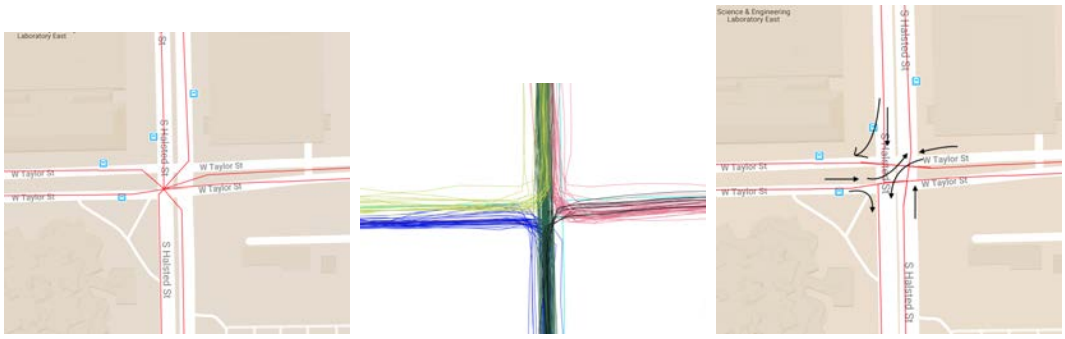


Figure 3.7: Steps for extract and create road segments between intersections

a specific intersection, we group all segments in a specific direction to identify all the allowed turns and refine the intersections accordingly. Figure 3.8a shows different road segments connected to the intersection and Figure 3.8b shows the different groups of trajectories where each color represents a direction. Figure 3.8c shows the final shape of the intersection and the allowed turns after the refinement step.

3.5 EXPERIMENTAL EVALUATION

In this section, we evaluate the accuracy of the proposed algorithm. First, we describe the two data sets that we used and the baseline methods to compare with our method. After that, we explain the evaluation methodologies on the generated road map. We then provide the results of our method as compared to the baseline algorithms.



(a) (b) (c)

Figure 3.8: Steps to refine intersections. a) Intersection before the update step. b) Using the direction to learn the shape and turn restrictions, where each color represents a directions. c) Final shape and turns allowed.

Table 3.4: Real data sets information

Data set	Chicago	Athens
No. of trajectories	889	129
Average sampling rate (sec)	3.6	34.7
Average speed (km/h)	33.14	19.55
Length(km)	2863	443

3.5.1 Data sets and comparing algorithms

3.5.1.1 Data sets:

We use two real data sets with different sampling rates and from different cities to evaluate our algorithm (Table 3.4). These data sets are:

1. *Chicago* data set [53, 58]: this data set consists of 889 trajectories that were collected by university shuttle buses, which cover an area of 7km x 4.5 km and the total length of 2863 km. Each trajectory has between 100 to 363 sample points. The average sample rate is 3.6 seconds with a range between 1s to 29s, which makes this data set a high sampling rate data set.

2. *Athens* data set: this data set consists of 129 trajectories that were collected by school buses, which cover an area of 2.6 km x 6 km and total length reaches 443 km. Each trajectory has between 13 to 47 sample points. The average sample rate is 34.7s with a range between 20s and 30s, so this has low sampling rate.

The OpenStreetMap (OSM)[68] is used as the ground-truth map. Since the trajectories do not cover all the streets in the ground-truth map, we manually remove any part of the ground-truth map that is not used by at least one trajectory. The data sets and the ground-truth map are obtained from the website http://mapconstruction.org by [56].

3.5.1.2 Comparing algorithms:

we compare our algorithm with two baseline methods, Karagiorgou's method [52] and Ahmed's method [51]. Each method follows a different approaches to map construction from GPS. Karagiorgou's method [52] is considered as *Intersection-linking* method while Ahmed's method [51] is consider as *Incremental Track Insertion* method (as described in Section **??**). The implementation of both methods are publicly available by [52, 69] and at the website http://mapconstruction.org by [56], which we used to compare with our algorithm.

Each algorithm has different parameters settings. Karagiorgou's method has the following parameters: angular difference = 12.5°, mean speed = 40km/h, turn cluster threshold 50 meters, and intersection clustering threshold 25m. For Ahmed's method, only one parameter is used to cluster sub-trajectories that is ϵ=80m for *Chicago* and 90m for *Athens*. We use the values of the parameters according to [56]. For our algorithm, we conduct different experiments to find the appropriate values for the parameters. *Locating intersection* (Section 4.2) has the following parameters: (i) the line simplification algorithm (DP) that is used to generate the candidate points

for intersections has the maximum allowed distance parameter, which was set to 40m. (ii) Cell size was set to 25m * 25m by analyzing the candidate points, which in most of the cases are within 25m distance from each other; in addition, the average of intersection area is around $125m^2$. (iii) The number of points in the cell in order to be selected should have at least 15% of the total number of points. (iv) change in direction $\angle = 35^{\circ}$.

3.5.2 Evaluation methodologies

To determine the accuracy of our generated road map, we use four evaluation methods. The first method is used to evaluate the geometry and topology of the generated map (GM), similar to the described method in [58]. In order to do that, we compute the precision and recall values to find the F-score. *Precision* (equ. 3.4) is computed as the matched length from the generated map *GM matched* to the total length of the GM. *Recall* (equ. 3.5) is computed as the ratio of *GM matched* to the total length of the ground truth map (GT). To compute the *GM matched* we extract two segments from the ground truth map that connect by an intersection and we check if any part of the GM match to these segments within a specific distance threshold. In order to ensure the connectivity of the generated road map, we count only the matched part from GM to the GT if it is in the same direction as the GT. The F-score (equ. 3.6) is computed as follows:

$$precision = \frac{\text{GM matched segments}}{\text{GM all segments}} \tag{3.4}$$

$$Recall = \frac{\text{GM matched segments}}{\text{GT segments}} \tag{3.5}$$

$$F - score = 2 * \frac{\text{precision} \times \text{Recall}}{\text{precision} + \text{Recall}} \tag{3.6}$$

The second method is used to check how accurately the location of intersections in the generated map compares to intersections in the ground truth map within a specific distance threshold. This measure checks the geometric accuracy of the extracted intersections. We also compute the F-score for this measure as follows:

$$precision = \frac{\text{GM Matched intersections}}{\text{Total number of extracted intersections in GM}} \tag{3.7}$$

$$Recall = \frac{\text{GM Matched intersections}}{\text{Total numbers of intersections in GT}} \tag{3.8}$$

GM Matched intersections are counted as the number of intersections that matched to the ground truth intersections within a specific threshold distance. The third measure is used to determine the complexity of the generated road map in term of the number of vertices and edges. A road map with fewer number of vertices and edges can be useful when it is used in small devices with low memory, low power, and can produce the results of map searching efficiently. Finally, we use the visualization method to draw and compare our map with the baseline algorithms' maps and we provide detailed information on the generated road map.

3.5.3 Results

3.5.3.1 Accuracy of the generated road map:

The result of this evaluation method for the *Chicago* data set is shown in Figure 3.9a. It shows a high value of F-Score for our method, which indicates the accuracy of our algorithm. We can see that the proposed algorithm with varying matching thresholds distance outperforms the two baseline algorithms. In addition, Figure 3.9b and 3.9c show that our algorithm achieves high values of precision and recall values compared to Karagiorgou's algorithm [52] and Ahmed's algorithm [51]. Our algorithm exceeds Ahmed's algorithm by almost 20% in precision value. Karagiorgou's algo-

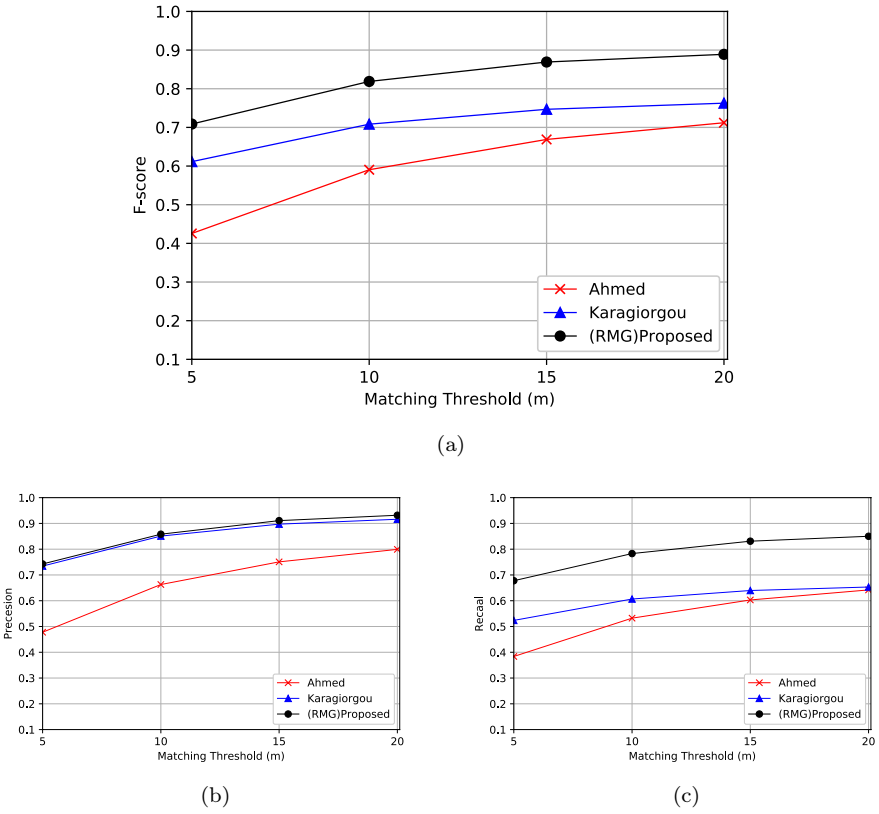


Figure 3.9: Comparison of F-score for the *Chicago* data set. a)F-score. b)Precision value. c)Recall value

rithm has high precision value when we compare it with our algorithm. However, it has lower recall since it does not differentiate roads in a different direction, which leads to cover a small part of the ground truth map. In addition, Figure 3.10 shows the F-score for the *Athens* data set where our algorithm also achieved higher F-score comparing to other two algorithms. The rank of the F-score on *Chicago* data set was higher than *Athens* due to the different number of trajectories, which is 889 trajectories in *Chicago* compared to 129 in *Athens*, and number of points in each trajectory. Overall, the F-score indicates that our algorithm achieves high accuracy on trajectories from different data sets, sampling rates, and the number of trajectories.

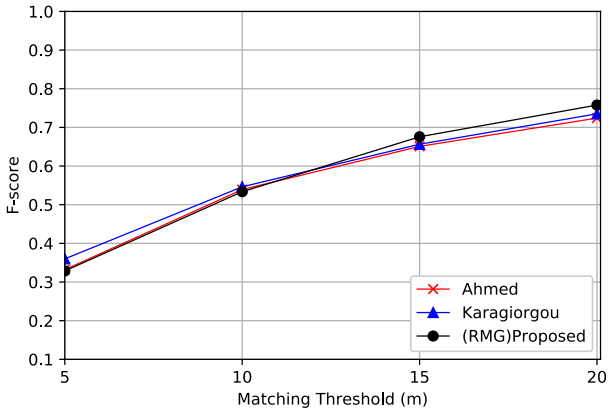


Figure 3.10: Comparison of F-score for *Athens* data set

3.5.3.2 Accuracy of the generated road intersections:

For the second evaluation method, we compare the generated road intersections with the ground-truth road intersections using the *Chicago* data set because it has more trajectories and intersections. We only compare with Karagiorgou's method because Ahmed's algorithm does not explicitly identify intersections. The ground-truth map has a total of 65 intersections and our algorithm extracted 60 intersections while it missed five. The Karagiorgou's algorithm extracted 77 locations of intersection, which results in a high recall value but lower precision and eventually low F-score. Figure 3.11 shows that our algorithm achieves high accuracy in extracting the road intersections and identified the actual locations within small distance deviation comparing to Karagiorgou's algorithm. More than *65%* of the extracted intersections are within *10m* distance from the actual location of the intersections, which indicates the accuracy of our algorithm.

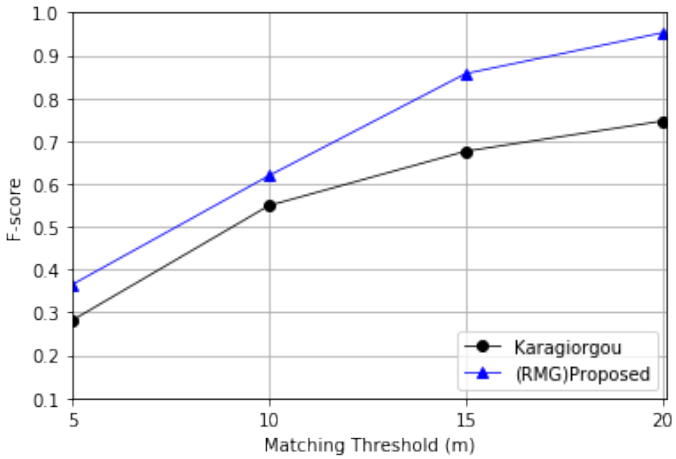


Figure 3.11: F-score for locations of intersections

3.5.3.3 Complexity of the generated road map:

In this evaluation measure, we show the complexity of the generated road map in terms of the number of vertices, edges, and the total length. Table 3.5 shows a summary of the generated road map complexity. Our algorithm on the two data sets generates a road map with lower complexity comparing to the baseline algorithms. The total length of our generated road map is high since we identify roads in different directions. From the above evaluation methods, we can see that our algorithm is able to generate a road map that is accurate with fewer number of vertices and edges.

3.5.3.4 Visualization of the generated road map:

For this method of evaluation, we draw our generated road map and we compare it with the baseline algorithms. For the *Chicago* data set, Figure 3.12 shows that our algorithm produces high quality and detailed map comparing to Karagiorgou's algorithm and Ahmed's algorithm. Our method distinguishes correctly the road type (one-way, two-way) and direction while other algorithms either generate one solid line for each segment (i.e. Figures 3.14b,3.14c), multiple lines for one segment (i.e.

Table 3.5: Generated road map complexity

Athens dataset	#Vertices	#Edges	#Length (Km)
Ahmed	344	378	37
Karagiorgou	660	637	37
Our method	278	305	48
Chicago dataset			
Ahmed	1195	1286	43
Karagiorgou	596	558	37
Our method	268	315	49

Figures 3.14g), detect road segments that do not exist (i.e. Figures 3.14f,3.14g), or misses road segments i.e. Figures 3.14f) . Furthermore, Figure 3.14e shows an area that has noisy GPS data, high density, and disparity in the number of trajectories in which our algorithm, Figure 3.14h, provides accurate map results in such an area and eliminates the trajectories with outliers. Since these trajectories will not be located within the same cells that have the intersections, this will lead to generating the final road segments from trajectories that are near to each other and pass by the intersections. For the *Athens* data set, Figure 3.13 shows the generated road map. While Ahmed's method produces a better map for *Athens* data set, our algorithm has high F-score value since we extract accurate intersections' locations then we choose the middle trajectory to represent the road segment.

3.5.3.5 Generated road map features:

As one of our work's aim is to extract detail in the generated road map, in this section, we show the road map features that we are able to extract from the trajectories. These features are extracted during the process of generating a road

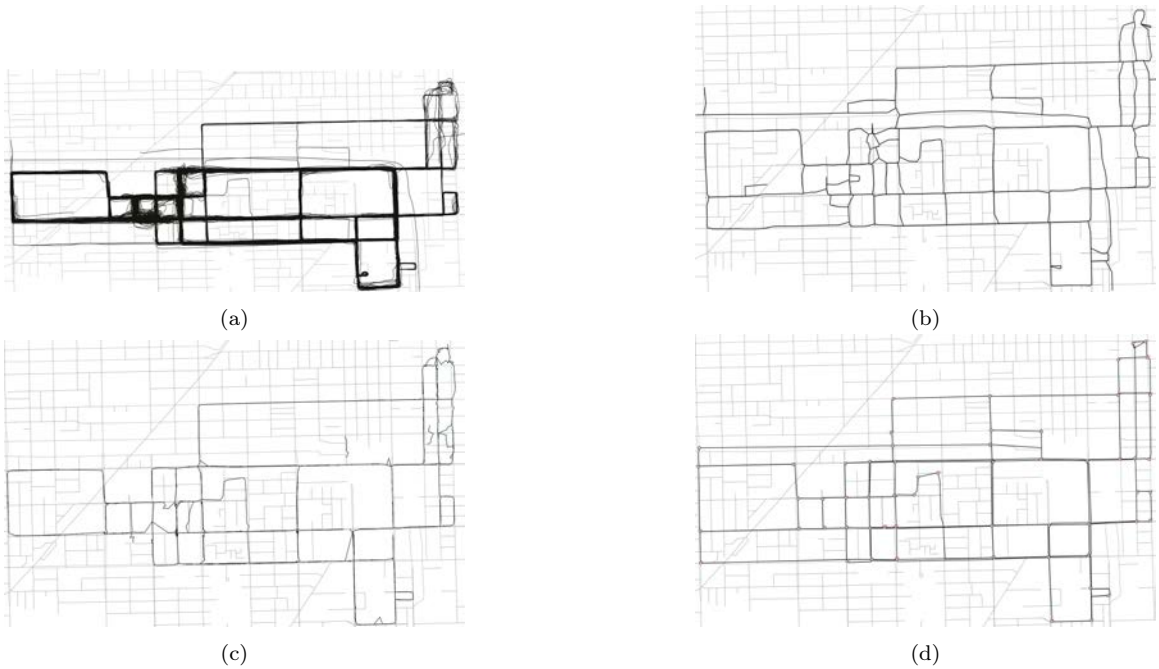


Figure 3.12: The generated road map for *Chicago* data set. a) GPS trajectories; b) Ahmed's method. c)Karagiorgou's method. d)RMG proposed

map from the first step of our algorithm *Trajectory data analysis* and the *building road segments* step. The features that we extracted are:

1. Average speed: since we divide the roads into segments where each segment is located between two intersections, we find the average speed from the trajectories portion in each road segment.
2. Road segments length: this attribute represents the length of each road segment between two intersections by computing the trajectory length between two intersections.
3. Number of cars traveled per road segment: this attribute can be found in the GCT information. It can be used to differentiate main roads from local roads and show the roads with heavy traffic.
4. Road type: this feature shows the road type (one-way, two-way) when the trajectories move in opposite directions. For example, in Figure 3.15, the road

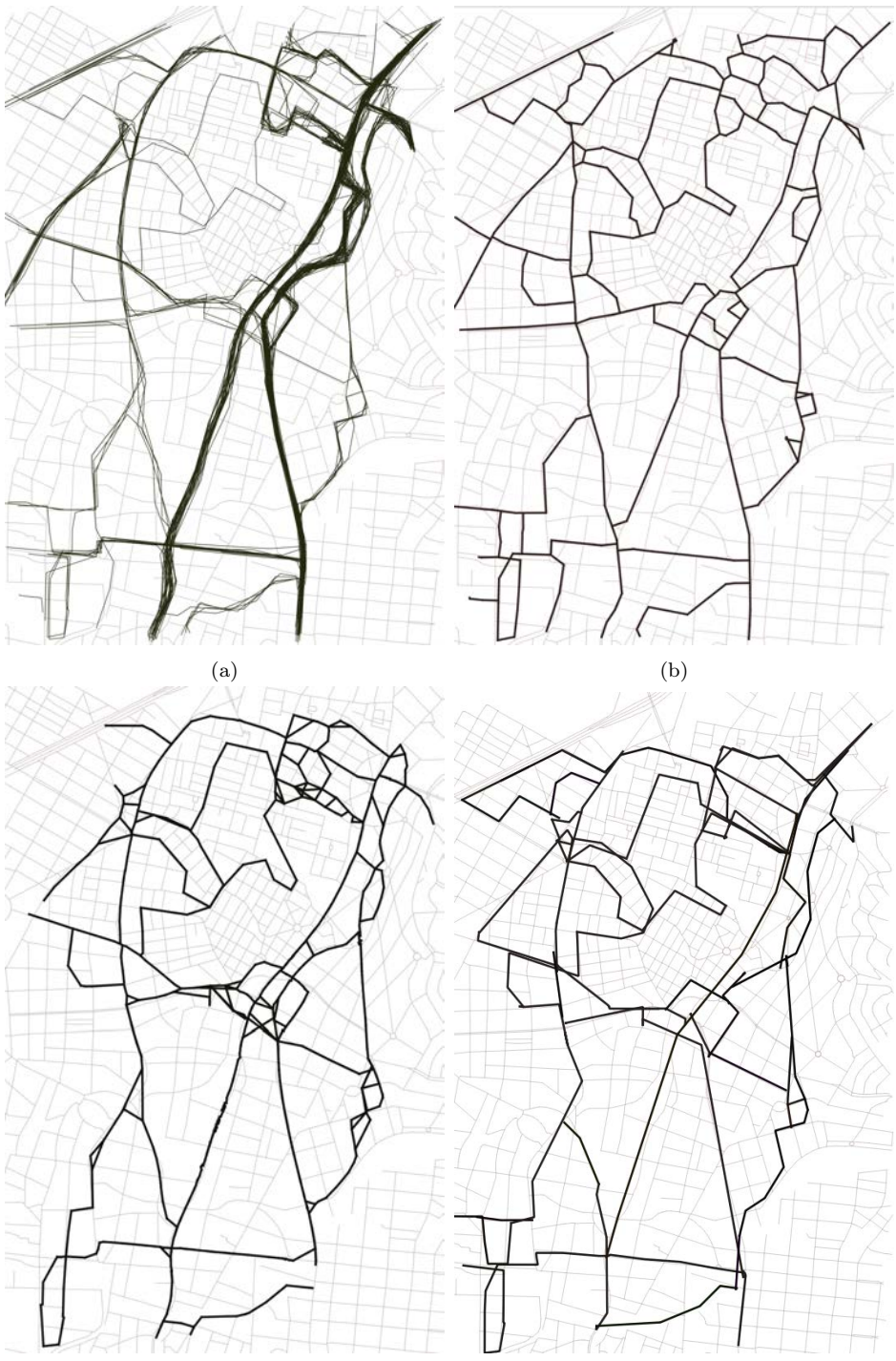


Figure 3.13: The generated road map for Athens data set. a) GPS trajectories; b) Ahmed's method. c)Karagiorgou's method. d)RMG proposed

(a) GPS trajectories (b) Karagiorgou (c) Ahmed (d) RMG proposed

(e) GPS trajectories (f) Karagiorgou (g) Ahmed (h) RMG proposed

Figure 3.14: Visualization comparison of the road segments on the Chicago data set

Table 3.6: Example of generated road map details

s.id	Avg speed (km/h)	length(m)	No. cars	Direction
1 to 19	25	290	46	N to NW
1 to 2	31	257	286	SE to SE
13 to 11	31	213	206	W to W
5 to 18	35	757	153	E to E
35 to 1	25	188	260	N to N

segment between intersection (8,39) is considered two-way road while the road segment between (23,35) is considered one-way road.

5. Road segments direction: this feature shows the direction of each road segment.
6. Intersections restrictions: using the trajectories' direction at each intersection and the GCT information, we find the turns restrictions.
7. Intersection connectivity information: this feature is used to show the intersection connectivity information with other intersections.

Figure 3.15 and Tables 3.6,3.7 show examples of the extracted features from the generated road map and the intersection information.

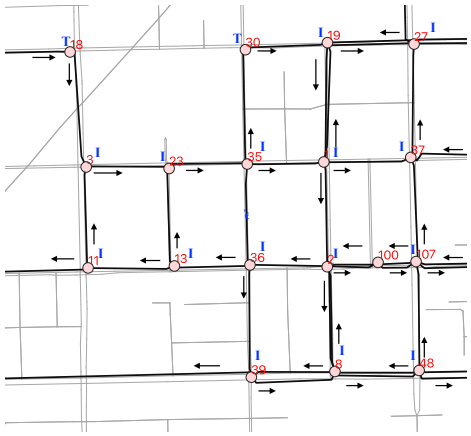


Figure 3.15: Generated road map with direction and type

3.5.4 Discussion

Our method achieves high accuracy in determining the locations of turns or intersections and generating accurate road map compared to the baseline algorithms. In addition, it extracts road features that provide information about the generated road map. The output map is beneficial to construct a map of an area that has no map or update an outdated road map. Furthermore, our method generates a road map with fewer vertices and edges that can be used in small devices with low memory, low power, and produce the results of map searching efficiently. However, there is some improvement needed to be done to our method. First, if a trajectory starts before an intersection and end before the next one, it will not be added to any road segment, since we only extract part of the trajectories that passed between two intersections. As a solution, we can check the nearest intersection to the last point of the trajectory using the direction information and the GCT table and connect this trajectory to the nearest intersection. Second, part of the road map will not be

Table 3.7: Generated intersection details

I.id	Type	Connect to	Allowed turns
19	I	1,27	L,R,S
18	T	3	R
23	I	35	S,R

I: intersection. T: turn. R:right. L:left. S:straight

generated if it is before an intersection. Our algorithm only creates road segments between intersections. To overcome this issue, we can group all the trajectories that start before a specific intersection and do not pass by any other intersections and generate road segments for these. Finally, we use the DP algorithm to simplify a trajectory between two intersections to represent the road segment, which works well in most cases and leads to a road map with fewer number of vertices and edges. A different smoothing method can be applied to generate road segments.

3.6 Conclusion

In this chapter, we presented an algorithm for generating a road map and extracting roads' features automatically from GPS trajectories. Our algorithm is based on the intersection linking approach. It detects the location of intersections and turns of a road network using the DP algorithm with spatial-constraints and the grid-based method. Then, a directed road segment is created relying on the graph connectivity information to connect the intersections and build the road map. Furthermore, we provided a refinement step to indicate the shapes and restrictions of intersections. Our algorithm also extracted the road map features such as average speed, road type, number of trajectories per road segment, road segment direction, and intersection connectivity information. Experimental results showed that our algorithm achieved

higher F-scores and generated a road map with fewer vertices and edges compared to the baseline methods. Furthermore, we showed our algorithm is able to extract road map features that were not considered in previous works.

In our future work, we intend to work on online road map generation from GPS data to provide a real-time update to the road map. Also, we intend to provide a machine learning model to better estimate the parameters of our algorithm for each GPS data set. In addition, we aim to extract more road details such as the locations of bridges, tunnels, or parking lots as the data sets become available.

CHAPTER 4

A Framework for Road Map Classification Using Machine Learning

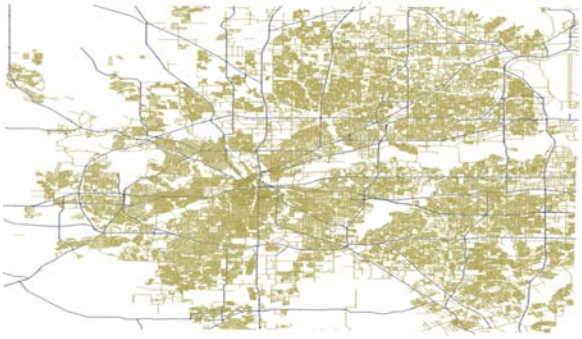
4.1 Introduction

Advancement in technology leads to the availability of different road network datasets, which offered by organizations such as governments (i.e., Topologically Integrated Geographic Encoding and Referencing system TIGER), Volunteered Geographic Information (VGI) (i.e., Open Street Map (OSM)), and commercial companies (HERE, Google map). It can be utilized by several applications such as in transportation systems for creating, classifying, and updating the road map, spatial data integration of two different road map data sets, and location-based applications.

The process of creating maps for roads network is still a challenging task. Previous studies create the road maps for road network using different approaches such as using satellite imagery to extract the road maps [70] [71] [72], using Global Positioning System (GPS) to collect points from the trajectory of moving objects and create the road network, and remotely sensed data [73] [74] [75]. However, to use the road map datasets for different applications, it required some pre-processing to prepare the data. One way of pre-processing is to classify a road map.

Looking at different approaches to creating the road map, we encourage to assist this process by proposing a framework for road map classification using different machine learning algorithms. It aims to distinguish the roads based on their types, which either are main roads that expand over multiple counties or local roads that do not exceed the counties' or cities' boundaries. The data set used in the framework obtained from the Texas Department of Transportation (TxDOT), which represent

Figure 4.1: Main roads (dark blue) and local roads (yellow)



the roads in the state of Texas. Figure 4.1 shows the main road (dark blue) and the local road (yellow). Our framework can be used as pre-processing for any applications that use the road types among its processes, such as in generating candidates for a similar road for spatial data integration, post-processing in classifying the extracted roads from satellite imagery, or any other methods of creating and classifying the road maps. Besides, it can be used to update information on the already existing road map database in case new information is available.

To create the framework, we examine different machine learning algorithms (Logistic Regression, K-Nearest Neighbor (KNN), Naïve Bayes, and Support Vector Machine (SVM)) to find the best algorithm that solves our problem. The main contribution of this chapter is to provide a model that can categorize roads of different types.

The rest of this chapter organized as follows: in section 4.2, we discuss related work for creating road maps. In section 4.3, we describe the road map classification framework. Next, in section 4.4, we present the result of the machine learning algorithms using several approaches such as learning curve and accuracy. Section 4.5 provides a discussion on the proposed framework. Finally, in section 4.6, we sum up with the conclusion and future works of this chapter.

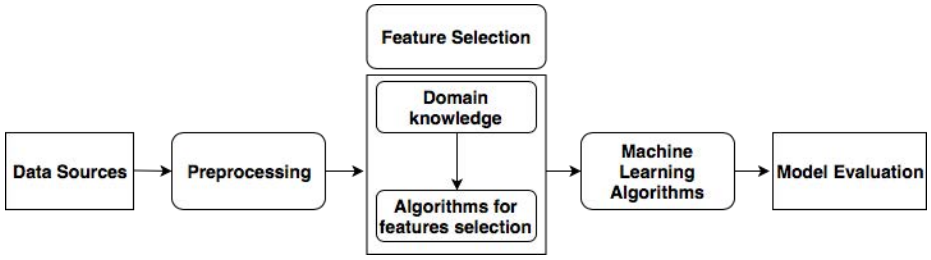
4.2 Related Work

Different studies on creating and updating road maps have been introduced. One approach is to use the satellite or aerial images for detecting and extracting different road map. The study in[76] provides a method to extract the road from satellite images using the geometry shape of the road. Another study [77] proposes a method to detect the straight lines in aerial images. In [78], the authors develop a method to extract the main roads from satellite images. The authors of [79] able to extract road centerline using a training spectral-spatial classifier. Several surveys on creating road maps using satellite images are provided in [80, 81, 82]. Another approach for constructing road maps using the data collected from GPS devices for moving objects (i.e., car trajectories) along the road network, as discussed in Chapter 3. The study in [79] constructs the road map using GPS data set by by creating the road segments and identifying junctions then constructing the road map. In paper [73], the authors able to cluster the trajectories in the same area and generate the road network accordingly. Several surveys and comparisons of different approaches are provided in [56, 57, 58]. Besides, the spatial data integration of different road map data sets can be used to update a digital road map. Comparing those approaches are out of the scope of this chapter. Our framework can assist both methods when the classification of the road map is required.

4.3 Framework for Road Map Classification

In this section, we describe different parts of our framework for road map classification, as shown in figure 4.2. We start by explaining the four machine learning algorithms that we use. Next, we explain the data set. Then, we move to the feature

Figure 4.2: Overview of Framework for Road Map Classification



selection to do the dimension reduction. After that, we implement machine learning algorithms and report the result.

4.3.1 Machine learning algorithms

We selected four machine learning algorithms to apply the classification tasks on our road map data set. 1) K-Nearest Neighbor algorithm (KNN), which considers a lazy learning algorithm and instance-based algorithms. In KNN, distances between different points are computed and the points which lack class label are assigned to the class of the k- nearest neighbor. 2) Support Vector Machine (SVM), which is designed for binary classification since our data set has only two classes. 3) Naïve Bayes (NB) which is considered as a generative model. 4) Logistic Regression (LR) as a discriminative algorithm. We pick four different algorithms and each technique has different characteristics. These algorithms are simple to implement and the result can be easily interpreted.

4.3.2 Data description

The data set is from the Texas Department of Transportation (TxDOT) [83], which is the road map for the State of Texas that has 640124 instances and 152 features. It has many classes besides the main and local roads. Since we are only interested in those type, we extracted them only and eliminate the rest.

Table 4.1: Data Pre-processing

Feature	Action
Length	Sum
Number of lanes	Frequency, Varince, Average
Row	Average
Surface width	Avearge,Mean
Road Width	Frequency,Avearge

4.3.2.1 Data pre-processing

The data set has many issues. The main two issues that we deal with are missing information for some features, and some roads have multiple records. To solve the first problem, we eliminate the features with missing information before that we try to fill some of the information. However, we found out it impossible to come with the right information. For instance, the feature 'Max Speed' for local road does not include any information, since the TxDOT does not provide any information. We had to remove this feature. For the second problem, which violates the Identical Independent Distribution (I.I.D) property, we aggregate the multiple records for one road together. Table 4.1 shows the selected features and explain the action we take to do the aggregation.

4.3.3 Feature Selection

Before we apply any of the machine learning algorithms, we reduce the feature dimension using two approaches that are domain knowledge and implementation of

different algorithms for feature selection. In this process, we aim to reduce the feature dimension and eliminate unrelated and redundant features.

4.3.3.1 Domain knowledge

We start with the domain knowledge since the data set comes with clear metadata that describes each feature. By studying the metadata for the whole data set, we choose the most relevant features for our framework. For example, the Length feature provides the notion of distance, which helps the classifier to distinguish between main and local roads. The number of lanes and width features play a critical role in identifying main and local roads since some local roads have a greater or lower number of lanes than main roads or vice versa. In addition, we eliminate more than a hundred features that have no roll in classifying the road map. Examples of the features that we eliminate are the date that specific record added to the database, name of the administration responsible for maintaining the road, and district id. Later, we eliminate any feature that has no information, such as minimum speed and maximum speed.

4.3.3.2 Algorithms for features selection

There are several techniques to automatically select features for training machine learning algorithms. In our work, we use the wrapper method (forward feature selection, backward feature selection), which works by selecting a subset of features and train the model and compare the performance of different combinations of features together.

Table 4.2 lists the 10-fold cross-validation accuracy results for forward and backward sequential selection search methods. Backward sequential selection should normally produce better accuracy results when features rely on each other to predict.

Table 4.2: 10-fold cross validation accuracy for each search method

Algorithm	Forward	Backward
LR	93%	93%
KNN	94%	93%
NB	88%	85%
SVM	93.5%	93.6%

Table 4.3: Feature sets for each search method of classification algorithms

Algorithm	Forward
LR	{rowd_mean,surf_mode row_min_mean}
KNN	{row_mean,surf_mode, roadw_mean}
NB	{surf_mode, row_mean,surf_mode}
SVM	{length,row_mean,surf_mode}

However, when we apply those methods on our road map data set using KNN and Naïve Bayes algorithms, forward selection produces slightly higher accuracy, but we get equal accuracy results when we apply a logistic regression algorithm on our data set. Also, using the SVM algorithm with a backward sequential selection method gives a marginal increase over the forward method. Table 4.3 and 4.4 show the features that produce high accuracy after applying the feature selection algorithms. Feature *surfacew_mode* appears in all algorithms, which indicates that this is an important feature.

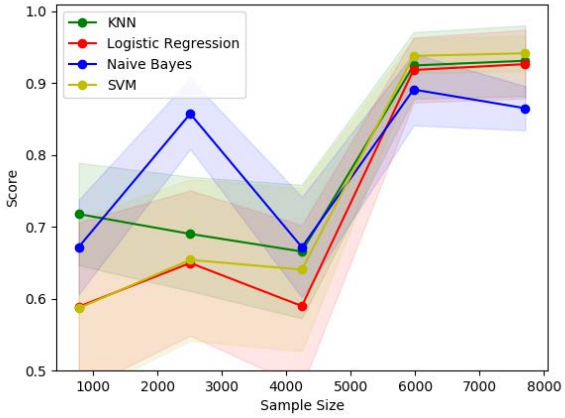
4.4 Algorithms Evaluation

In order to evaluate the machine learning algorithms, we use the learning curve, accuracy, and receiver operating characteristic (ROC).

Table 4.4: Feature sets for each search method of classification algorithms

Algorithm	Backward
LR	{surf_mode, surf_mean, roadw_mean}
KNN	{roadesign, row_min_mean,surf_mode}
NB	{avg_lane, surf_mode, row_min_mean}
SVM	{row_mean, surf_mean road_mean}

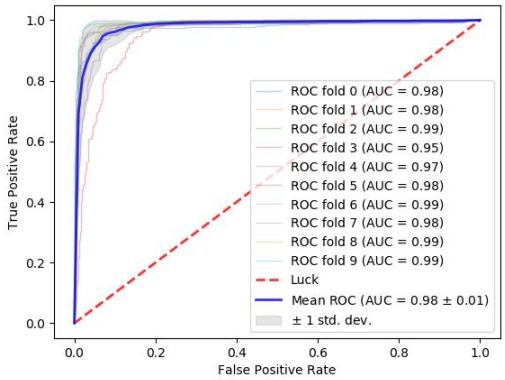
Figure 4.3: Learning Curve



4.4.0.1 Learning curve

Figure 4.3 illustrates the result of the machine learning algorithms that are evaluated on the road map data set. The data set is randomly partitioned using a 10-fold cross-validation technique. When the sample size reaches 8000 instances, SVM outperforms other machine learning algorithms. The accuracy that results from running KNN is slightly higher than the logistic regression's accuracy. In addition, Naïve Bayes works very well when the sample size is small, starting from 100 to 4000. It is expected from Naïve Bayes as a generative model to derive such results because it works very well when the size of the data set is small.

Figure 4.4: Receiver operating characteristic



4.4.0.2 Accuracy

The receiver operating characteristic (ROC) curve for SVM is plotted in figure 4.4, which shows multiple curves for each fold in the 10-fold cross-validation technique. SVM gives the high area under the curve (AUC) for each fold, which is closer to one. The average area under the curve is 0.98, with a confidence interval of -0.01 and 0.01. KNN and Logistic regression generate the same area under the curve, 97%. However, logistic regression produces a lower area under the curve result, which includes one fold that causes the average to be lower than other algorithms. The first fold generates 90% area under the curve.

4.5 Discussion

As shown in the algorithms evaluation section, all the algorithms provide good accuracy for classifying the roads to main and local roads. Our evaluation also shows that the accuracy is similar even with different algorithms on the same set of features and training/ testing. Such results show that our framework has a promising output when it incorporates with other methods such as constructing and updating maps for

the road map. Furthermore, this model can be used as pre-processing for spatial data integration of road maps to generate a candidate of the same road type semantically. Moreover, it can be used as post-processing after extract roads from satellite images. Nevertheless, there are still some issues with the road map data set. One of the problems is when some roads have missing feature data that may affect the result of our framework. Another issue is that there is more than one record representation for the same road (object), which violates the dependency condition and eventually impact the accuracy of the results. Hence, this framework can be improved by using different algorithms for feature selection. Also, the performance of the four machine learning algorithms needs to test before the features are selected and after on different data set.

4.6 Conclusion

In this chapter, we propose a framework using machine learning algorithms to classify the roads based on their types, whether they are main or local roads. It starts with pre-processing of the input data and then implementing several methods for feature selection. After that, we train our model by utilizing four different algorithms (KNN, SVM, Logistic regressions, and Naïve Bayes). The output of the framework is the roads labeled as main or local roads. The experimental result shows good accuracy in classifying the road maps. As future work, we intend to use the framework as an initializing process for generating similar candidate roads to apply spatial data integration. Also, the framework will be tested using a different dataset and machine learning algorithms

CHAPTER 5

Data Fusion of Heterogeneous Data Sources for Intelligent Transportation Systems

5.1 Introduction

Intelligent Transportation Systems (ITS) play a critical role in our daily life by enhancing road safety, minimizing travel time, reducing fuel consumption, and detecting incidents [84]. ITS applications can provide, for example, traffic forecast information, travel time estimation, incident prediction, and other applications. Providing accurate information for road status is considered as a challenging task due to the different factors that effect road conditions, which include drivers behaviors, road network structures, spatio-temporal dependencies, and external factors such as weather, incidents, traffic congestion, road closings, and nearby events [85, 86].

These days, big data from various sources (government agencies or crowd data sources) provide valuable information to transportation systems. These data can be acquired and collected from road network sensors (loop detectors), weather stations, incidents reports, congestion reports, points of interest locations, temporal information (time of day or day of the week), and social events. Relying only on one data source for ITS applications, such as road network sensors, which provide traffic flow information, may not be an effective way to provide traffic status [87]. Therefore, data fusion techniques of heterogeneous data sources may lead to better estimation of a situation, reduce the uncertainty from one data source, and provide more accurate information [87]. Thus, combining various data sources with traffic information can improve traffic prediction [84].

Several approaches and techniques have been studied for ITS applications. In recent years, data-driven models using deep learning approaches for transportation systems have achieved better results in traffic flow prediction, incidents prediction, and travel time estimation when compared to other methods as in [88, 89, 90, 91, 92, 93, 94, 95]. [96, 97] Some of the motivations behind using the deep learning approaches are utilizing high-performance computational power, which allows processing large amounts of data with different features. Furthermore, deep learning approaches could learn features with minimum knowledge of a specific domain. However, to train deep learning methods, the input data should be in large amounts and in a uniform representation, which is considered to be a complex and time-consuming task [98].

To benefit from deep learning approaches and utilize the availability of heterogeneous data sources for ITS applications, this chapter has two main goals, which are: (1) providing data fusion techniques to integrate and extract features from heterogeneous data sources with different characteristics (such as format and size) in a uniform representation to be ready for deep learning training approaches, and (2) generating preprocessed real-world traffic datasets and make them publicly available for other researchers. These generated traffic datasets can be used for different ITS applications such as traffic analysis and visualization, traffic forecast prediction, incident prediction, travel time estimation. The preprocessed data can and also be used for benchmarking different prediction approaches that used deep learning. We also describe with details the steps to generate the traffic datasets from heterogeneous sources to be used as guidance for other researchers.

The generated traffic datasets provide rich features such as *traffic flow, average speed, vehicle occupancy, weather conditions, incidents information, congestion reports, point of interest locations*. It also provides temporal features such as a month of the year, day of the week, an hour of a day, weekend vs. weekdays, and other

features for different road network locations. These features can allow researchers in different ITS fields to gain insight and apply their methods to solve ITS-related problems. Our contributions can be summarized as follows:

- Gathering heterogeneous data sources that include traffic information, weather information, temporal information, road network graph, and point of interest locations and introducing data fusion techniques that include semantic matching, map matching, spatial matching, temporal matching, and data association to extract features out of the collected datasets.
- Providing as an output of the data fusion techniques two traffic datasets with different characteristics that are publicly available.
- Utilizing the two traffic datasets and providing two applications in chapter 6. (1) *Traffic data analysis and visualization* where we create a data cube that provides in-depth analysis of the traffic datasets with a visualization tool to provide the results in different ways such as charts, maps, and tables. (2) *Traffic flow forecasting using deep learning* where we perform a comprehensive study on how different features can improve the traffic flow prediction models. The experimental results show that combing features with traffic data achieve lower Mean Absolute Error (MAE) comparing to using only traffic data for different deep learning models.

The remainder of this chapter is organized as follows: Section 5.2 describes related work. In Section 5.3, we provide descriptions of the different data sources. Section 5.4 explains the steps of our data fusion techniques to generate preprocessed traffic datasets.

5.2 Related Work

Data fusion (DF) techniques have been used widely in different domains and one of these domains is Intelligent transportation systems (ITS) [87]. Several definitions have been introduced for data fusion and we mentioned the most common definitions of them here:

The Joint Directors of Laboratories (JDL) [99] defines data fusion is as"*a process dealing with the association, correlation, and combination of data and information from single and multiple sources to achieve refined position and identity estimates, and complete and timely assessments of situations and threats, and their significance. The process is characterized by continuous refinements of its estimates and assessments, and the evaluation of the need for additional sources, or modification of the process itself, to achieve improved results.*"

Castanedo [100] defines data fusion as " *a combination of multiple sources to obtain improved information; in this context, improved information means less expensive, higher quality, or more relevant information".*

Sidek and Quadri [101] define data fusion as "*Data fusion deals with the synergistic combination of information made available by different measurement sensors, information sources and decision makers. Thus, sensor fusion is concerned with distributed detection, sensor registration, data association, state estimation, target identification, decision fusion, user interface and database management.*"

Data fusion goals are to combine data from heterogeneous sources to provide a better understanding of the situation, improve decision making, and provide more reliable result from many data sources instead of one data source [87, 102, 103]. DF techniques can be classified and categorized in several ways. One of the well-known classifications of DF is provided by Dasarathy [104] where the classification is based on the input and the output data types. It has five categories [100], which are:

1. Data In–Data Out (DAI-DAO): This is the basic data fusion method. The result of the data fusion also is raw data. However, the output data will be more reliable or accurate than the input data.
2. Data In–Feature Out (DAI-FEO): The data fusion methods use the input data from different sources to extract features and provide more information.
3. Feature In–Feature Out (FEI-FEO): The input and output to the fusion process are features to extract new features or improve existing features.
4. Feature In–Decision Out (FEI-DEO): The input to the fusion method is sets of features and the output is sets of decisions.
5. Decision In–Decision Out (DEI-DEO): The input to the fusion method is sets of decisions and the output is sets of decisions to generate new decisions or improve existing decisions.

Another classification of DF is developed by the U.S. Department of Defense, which is called the JDL model [99]. It consists of five levels [84, 100]. Level 0 - Source prepossessing: This level deals with data prepossessing and preparation from different sources and reduces the amount of data. Level 1 - Object refinement: This level uses the processed data from level 0 and fuses them by applying several functions such as spatio-temporal alignment, association, correlation, clustering, and others techniques. The fused data can be traffic flow from sensors, toll data, probe information from vehicles sensors, and weather stations. Level 2 - Situation assessment: This level combines data from other data sources and provides a higher inference level than level 1. Data sources can be incident reports, temporal information, events, weather report, and traffic patterns. Level 3 - Threat assessment: This level evaluates the impact of fusing different data sources. For example, the impact of incidents on the traffic flow. Level 4 - Process refinement: This level aims to reevaluate the data fusion process and improve it and add new data sources.

Previous paragraphs describe DF techniques in general. We now discuss DF techniques in ITS. These involve several functions such as temporal and spatial joining of raw data, data association, and data mining [84]. Algorithms that have been introduced for DF in ITS, for example, include Bayesian inference, Dempster-Shafer evidential theory, artificial neural network, fuzzy logic, Kalman filter or extended Kalman filter, Monte Carlo techniques, and particle filters [84]. Most of these algorithms focus on combing different sources of multiple input to provide a single type of output prediction. Several surveys about data fusion in general and in ITS applications are provided in [84, 100, 87, 101, 102].

A fusion of heterogeneous data using deep learning approaches to solve ITS problems has shown promising results by including traffic flow information and external factors such as weather data, incidents, traffic congestion, road closings, and events. Zhang and Kabuba [105] predicted traffic flow in an urban area by taking into consideration weather data along with with traffic flow information. A Gated Recurrent Unit (GRU) is used to predict the traffic flow and the result showed that the model produced higher accuracy when the weather data is included. Dunne and Ghosh [106] proposed neurowavelet algorithm to predict traffic forecast with rainfall information. The model produced better results when using rainfall information with traffic data during rainfall periods. Koesdwiady et al. [96] proposed a deep belief network (DBN) to predict traffic flow with weather information. The traffic flow and weather information are fused at the decision level using DBN to predict the traffic flow; therefore, better results are obtained with weather information instead of only traffic information. Jia et al. [107] proposed a long-short term memory (LSTM) model to predict traffic speed with rainfall information. Their result showed that the model produced better traffic speed prediction with the rainfall than the prediction without rainfall information. Yang et al. [108] predicted short-term traffic speed

and included external factors such as road information, weather condition, and air quality. Combining these factors led to an improvement in predicting traffic speed. Essien et al. [85] proposed a Long Short-Term Memory Neural Network (LSTM-NN) to predict traffic speed by adding weather data with traffic information on an urban road network. Their experiments indicated the importance of combining the weather data with traffic information in which the model produced improved results instead of using traffic data only. Alammari [109] utilized the availability of different data sources such as traffic information, weather data, traffic jams, traffic incidents, and temporal information to predict traffic flow using several features. The experiments indicated that combining the traffic flow, accidents, weather, time information, and the number of lanes produced the best results comparing to use only traffic flow. Polson and Sokolov [110] introduced a deep learning approach to predict traffic flow under two events: Chicago Bears football game and an extreme snowstorm. Their comparison showed that predicting traffic flow using deep learning under these two events produces better results than a neural network with one hidden layer. Yu et al. [111] predicted traffic flow under extreme conditions such as accidents and peak-hours. Their model utilized LSTM to predict traffic and Mixture Deep LSTM to predict traffic after accidents. The input to their model is the traffic information and accident reports. By combining the accident reports with traffic information, the model provided better performance in forecasting traffic. Moosavi et al. [94] proposed a Deep Accident Prediction (DAP) model to predict traffic accidents with heterogeneous data sources such as weather data, POI, and time-information. The temporal factors, traffic events, and POI have an important contribution for predicting accidents from their experiment.

Motivated by these works, in this chapter, we provide a DF framework on multiple heterogeneous data sources to extract features and provide processed unified

datasets to prepare them for deep learning approaches. In addition, these preprocessed and fused data sources provide traffic databases for analyzing and visualizing traffic data. These datasets will help study and solve ITS-related problems such as traffic flow prediction, incidents location prediction, and travel time estimation. Our DF framework follows the *Data In–Feature Out (DAI-FEO)* model and apply different techniques such as map matching, semantic matching, temporal matching, spatial matching, and data association.

5.3 Datasets descriptions

In this section, we describe the datasets that we used with examples of each dataset.

5.3.1 Traffic Data

We obtained the traffic data from two sources that are the California Department of Transportation (Caltrans) Performance Measurement System (PeMS) [112, 113] and Municipal.systems [114]. The PeMS provides historical and real-time traffic information such as flow, speed, occupancy, and other data using over 39,000 individual detectors (sensors) distributed across the states of California. The Municipal.systems provides several types of data and we utilized one of their datasets, namely the Waze Traffic Jams dataset.

5.3.1.1 PeMS datasets (Traffic flow):

we obtained traffic information of the sensors that are located in *PeMS D7 (LA county)* and *PeMS D4 (Bay area)* for the period from 7/1/2019 to 12/31/2019. Each sensor associated with the sensors type according to their location on the road network,which can be a main-line, fwy-fwy connector, off ramp, and on ramp, see

figure 5.1 as is example. Each record provided by the sensors has aggregated data

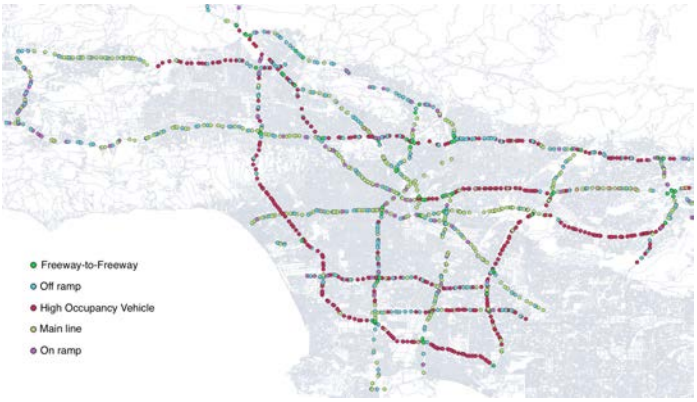


Figure 5.1: LA sensors location with different type

over 5-min periods with the following information [113]:

- *Total Flow*: Sum of flows over the 5-min period across all lanes.
- *Average Occupancy:* The proportion of time in which vehicles are passing over the sensors.
- *Average speed:* Flow-weighted average speed over the 5-min period across all lanes.

Figure 5.2a shows the record information from sensors for one day and figure 5.2b shows the traffic information (flow, speed, and occupancy) from one sensor for two consecutive days.

5.3.1.2 Traffic incidents:

We obtained traffic incidents from California Highway Patrol (CHP) that are stored in the PeMS website [112]. Each traffic incident is associated with *incident id, time, location, duration, type, freeway number, and direction.* We extracted only incidents that are located on the same roads where the sensors are. Table 5.1 sum-

TIMESTAMP	ID	Freeway	Dir	FLOW	OCCUPANCY	SPEED
07/01/2019 00:00:00	715898	5	S	228.0	0.0368	70.2
07/01/2019 00:00:00	715918	5	N	159.0	0.0296	69.4
07/01/2019 00:00:00	715920	5	S	228.0	0.0368	70.1
07/01/2019 00:00:00	715929	5	S	239.0	0.0568	64.2
07/01/2019 00:00:00	715930	5	N	217.0	0.0486	68.0
...	...	...	...	...	...	...
07/01/2019 23:55:00	776690	60	W	198.0	0.0642	59.6
07/01/2019 23:55:00	776708	60	E	134.0	0.0470	58.4
07/01/2019 23:55:00	776739	405	N	224.0	0.0474	64.4
07/01/2019 23:55:00	776751	101	S	112.0	0.0594	61.7
07/01/2019 23:55:00	776752	101	N	144.0	0.0595	66.8

(a) Sample of sensors data for one day

(b) Traffic information (flow, occupancy, speed) for one sensor for two days

Figure 5.2: Traffic information from sensors. a) Raw data from different sensors for one day. b) Plotted traffic information

Table 5.1: Summery of Incidents

Incident type	#D7(LA)	#D4(Bay area)
Traffic Hazard	32020	3902
Traffic Collision	24242	3504
Hit and Run	3314	394
Car Fire	567	94
Traffic break	549	72
Closure of a Road	229	3
Wrong way driver	215	65
Total	61136	8034

marizes the types of incidents that occurred in D4 and D7 and figure 5.3 shows an example of the incidents report.

5.3.1.3 Municipal.systems dataset (Traffic Jams)

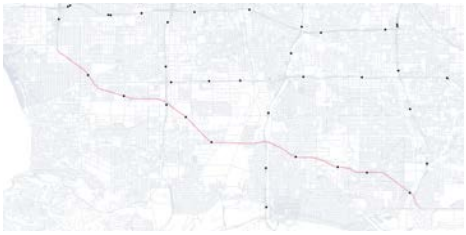
Municipal.systems is a website by Stae Inc. that has different types of datasets for different countries [114]. We obtained the traffic jams data collected by Waze app, a mobile application that allows users to navigate their roads and report different

datetime	Y	X	sens	inc	timestamp	acs_fwy_n	acs_fwy_D	Descriptio	Duration
2019-07-01 04:10:00	34.168194	-118.301596	759747	18836653	07/01/2019 04:08:00	5	S	Traffic Hazard	25.0
2019-07-01 07:30:00	33.995464	-118.145272	716922	18836927	07/01/2019 07:29:00	5	S	Traffic Hazard	1.0
2019-07-01 09:50:00	34.010233	-118.160044	715947	18837190	07/01/2019 09:48:00	5	S	Traffic Hazard	70.0
2019-07-01 16:00:00	33.964275	-118.120211	716922	18837994	07/01/2019 16:01:00	5	S	Traffic Hazard	19.0
2019-07-01 16:15:00	34.065974	-118.216387	716943	18838022	07/01/2019 16:13:00	5	S	Traffic Hazard	84.0
...	...	...	...	...	...	...	...	...	...
2019-10-04 20:25:00	33.987383	-118.392542	774279	19044337	10/04/2019 20:24:00	90	E	Hit and Run	12.0
2019-10-08 23:05:00	33.987383	-118.392542	774279	19052891	10/08/2019 23:05:00	90	E	Hit and Run	65.0
2019-11-06 19:50:00	33.987951	-118.403783	774279	19116887	11/06/2019 19:48:00	90	E	Hit and Run	18.0
2019-09-09 17:40:00	33.987383	-118.392542	774279	18988555	09/09/2019 17:41:00	90	E	Car Fire	65.0
2019-10-07 04:00:00	33.988882	-118.389296	774279	19048481	10/07/2019 03:59:00	90	E	Car Fire	10.0

Figure 5.3: Example of traffic incidents report

id	startedAt	endedAt	street	endStreet	severity	delay	geometry
0 1406773019	2019-07-02 00:36:16	2019-07-02 08:14:00	to I-5 N	I-5 N	0.6	82.0	LINESTRING (-118.271637 34.119415, -118.271632...
1 1441302736	2019-07-02 16:08:25	2019-07-02 23:53:00	I-5 N	to I-5 N	0.2	60.0	LINESTRING (-118.136859 33.98717, -118.13763 3...
2 1429837172	2019-07-02 00:09:49	2019-07-02 08:14:00	I-5 N	I-5 N	0.2	52.0	LINESTRING (-118.268272 34.115245, -118.268855...

(a) Example of three traffic jams report



(b) Pathway of one traffic jam

Figure 5.4: Traffic jams report. a) Traffic jams report. b) Plotted traffic jams on the road network

traffic events such as traffic jams, accidents, road closure, and others [115]. Each report in the traffic jams data is associated with different sets of features such as *the report id, start and end time, start and end street, the severity of traffic jams, and the path of the traffic jams.* Figure 5.4a shows an example of three traffic jams records and figure 5.4b depicts one traffic jams report on the road network that expanded over several sensors. We extracted only traffic jams that are located on the same roads where the sensors are.

5.3.2 Weather Data

We obtained the hourly weather data from the *Dark Sky* website [116]. It provides weather forecasting using APIs. The historical hourly weather data has features that include *temperature, precipitation intensity, precipitation probability, humidity, wind speed, wind gust, cloud cover, and visibility.* The raw weather data

was collected by providing the location (latitude and longitude) of each sensor and the timestamp. We collected the weather data for each sensor location. Figure 5.5 shows an example of the weather report for different locations and timestamps.

datetime	ID	precipIntensity	precipProbability	humidity	windSpeed	windGust	cloudCover	visibility	temperature
2019-07-01 00:00:00	759747	0.0000	0.00	0.56	2.71	3.91	0.00	10.000	67.30
2019-12-29 00:00:00	762353	0.0000	0.00	0.82	3.64	4.37	0.01	10.000	49.95
2019-11-20 10:40:00	716943	0.0384	0.46	0.67	5.28	8.38	0.59	9.236	57.69
2019-08-29 01:20:00	716922	0.0000	0.00	0.87	2.19	3.85	0.07	10.000	69.19

Figure 5.5: Example of weather data

5.3.3 Points of Interest (POI)

We obtained Points of Interest (POI) using OpenStreetMap API [68] in D4 and D7 located within a specific distance (within 200 m) from each sensor. Each POI is associated with *POI's type and location*. We choose the following POI's types: *Hospital, Parking, Fuel, Fast Food Restaurant, Restaurant, school, and university*. Figure 5.6 shows an example of POI's that are near the sensors.

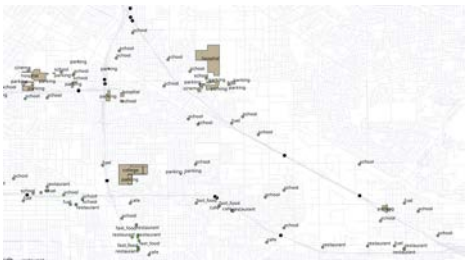


Figure 5.6: Example of point of interest located near the freeways

5.3.4 Temporal data

The temporal data such as an hour of the day, day of the week, the month of the year, and weekend indicator can be generated from each traffic flow record's timestamp.

5.4 Traffic Datasets Fusion Process

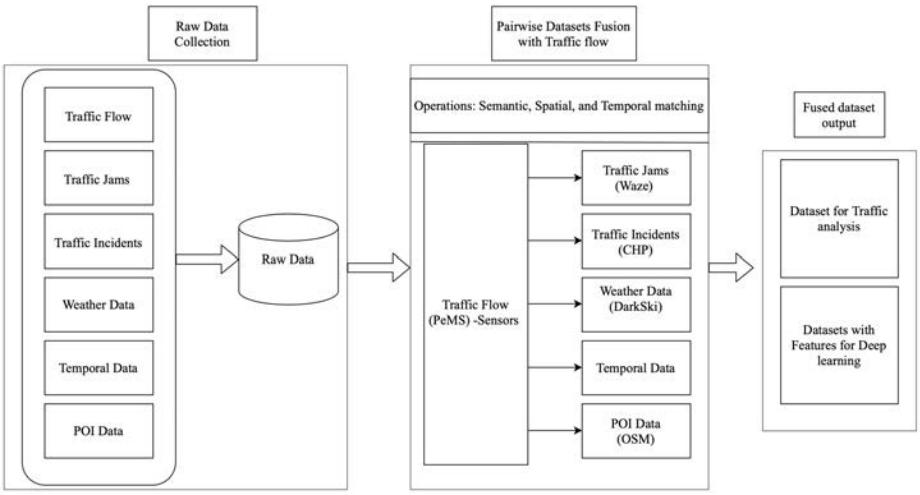


Figure 5.7: Data Fusion Framework for heterogeneous data sources

This section describes the pairwise data fusion process. We fuse each data source individually with traffic flow to integrate and extract features from heterogeneous data sources with details of each step. Figure 5.7 shows the framework for our DF process that receives raw data from different sources and applies the pairwise fusion process to provides the two outputs.

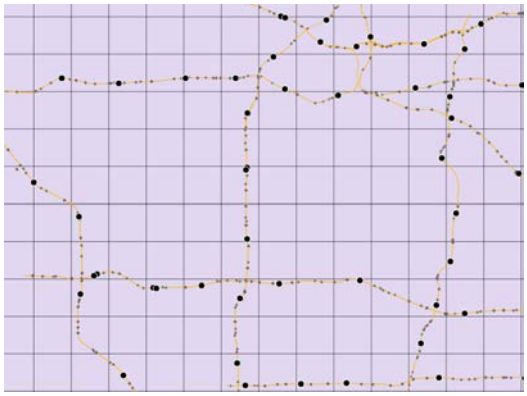
5.4.1 Traffic Data fusion

5.4.1.1 Traffic flow preparation:

The obtained data from *PeMS* has a separate file for each day with the traffic information from all sensors, figure 5.8a, for example. In total, we have 184 files for the period from 7/1/2019 to 12/31/2019 for each district. The number of observed traffic records for D4 is 126891250 and D7 is 100313856. The steps for the traffic

TIMESTAMP	ID	Freeway	Dir	FLOW	OCCUPANCY	SPEED
10/10/2019 00:00:00	400001	101	N	41.0	0.0071	71.1
10/10/2019 00:00:00	400002	101	S	214.0	0.0286	72.6
10/10/2019 00:00:00	400006	880	S	93.0	0.0159	68.4
10/10/2019 00:00:00	400007	101	N	138.0	0.0196	72.5
...	...	...	...	...	...	...
10/10/2019 23:55:00	424105	4	W	13.0	0.0065	66.4
10/10/2019 23:55:00	424106	4	W	11.0	0.0058	67.7
10/10/2019 23:55:00	424110	4	W	13.0	0.0142	64.7
10/10/2019 23:55:00	424111	4	W	9.0	0.0125	64.2
10/10/2019 23:55:00	424420	580	W	158.0	0.0312	65.7

(a) Sample of one day traffic data of sensors in PeMSD4



(b) Example of the remaining sensors (black dots)

Figure 5.8: Traffic flow preparation. a) Sample of one day traffic data. b) Remaining sensors.

flow preparation as fallow:

1. Choose only the sensors with type ML (Mainline). These sensors are located on major freeways.
2. Create a grid with identical cell size (2*2 miles) and apply spatial join between the grid and the sensors. For each cell in the grid, keep only one sensor for each freeway and each direction. This method eliminates possible duplicate reading of sensors located in the same cell and provides good road network coverage. Figure 5.8b shows a grid and the remaining sensors in each cell.

3. Remove the sensors with more than 10% null-values due to the sensors being faulty and use the linear interpolation method to fill missing values.

After this process, the total number of remaining sensors for D4 is 189 sensors and the numbers of observed traffic records are *10015488*, where each sensor has *52992* record points. For D7, the remaining sensors are 378 and the numbers of observed traffic records are *20507904*. The extracted features for each sensor in this step are: *sensor_ID, flow, speed, and average occupancy.*

5.4.1.2 Traffic incidents fusion:

The incident reports, as the traffic flow, are gathered in one file per day. However, the traffic flow information is evenly spaced, where we have a record for each 5-min interval. The incidents report is not evenly spaced and incidents can occur at any time. Therefore, the incidents fusion process requires a temporal aggregation for incidents that occur near each other and within a 5-min window. In addition, each incident report needs to be spatially associated with the nearest sensor on the same road and direction. Using the information associated with each incidents report, the traffic incidents fusion with the traffic flow is as follows:

1. *Semantic matching:* Extract all incidents and sensors for each road and direction, for example, all incidents and sensors that are located on freeway *5* and direction *N*.
2. *Spatial matching:* For each incident, compute the *Euclidean distance* to the nearest sensor and add the *sensor id* to the incident.
3. *Temporal matching:* Round the timestamp of each incident to the nearest 5-min interval. Then, fuse the incident report with the traffic flow using the *sensors ID* and *timestamp*. In case two incidents happened at the same time window (within 5-min) and at the same sensor, we aggregate the number of incidents

and duration. Also, we create a feature for each incident type and duration. Figure 5.9c shows an example of aggregated incidents from figure 5.9a that happened at the same time and location.

Algorithm 5 summarizes the steps for traffic incidents fusion. After this process, the following features will be merged with traffic flow as a vector of [0 and 1], where zero indicates no incident and one indicates there is an incident: *Traffic Collision, Car Fire, Traffic Hazard, Wrong-way driver, Traffic break, Closure of a Road, Hit and Run, total incidents, and duration of each incident.*

datetime	Y	X	sens	inc	timestamp	acs_fwy_n	acs_fwy_D	Descriptio	Duration
2019-10-23 20:55:00	34.027569	-118.211863	716943	19085480	10/23/2019 20:56:00	5	S	Traffic Collision	3.0
2019-10-23 20:55:00	34.029961	-118.218515	716943	19085472	10/23/2019 20:54:00	5	S	Hit and Run	149.0

(a) Example of two incidents report associated with one sensor at the same time

datetime	Y	X	sens	inc	timestamp	Descriptio_Traffic Collision	Traffic Collision_Duration	Descriptio_Hit and Run	Hit and Runs_Duration
2019-10-23 20:55:00	34.027569	-118.211863	716943	19085480	10/23/2019 20:56:00	1	3.0	0	0.0
2019-10-23 20:55:00	34.029961	-118.218515	716943	19085472	10/23/2019 20:54:00	0	0.0	1	149.0

(b) Create a feature for each incident type

datetime	flow	occupaancy	ID	Descriptio_Traffic Collision	Traffic Collision_Duration	Descriptio_Hit and Run	Hit and Runs_Duration	Incident	Incident_Duration
2019-10-23 20:55:00	371.0	0.0749	716943	1	3	1	149	2	152

(c) Adding the incidents information to the traffic flow with aggregated incidents report

Figure 5.9: Associated incidents to sensor. a) Two incidents report. b Create feature for each incident type. c) Fuse the incidents report with the traffic flow.

5.4.1.3 Traffic jams fusion:

The traffic jam reports are similar to the traffic incidents in which we must semantically, spatially, and temporally associate each report with the nearest sensors. However, each traffic jam is represented as a line, which is the part of the road where the traffic jam occurred, while the incident report is represented as a point. Before we spatially match each traffic jam, we create the road network graph. It represents

Algorithm 4 Traffic incidents Fusion

Input: IncidentsReport, TrafficFlow

Output: Incidents Features fused with traffic flow

1: $FwyDir \leftarrow$ List of unique FwyNumber and Direction from TrafficFlow
2: $Sid \leftarrow$ List of unique sensors id from TrafficFlow
3: // Extract Incidents and Sensors in the same Fwy and dir and compute the distance
4: **for** F $\in$ $FwyDir$ **do**
5: *// Semantic matching*
6: $Incidents \leftarrow$ List of all incidents from IncidentsReport that occurred at *FwyDir[F]*
7: $Sensor \leftarrow$ List of all sensors from TrafficFlow located at *FwyDir[F]*
8: **for** I $\in$ $Incidents$ **do**
9: *// Spatial matching*
10: *senosorid* $\leftarrow$ compute the *Euclidean distance* of I to the nearest $Sensor$
11: Add the *senosorid* to incident I in the *IncidentsReport*
12: **end for**
13: **end for**
14: // Fuse the incidents with the traffic flow
15: *// Temporal matching*
16: Round the timestamp of each incident to the nearest 5-min
17: From the type of incident, create feature of each type and add the duration

Algorithm 5 Continue Traffic incidents Fusion

18: **for** $i \in Sid$ **do**

19: // Get list of all incidents with the same sensor id

20: $Incidents[i] \leftarrow$ List of all incidents from IncidentsReport where $senosorid \in Sid[i]$

21: **if** two *Incidents[i]* occurred at same timestamp **then**

22: Aggregate the incidents and duration

23: **end if**

24: // Update the traffic flow

25: $TrafficFlow \leftarrow$ Fused $Incidents[i]$ features with $TrafficFlow$ using *senosorid* and *timestamp*

26: **end for**

each sensor and the previous and following sensors; since the traffic jam may pass over several sensors, as in figure 5.4b. The traffic jams fusion steps are as follows:

1. *Semantic matching:* Extract all traffic jams and sensors for each road and direction.
2. Create the road network graph and add to each sensor the *ids* of the previous and following senors.
3. *Spatial matching:* For each traffic jam report, get the start and endpoint to compute the *Euclidean distance* to the nearest sensor of each point. If the nearest distance of the start and endpoint is associated with one sensor, add the *sensor id* to the traffic jam report. Otherwise, if the nearest sensor to the start point of a traffic jam is different from the endpoint, we check using the road network graph and return all sensors between the traffic jam's start and endpoint. We add ids of these sensors to the traffic jam report.

4. *Temporal matching:* Round the timestamp of each traffic jam report to the nearest 5-min interval and fuse the traffic jam report with the traffic flow using the *traffic jam Id, sensors Id, and the timestamp.*

Algorithm 7 summarizes the traffic jams fusion steps with traffic flow. After this process, the following features merge with the traffic flow:*delay, severity, and duration of traffic jam that we compute using the start and end time.*

5.4.2 Weather Data fusion

The obtained weather data for each sensor's location from the *Dark Sky API* are sampled each hour while the traffic flow is sampled each 5-min. Thus, before we fuse the weather data with the traffic flow, we repeated each record of the weather data (12 times) to match the traffic flow data. Then, the weather data is merged with the traffic flow using the timestamp of each sensor. The weather features are represented as a one-dimensional vector of real numbers, which are *temperature, precipitation intensity, precipitation probability, humidity, wind speed, wind gust, cloud cover, and visibility.*

5.4.3 Point-of-Interest fusion

The obtained POI's are fused with the traffic flow by spatially joining to the nearest sensors. Each POI will be represented as a vector of [0 and 1] within the traffic flow data. For example, if sensor *a* has a *school* nearby, the *school* feature will be one.

Algorithm 6 Traffic Jams Fusion

Input: TrafficJamsReport, TrafficFlow

Output: Traffic Jams Features fused with traffic flow

1: $FwyDir \leftarrow$ List of unique FwyNumber and Direction from TrafficFlow
2: *RoadGraph* $\leftarrow$ Create road network graph where each sensor associated with previous and following sensors
3: $Sid \leftarrow$ List of unique sensors id from TrafficFlow
4: // Extract Traffic Jams and Sensors in the same Fwy and dir and compute the distance
5: **for** F $\in$ $FwyDir$ **do**
6: // *Semantic matching*
7: $TrafficJams \leftarrow$ List of all Traffic Jams from TrafficJamsReport that occurred at *FwyDir[F]*
8: *Sensor* $\leftarrow$ List of all sensors from TrafficFlow located at *FwyDir[F]*
9: **for** I $\in$ $TrafficJams$ **do**
10: // *Spatial matching*
11: Extract the start and endpoint of traffic jams I
12: *startsenosorid* $\leftarrow$ compute the *Euclidean distance* of start point of traffic jams I to the nearest *Sensor*
13: *endpointsenosorid* $\leftarrow$ compute the *Euclidean distance* of the endpoint of traffic jams I to the nearest *Sensor*
14: **if** *startsenosorid* = *endpointsenosorid* **then**
15: Add the *senosorid* to traffic jam I in the *TrafficJamsReport*

Algorithm 7 Continue Traffic Jams Fusion

else

sensorsbetween ← Check *RoadGraph* and return all sensors id between *startsenosorid* and *endpointsenosorid*

Add *sensorsbetween* to the traffic jam I in the *TrafficJamsReport*

end if

end for

end for

// Fuse the traffic jams with the traffic flow

// *Temporal matching*

Round the timestamp of each traffic jam to the nearest 5-min

for $i \in Sid$ **do**

// Get list of all traffic jams with the same sensor id and update the traffic flow

$TrafficFlow$ ← Fused $TrafficJam[i]$ features with $TrafficFlow$ using *senosorid* and *timestamp*

end for

5.4.4 Temporal Data fusion

Using the timestamp of each record in the traffic flow, we generate the following temporal data: *hour of day, day of the week, month of the year, and weekend indicator.* Each feature is represented as one-hot encoded vectors.

5.4.5 Summary of the traffic datasets fusion process

After we fused each dataset with the traffic flow, we have the following features, which are summarized in figure 5.10:

- Traffic features: Sensor_ID, flow, speed, and average occupancy.

- Traffic Incidents features: Traffic Collision, Car Fire, Traffic Hazard, Wrong-way driver, Traffic break, Closure of a Road, Hit and Run, total incidents, and duration of each incident.
- Traffic Jams features: Delay, severity, and duration of traffic jam
- Temporal features: Hour of day, day of the week, month of the year, and weekend indicator
- Weather features: Temperature, precipitation intensity, precipitation probability, humidity, wind speed, wind gust, cloud cover, and visibility
- POI features: Hospital, Parking, Fuel, Fast Food Restaurant, Restaurant, school, and university.

Figure 5.11 shows a sample of the output of the data fusion process for one traffic flow record.

Traffic	Temporal	Weather	Incidents	Traffic Jams	POI
flow	weekend_indi	precipIntensity	Descriptio_TrafficCollision	tj_startedAt	Hospital
occupaancy	day_of_week (7)	precipProbability	TrafficCollision_Duration	tj_id	Parking
speed	hour_of_day (24)	humidity	Descriptio_CarFire	tj_delay	Fuel
ID	month_of_year (7)	windSpeed	CarFire_Duration	tj_speed_jam	Fast Food Restaurant
flow_aft		windGust	Descriptio_ClosureofaRoad	tj_street	Restaurant
speed_aft		cloudCover	ClosureofaRoad_Duration	tj_endStreet	school
occupaancy_aft		visibility	Descriptio_HitandRun	tj_severity	university
flow_bef		temperature	HitandRuns_Duration	tj_duration_m	
speed_bef			Descriptio_TrafficHazard	tj_duration_h	
occupaancy_bef			TrafficHazard_Duration	tj_endedAt	
Fwy			Descriptio_Trafficbreak		
Dir			Trafficbreak_Duration		
			Descriptio_Wrongwaydriver		
			WrongWayDriver_Duration		
			Incident		
			Incident_Duration		

Figure 5.10: Summary of the extracted features by type

datetime	flow	occupaancy	speed	sensor_ID	weekend_indi
11/14/2019 2:40:00 Pm	439	0.0906	63.5	716986	0
day_of_week_0	day_of_week_1	day_of_week_2	day_of_week_3	day_of_week_4	day_of_week_5
0	0	0	1	0	0
day_of_week_6	hour_of_day_0	hour_of_day_1	hour_of_day_2	hour_of_day_...	hour_of_day_14
0	0	0	0	0	1
month_of_year_7	month_of_year_8	month_of_year_	month_of_year_	month_of_year_	month_of_year_12
0	0	0	0	1	0
precipIntensity	precipProbability	humidity	windSpeed	windGust	cloudCover
0	0	0.44	5.93	5.93	0.25
visibility	temperature	tj_delay	tj_severity	tj_duration_m	Incident
10	72.87	544	0.8	646	2
Incident_Duration	School	Fuel	Hospital	University	Resturant
25	0	1	1	0	1

Figure 5.11: Sample record of one timestamp after the DF process where each color represents a set of features

CHAPTER 6

Application for data fusion of traffic dataset

The generated traffic datasets have rich features that can be employed in different ITS applications. In this chapter, we provide two applications that utilize each output of our DF framework.

6.1 Traffic Datasets analysis and visualization

This section utilizes one of the DF framework's output by creating a data cube that provides in-depth analysis of the traffic datasets. Furthermore, we design a visual-interactive Geographical Information Systems (GIS) tool to present the results in different ways such as maps, charts, and tables according to the user requirements. We present some of the analysis results that show the importance of fusing heterogeneous data sources.

6.1.1 Traffic Datasets cube

To explore the traffic dataset, we store the traffic dataset in data cube with dimensions similar to a data cube in a trajectory data warehouse [1], as in figure 6.1. With this method, we can have 3D analysis view to analyze and explore the dataset, as follows:

- Spatial analysis: This dimension offers an analysis related to the locations of sensors, freeways, and POI's. Queries can be answered, such as: *What is the location of sensors with a large number of cars? Which freeway has the least*

traffic? What is the difference between traffic on the same freeway in a different direction? What is the average speed on freeway I-405?

- Temporal analysis: From this dimension, we can explore the data with time information. For example, *Which freeway has the most traffic in the morning? What is the freeways status during weekends in July? Which freeway has the most number of cars during the last week of December? What is the average speed on freeway I-5 during the afternoon period?*
- Spatio-temporal analysis: This dimension offers spatial and temporal analysis for traffic flow, incidents, traffic jams, and weather. Queries can be answered: *What is the total number of incidents during July on freeway I-405? Which sensor has the largest number of traffic jams report during the weekdays in the afternoon? What is the total number of traffic jams reports during a rainy day?*

[18, 1] The data stored in aggregation form and we can apply the Online Analytical Processing (OLAP) operations such as *roll-up*, which is used for data aggregations along with dimensions (e.g., aggregate data from lower dimensional to higher dimensional, for example, from hour to a day to a week to month...). The *drill-down* operation is used to decrease the level of aggregation (e.g., from higher dimensional to lower-dimensional, for example, from the city to freeways to individual sensor location.). The *Slice* and *Dice* operations are used to select part of the data cube. *Pivot* operation is used to change the view of the data cube. Besides the OLAP, data mining techniques can be applied to extract hidden information and discover useful knowledge. For example: discovering the different patterns of traffic flow with incidents in different freeways, detecting the traffic jams between different hours of a day and days of the week, discovering the traffic pattern during days with different weather patterns (low visibility, rainy days, clear days,..), comparing the traffic flow during normal and rush hours.

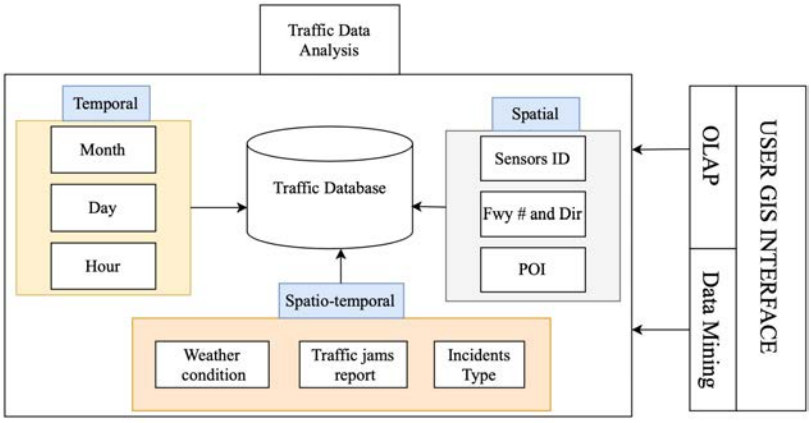


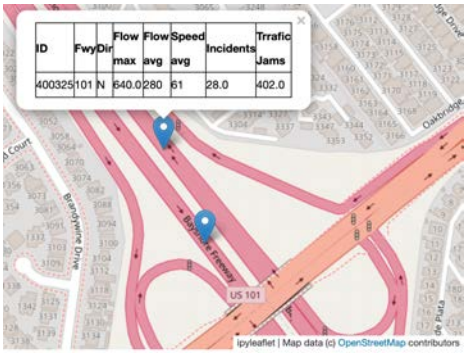
Figure 6.1: Traffic dataset analysis

6.1.2 Traffic Datasets Visualization

This section presents our visual-interactive GIS tool to show the results on the traffic dataset cube. It is designed using Python [117] as a programming language with ipywidgets, ipyleaflet, Pandas [118], folium libraries. The interface allows users to see the results on maps, charts, and tables. The following are examples of different type of analysis that can be done on the traffic datasets:

Visualization of the sensors distributed over the road network: The interface allow users to see locations of sensors over a map with summary information of each senor such as the flow, freeway number and direction, average speed, total incidents, and total traffic jams report. Figure 6.2a shows an example of the sensors in D4 and figure 6.2b shows details of one sensor.

(a) Senors locations in D4



(b) One sensor with information

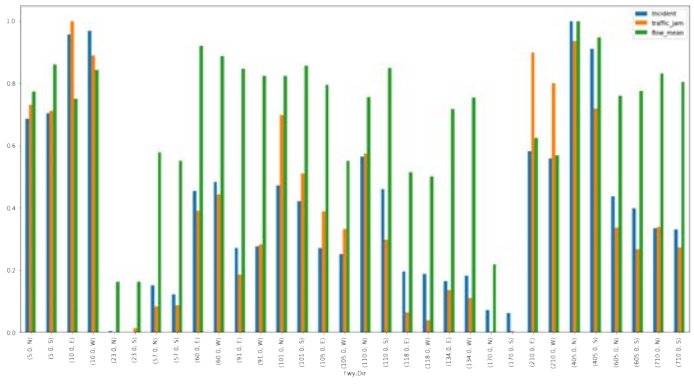
Figure 6.2: Sensors locations over the map in D4

Analysis of different freeways: Since the sensors located on freeways, we aggregated each freeway's sensors to provide an overview of the traffic on each freeway. To provide the results, we used two OLAP operations: *roll-up* and *drill-down*. Figure 6.3 shows how the user can retrieve the information by choosing the location (Bay area *D4* or LA *D7*), freeway number, month, day, then an hour. Then, the result will be presented in a table and chart. Figure 6.4 shows the user interface to choose and the presented result for all freeways in *D7* over the six months. This figure shows that freeways with high traffic flow volumes have large numbers of traffic incidents and traffic jams. Figure 6.5 shows the result of a specific freeway and a specific month.

(a) User options to choose

Fwy	Dir	Incident	traffic_jam	flow_max	flow_mean
5.0	N	3144	21080	944.0	332.773852
	S	3224	20531	1062.0	360.737660
10.0	E	4336	28587	913.0	325.199951
	W	4385	25497	954.0	355.218664
23.0	N	150	740	690.0	136.021039
	S	127	1129	670.0	135.600657
57.0	N	790	3088	755.0	270.049175
	S	664	3208	653.0	261.313767

(b) Sample of the result in a table



(c) Chart result

Figure 6.4: Result of traffic flow,incidents, and traffic jams report in different freeways in D7 over the six months

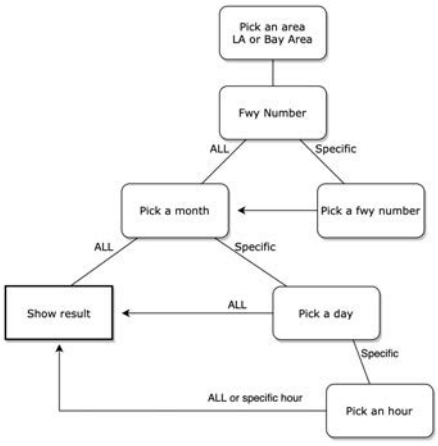
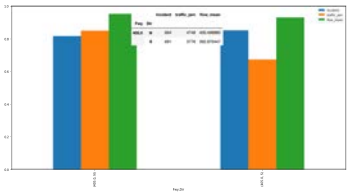


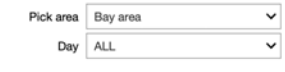
Figure 6.3: Ways to retrieve the data from the traffic database

(a) User options to choose



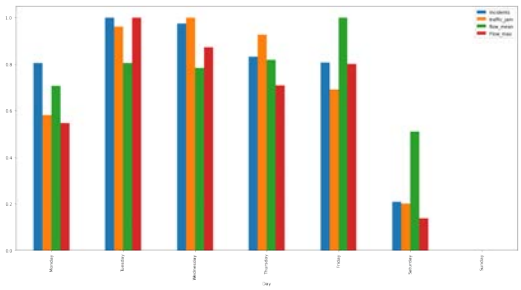
(b) Table and chart result

Figure 6.5: Result of traffic flow,incidents, and traffic jams report for a specific freeway and month



Day	Flow_max	flow_mean	traffic_jam	Incidents
Monday	871.0	237.979865	7339	1241
Tuesday	910.0	243.541154	11442	1371
Wednesday	899.0	242.358612	11859	1355
Thursday	885.0	244.315873	11069	1259
Friday	893.0	254.506942	8539	1243
Saturday	836.0	226.884321	3266	845
Sunday	824.0	198.062284	1083	705

(a) User options to choose and the result in a table



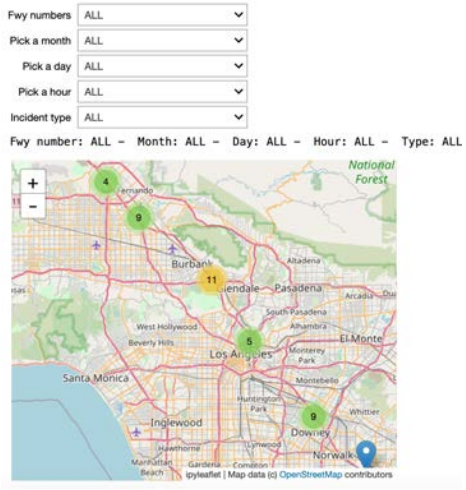
(b) The result in a chart.

Figure 6.6: Result of traffic flow,incidents, and traffic jams report for days of the week of the six month

Analysis of different days of the week: To study the traffic patterns during the weekdays and weekends, we compare the traffic flow with incidents and traffic jams reports during the weekdays and weekends, as in figure 6.6. The user also can have the option to choose a specific day of the week.

Incidents analysis: To analyze the different types of incidents, our interface will allow users to see the aggregated result over the six months or detailed. The result will be on a map with each incident information, as in figure 6.7.

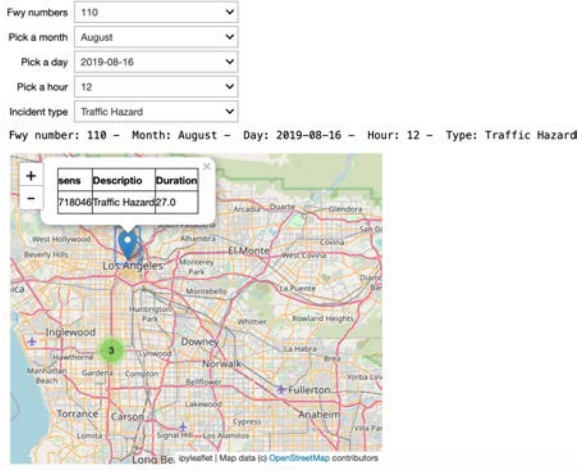
Another type of incident analysis that we added is the *heatmap visualization* of incidents. We aggregated the incidents by an hour of a day such as figure 6.8 that shows the result at different times. From this figure, we can see that most of the incidents happened during rush hours.



(a) User options to choose and result on a map

timestamp	Y	X	sens	inc	acs_fwy_n	acs_fwy_D	Descriptio	Duration
07/01/2019 04:08:00	34.168194	-118.301596	759747	18836653	5	S	Traffic Hazard	25.0
07/01/2019 07:29:00	33.995464	-118.145272	716922	18836927	5	S	Traffic Hazard	1.0
07/01/2019 09:48:00	34.010233	-118.160044	715947	18837190	5	S	Traffic Hazard	70.0
07/01/2019 16:01:00	33.964275	-118.120211	716922	18837994	5	S	Traffic Hazard	19.0
07/01/2019 16:13:00	34.065974	-118.216387	716943	18838022	5	S	Traffic Hazard	84.0

(b) Table result of all freeways

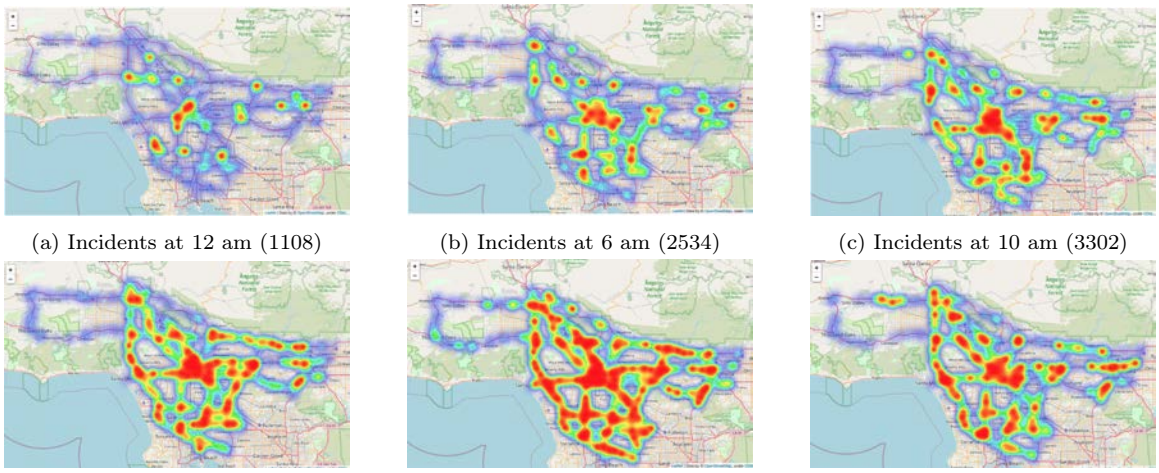


(c) Result of specific incidents on a specific freeways

timestamp	Y	X	sens	inc	acs_fwy_n	acs_fwy_D	Descriptio	Duration
08/16/2019 12:06:00	33.928497	-118.280409	716495	18936060	110	N	Traffic Hazard	12.0
08/16/2019 12:55:00	34.059258	-118.252587	718046	18936190	110	N	Traffic Hazard	27.0
08/16/2019 12:11:00	33.928931	-118.281073	716494	18936074	110	S	Traffic Hazard	7.0
08/16/2019 12:13:00	33.928931	-118.281073	716494	18936072	110	S	Traffic Hazard	3.0

(d) Table result of a specific freeway

Figure 6.7: Result of incidents report on all freeway and on a specific freeway and time



(a) Incidents at 12 am (1108) (b) Incidents at 6 am (2534) (c) Incidents at 10 am (3302)

(d) Incidents at 2 pm (4077) (e) Incidents at 5 pm (4174) (f) Incidents at 8 pm (2097)

Figure 6.8: Heat map visualization of incidents that happened in LA during the six month aggregated by hour of a day with total incidents in each hour

Analysis of Traffic Jams: We analyze the traffic jams by aggregated all traffic jam reports for each freeway and direction. Figure 6.9 shows the heatmap for traffic jams that were occurred in D7 over the six months with the total of traffic jams for each freeway. Figures 6.8 and 6.9 indicate that freeways with large number of incidents have large numbers of traffic jams report.

(a) Heatmap shows the total of traffic jams reports in LA over the six months

(b) Heatmap of a specific freeway with total number of traffic jams report

Figure 6.9: Heat map visualization of traffic jams reports in LA over the six months for all freeways and a specific freeway

Traffic dataset as recommendation systems: Due to the rich features in the traffic dataset, we can use it for different purposes. For example, we can analyze the locations with a large number of cars and find the best location to place a billboard (Ads), as in figure 6.10.

(a) Sensors with high traffic flow at 12 pm

(b) Sensors with high traffic flow at 4 pm

Figure 6.10: Best location for placing billboard according the number of cars for each sensors

6.2 Traffic flow forecasting using deep learning

One of the essential parts of ITS applications is traffic flow forecasting, where it estimates numbers of vehicles on each roads for different time-intervals in the future [119]. This task is important for several ITS services, such as flow control systems, route planning, navigation systems, and so on. There are some factors that can impact the traffic conditions [120], such as: (1) *Spatial dependencies on a directed road network*. Sensors on the same road and direction are influenced by each other more than sensors in the opposite direction. Figure 6.11 shows three sensors a,b,c, where sensors a,b are in the same direction and sensor c is in the opposite direction. Even if c is closer in Euclidean distance to a, it may not affect the traffic flow in a since they are far away in the directed network distance. (2) *Multiple temporal dependencies*, as in figures 5.2b and 6.6b, where the traffic flow varies by hour of

the day, day of the week, and during incidents and traffic jams. (3) *External factors* such as incidents, weather conditions, road closings, and social events. These factors may need to be considered when designing a model to predict traffic flow. Recently, deep learning approaches have been applied to traffic flow forecasting and achieved promising results [88, 89, 90, 91, 92, 93, 94, 95].

In this section, the goal is to examine the second output of our DF framework, which is Traffic datasets with features for deep learning approaches, and to check if incorporating traffic features with external factors will enhance the traffic flow forecasting results or not. Our experiments focus on short-term traffic flow (e.g., next hour) with multiple features.

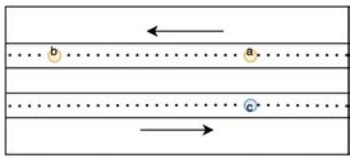


Figure 6.11: Spatial correlation between sensors on same and opposite direction

6.2.1 Deep learning approaches:

Among several deep learning approaches, in our experiments, we chose the recurrent neural network (RNN) specifically:

- Long Short-Term Memory networks (LSTM) [121]: The goal of LSTM NN is to deal with long-term dependencies, which is required for traffic flow forecasting, and to handle the vanishing gradient problem [122]. LSTM is similar to RNN with different in the hidden layer that is treated as memory blocks. [122] Each memory block contains one or more self-connected memory cells and three multiplicative called: the input, output, and forget gates that provide continuous

write, read and reset operations on the cells. These gates allow LSTM to store and access information for long periods, which reduces the vanishing gradient problem, and to keep relevant information for making predictions.

- Gated Recurrent Unit (GRU) [123]: GRU is similar to LSTM with less computing complexity and faster to train. [97] Each GRU unit has an update gate to decide which information to be kept and the rest gate to decide which information will be added to the previous state.
- Bidirectional (BDLSTM): [124] BDLSTM processes sequence data in both forward and backward directions with two separate hidden layers and connect them to the same output layer. The difference between LSTM and BDLSTM is that LSTM stores information from the past only to make prediction while BDLSTM stores information from the past and future.

6.2.2 Evaluation metrics:

We use two well-known evaluation metrics for traffic flow forecasting [109]:

- Mean absolute error (MAE): This metric measures the average absolute difference between predicted and actual values.

$$MAE = \frac{\sum_{t=1}^{n} |A_t - P_t|}{n} \tag{6.1}$$

Where n is the number of data points, A_t is the actual value, P_t is the forecast value at time t.

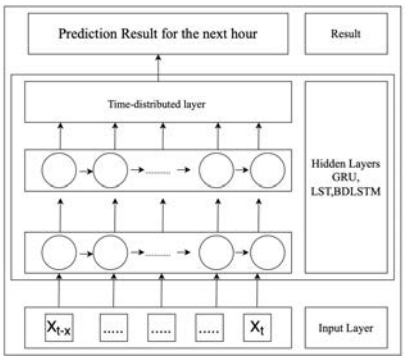
- Root mean square error (RMSE): This metric measures the square root of the mean of the square difference between actual and predicted values.

$$RMSE = \sqrt{\frac{\sum_{t=1}^{n} (A_t - P_t)^2}{n}} \tag{6.2}$$

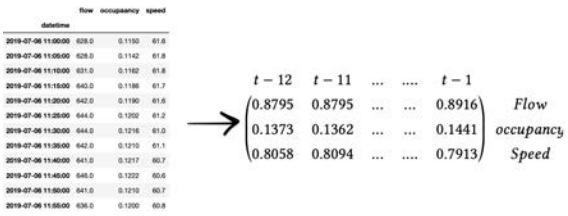
Where n is the number of data points, A_t is the actual value, P_t is the forecast value at time t.

6.2.3 Experimental setting:

Figure 6.12a shows the architectures for the deep learning approaches. We design a deep LSTM, deep GRU, and deep BDLSTM neural network where each model has stacked hidden layers. Each model's input is the previous hour (12-time steps - since the traffic data divide per 5-min interval). Figure 6.12b shows a sample input at different time-stamp with three traffic features only. The hidden layers' output is connected with a fully time-distributed layer to forecast the traffic flow for the next hour. Each model was designed with two hidden layers (*300,100* units) and was trained for around *100* epochs, with Adam optimizer [125]. The learning rate is *0.00001* and the activation function is *ReLu*. The batch size is *128*. The mentioned setting has been chosen using a grid search method since they provide the best result. We run other experiments with adding a dropout layer but the performance was not improved.



(a) Architectures for the deep learning approaches



(b) Input sample to the deep learning models

Figure 6.12: Designed models and sample input

From the traffic datasets for *PeMSD4* and *PeMSD7*, we select the first two months as a training set from 7/1/2019 to 8/31/2019. For the validation set, we select from 9/1/2019 to 9/20/2019, and for the testing set, we select from 9/21/2019

to 9/27/2019. The number of sensors to forecast traffic flow for *PeMSD4* are *189* and for *PeMSD7* are *378*. All the models are implemented using Kears [126]. We model and train each sensor with its features independently. Before providing the data to each model, the traffic features (flow, speed, occupancy), incidents duration, traffic jams duration, weather features are normalized between 0 and 1 using the equation in 6.3. For the temporal features, we use a one-hot encoding. For the evaluation, we re-scale the traffic flow forecast value to the normal values and we compare with the actual value.

$$N = \frac{x\text{- } min(x)}{max(x)\text{-}min(x)} \tag{6.3}$$

6.2.4 Experimental Results

Our experiments evaluated the traffic flow forecasting performance using traffic features without and with extra features. The features that we examine are: *Traffic features* (flow, speed, and average occupancy), *Temporal features* (hour of the day, day of the week, and weekend indicator), *Incidents features* (Incident indicator and Incident Duration), *Weather features* (temperatures, wind speed, wind gust, and visibility), and *Traffic jams features* (severity, delay, speed during a traffic jam, duration). As a baseline, we used the traditional method called *Historical Average (HA)* [127], where the average value of the last hour (12-time steps) is the prediction for the next hour and traffic features for each model.

Table 6.1 shows the average results of the traffic flow forecasting performance of the next hour for two traffic datasets on different models. The results indicate that better performance was achieved when extra features were included. The deep learning approaches achieve better results compared to HA. The BDLSTM achieves the best performance result comparing to LSTM and GRU. The traffic features with

temporal and incidents features provide the best result comparing to other features. Figure 6.13 shows samples of the forecast result for one sensor in $D7$ of different models for traffic features only and with adding extra features. From these figures, we can see that using extra features with traffic features provide better prediction result. The overall results indicate our data fusion techniques' advantages, even with a simple design of a deep neural network. In our experiments, we train each sensor individually and only consider the features for each location. This method's advantage is that we can analyze all sensors and group sensors with similar behavior. Then, we can train one sensor of each group and use transfer learning to other sensors, which would reduce the training time for the whole network. Additionally, ways may improve the result of the deep learning approaches when using extra features such as:

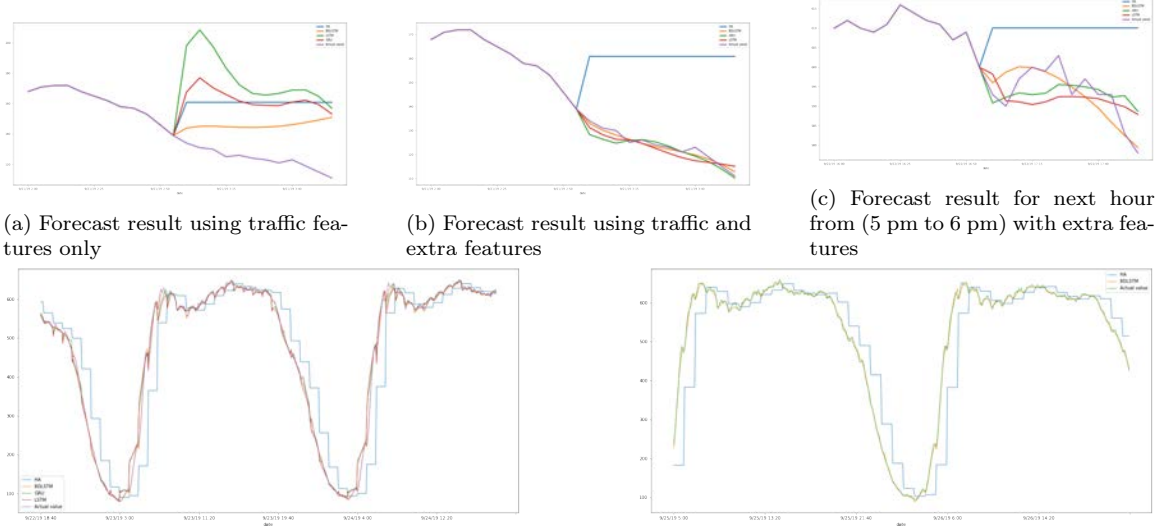
- Considering the spatial correlation between sensors on the same road and direction by utilizing the previous and follower senors' information. In this way, we can model the deep learning by creating a graph for the road network and consider the spatio-temporal dependencies between sensors.
- Designing a hybrid deep learning model with several pipelines to learn different traffic patterns. For example, we can split the data according to traffic patterns such as normal hours, rush hours, incidents duration, traffic jams duration, rainy days, and train the deep learning model according to each pattern.

6.3 Conclusion

In chapter 5 and 6, we utilized the availability of heterogeneous data sources for ITS applications and provided a data fusion framework with semantic matching, map matching, spatial matching, temporal matching, and data association techniques. The DF techniques aimed to integrate and extract features from multiple data sources to be ready for deep learning training approaches. In addition, it generated preprocessed

		PeMSD7		PeMSD4	
Model	Features	MAE	RMSE	MAE	RMSE
HA	Traffic	48.797	68.636	44.114	61.680
LSTM	Traffic	47.775	66.926	44.640	62.250
	Traffic+Temporal(H)	30.200	42.938	28.041	40.694
	Traffic+Temporal+Incidents	**26.880**	**38.043**	**25.246**	**37.045**
	Traffic+Temporal+Weather	27.189	38.394	25.362	37.150
	Traffic+Temporal+Traffic Jams	26.921	38.067	25.241	37.046
	Traffic+Temporal+Incidents+T Jams+Weather	27.186	38.367	25.391	37.173
GRU	Traffic	44.725	63.653	42.174	59.997
	Traffic+Temporal(H)	28.732	40.649	26.884	39.266
	Traffic+Temporal+Incidents	**26.224**	**37.165**	**24.775**	**36.540**
	Traffic+Temporal+Weather	26.593	37.542	24.883	36.617
	Traffic+Temporal+Traffic Jams	26.258	37.189	24.825	36.581
	Traffic+Temporal+Incidents+T Jams+Weather	26.600	37.557	24.871	36.588
BDLSTM	Traffic	29.954	41.835	28.999	41.871
	Traffic+Temporal(H)	24.262	34.520	23.022	34.205
	Traffic+Temporal+Incidents	**23.576**	**33.524**	**22.370**	**33.375**
	Traffic+Temporal+Weather	24.103	34.224	22.531	33.511
	Traffic+Temporal+Traffic Jams	23.638	33.582	22.408	33.395
	Traffic+Temporal+Incidents+T Jams+Weather	24.158	34.267	22.570	33.556

Table 6.1: Average performance comparison of different deep learning approaches with different features on PeMSD7 and PeMSD4.



(a) Forecast result using traffic features only

(b) Forecast result using traffic and extra features

(c) Forecast result for next hour from (5 pm to 6 pm) with extra features

(d) Forecast result for two days for all models with extra features

(e) Forecast result for two days for BDLSTM with extra features

Figure 6.13: Comparison of traffic flow forecasting between actual and predicted values where using traffic only features vs. each model with extra features.

real-world traffic datasets that are publicly available. These datasets had several features as traffic flow, average speed, vehicle occupancy, weather conditions, incident information, congestion reports, point of interest locations, and temporal features. ITS applications such as traffic analysis and visualization, traffic forecast prediction, incident prediction, and travel time estimation can use these datasets.

Furthermore, we provided two applications that utilize each output of our DF framework. The first application is Traffic datasets analysis and visualization, where we built a data cube to provide in-depth analysis of the dataset and designed a visual-interactive GIS tool. The second application is Traffic flow forecasting using deep learning, where we performed a comprehensive study on how different features can improve the traffic flow prediction models. The results of these models showed that deep learning approaches achieved better results when extra features are considered. These results can encourage researchers to build and design more complex deep learning models to utilize the datasets' different features.

CHAPTER 7

CONCLUSION

7.1 Summary of Contributions

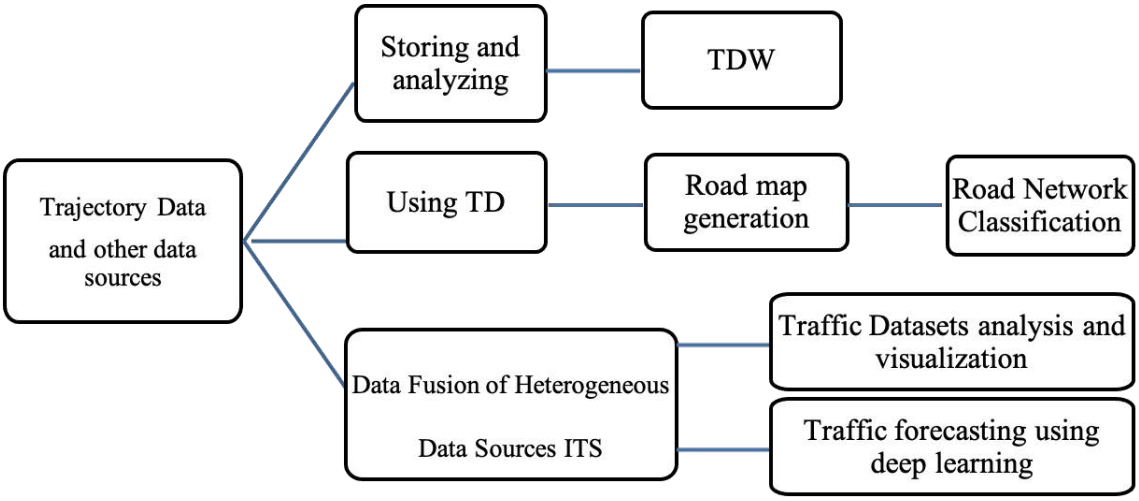
This dissertation utilized the availability of different types of spatio-temporal such as GPS data, road network sensors, incidents reports, traffic jams report, weather information, points of interest locations, and temporal information and provided methods and applications to deal with them. In chapter 2, we presented a framework called Trajectory Data Warehouse (TDW) in which we reviewed existing studies on storing, managing, and analyzing TD using data warehouse technologies. Furthermore, we provided the requirements for building the TDW with different applications using the TDW. After that, in chapter 3, we proposed a novel method to generate road maps using GPS trajectories. In chapter 4, we provided a machine learning framework for road network classification that aims to distinguish the roads based on their types. Then, in the last two chapters 5 and 6, we presented our data fusion framework with different techniques for ITS applications. Then, we introduced two applications to show the importance of the DF techniques. Figure 7.1 summarizes our contribution.

7.2 Future work

In our future work, we intend to:

- Study the impact of incidents and traffic jams on traffic speed for longer time forecasting (e.g., next three hours after incidents or traffic jams occurred).

Figure 7.1: Summary of contribution



- Study the role of extra features on traffic forecasting for different time predictions (e.g., next 15,30,45 minutes using the previous two or three hours instead of one hour only).
- Build a model that can predict traffic flow, incidents locations, and traffic jams simultaneously since this information is already included in the traffic datasets.
- DF techniques and steps can be reevaluated and changed according to the deep learning models' results. For example, when we choose the sensor's location, we picked a sensor every 2 miles. Finding the optimal distance to keep sensors can be reevaluated. Also, we added all incidents and traffic jams to the deep learning models. An analysis step can be added to pick the types of incidents and traffic jams with the most impact on traffic flow before training the deep learning models.
- Enhance the traffic dataset with other information such as social events, holidays, and road work zones.

- Examine different data fusion levels such as *Feature In–Feature Out* or *Feature In–Decision Out* and find out the best level that produces the best performance accuracy result.

Bibliography

[1] T. Alsahfi, M. Almotairi, and R. Elmasri, "A survey on trajectory data warehouse," Spatial Information Research, vol. 28, no. 1, pp. 53–66, 2020.

[2] T. Alsahfi, M. Almotairi, R. Elmasri, and B. Alshemaimri, "Road map generation and feature extraction from gps trajectories data," in Proceedings of the 12th ACM SIGSPATIAL International Workshop on Computational Transportation Science, ser. IWCTS'19. New York, NY, USA: Association for Computing Machinery, 2019. [Online]. Available: https://doi.org/10.1145/3357000.3366140

[3] L. Leonardi, S. Orlando, A. Raffaetà, A. Roncato, C. Silvestri, G. Andrienko, and N. Andrienko, "A general framework for trajectory data warehousing and visual olap," GeoInformatica, vol. 18, no. 2, pp. 273–312, 2014.

[4] G. Marketos, E. Frentzos, I. Ntoutsi, N. Pelekis, A. Raffaetà, and Y. Theodoridis, "Building real-world trajectory warehouses," in Proceedings of the Seventh ACM International Workshop on Data Engineering for Wireless and Mobile Access, ser. MobiDE '08. New York, NY, USA: ACM, 2008, pp. 8–15.

[5] L. Leonardi, G. Marketos, E. Frentzos, N. Giatrakos, S. Orlando, N. Pelekis, A. Raffaetà, A. Roncato, C. Silvestri, and Y. Theodoridis, "T-warehouse: Visual olap analysis on trajectory data," in 2010 IEEE 26th International Conference on Data Engineering (ICDE 2010), March 2010, pp. 1141–1144.

[6] G. Marketos and Y. Theodoridis, "Mobility data warehousing and mining," in Proceedings the 35th International Conference on Very Large Data Bases PhD Workshop, 2009.

[7] J. Biagioni and J. Eriksson, "Map inference in the face of noise and disparity," in Proceedings of the 20th International Conference on Advances in Geographic Information Systems, ser. SIGSPATIAL '12. New York, NY, USA: ACM, 2012, pp. 79–88.

[8] Douglas, D. H, Peucker, and T. K, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," Cartographica: the international journal for geographic information and geovisualization, vol. 10, no. 2, pp. 112–122, 1973.

[9] J. Macedo, C. Vangenot, W. Othman, N. Pelekis, E. Frentzos, B. Kuijpers, I. Ntoutsi, S. Spaccapietra, and Y. Theodoridis, Trajectory Data Models. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 123–150.

[10] S. Spaccapietra, C. Parent, M. L. Damiani, J. A. de Macedo, F. Porto, and C. Vangenot, "A conceptual view on trajectories," Data Knowl. Eng., vol. 65, no. 1, pp. 126–146, Apr. 2008.

[11] Z. Feng and Y. Zhu, "A survey on trajectory data mining: Techniques and applications," IEEE Access, vol. 4, pp. 2056–2067, 2016.

[12] C. Parent, S. Spaccapietra, C. Renso, G. Andrienko, N. Andrienko, V. Bogorny, M. L. Damiani, A. Gkoulalas-Divanis, J. Macedo, N. Pelekis, Y. Theodoridis, and Z. Yan, "Semantic trajectories modeling and analysis," ACM Comput. Surv., vol. 45, no. 4, pp. 42:1–42:32, Aug. 2013.

[13] F. M. Nardini, S. Orlando, R. Perego, A. Raffaetà, C. Renso, and C. Silvestri, Analysing Trajectories of Mobile Users: From Data Warehouses to Recommender Systems. Cham: Springer International Publishing, 2018, pp. 407–421.

[14] R. H. Güting and M. Schneider, Moving objects databases. Elsevier, 2005.

[15] C. Düntgen, T. Behr, and R. H. Güting, "Berlinmod: a benchmark for moving object databases," The VLDB Journal, vol. 18, no. 6, p. 1335, Apr 2009.

[16] W. H. Inmon, Building the Data Warehouse, 3rd ed. New York, NY, USA: John Wiley & Sons, Inc., 2002.

[17] I. S. Mumick, D. Quass, and B. S. Mumick, "Maintenance of data cubes and summary tables in a warehouse," SIGMOD Rec., vol. 26, no. 2, pp. 100–111, Jun. 1997.

[18] S. Chaudhuri and U. Dayal, "An overview of data warehousing and olap technology," SIGMOD Rec., vol. 26, no. 1, pp. 65–74, Mar. 1997.

[19] A. Vaismane and E. Zimányi, Data Warehouses: Next Challenges. Springer Berlin Heidelberg, 2012, pp. 1–26.

[20] J. Han, N. Stefanovic, and K. Koperski, "Selective materialization: An efficient method for spatial data cube construction," in Proceedings of the Second Pacific-Asia Conference on Research and Development in Knowledge Discovery and Data Mining, ser. PAKDD '98. London, UK, UK: Springer-Verlag, 1998, pp. 144–158.

[21] D. Papadias, P. Kalnis, J. Zhang, and Y. Tao, Efficient OLAP Operations in Spatial Data Warehouses. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 443–459.

[22] F. Braz, S. Orlando, R. Orsini, A. Raffaeta, A. Roncato, and C. Silvestri, "Approximate aggregations in trajectory data warehouses," in IEEE 23rd International Conference on Data Engineering Workshop, April 2007, pp. 536–545.

[23] N. Pelekis, A. Raffaetà, M. L. Damiani, C. Vangenot, G. Marketos, E. Frentzos, I. Ntoutsi, and Y. Theodoridis, Towards Trajectory Data Warehouses. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 189–211.

[24] R. Wagner, J. A. F. de Macedo, A. Raffaetà, C. Renso, A. Roncato, and R. Trasarti, Mob-Warehouse: A Semantic Approach for Mobility Analysis with a Trajectory Data Warehouse. Cham: Springer International Publishing, 2014, pp. 127–136.

[25] N. Cho and Y. Kang, "Space-time density of field trip trajectory: exploring spatio-temporal patterns in movement data," Spatial Information Research, vol. 25, no. 1, pp. 141–150, Feb 2017.

[26] L. Guo, G. Huang, X. Gao, J. He, B. Wu, and H. Guo, "Dostra: discovering common behaviors of objects using the duration of staying on each location of trajectories," in Workshops at the Twenty-Ninth AAAI Conference on Artificial Intelligence, 2015.

[27] S. Orlando, R. Orsini, A. Raffaeta, and A. Roncato, "Trajectory data warehouses: Design and implementation issues," Journal of Computing Science and Engineering, vol. 1, no. 2, pp. 211–232, 2007.

[28] F. J. Braz, "Trajectory data warehouses: Proposal of design and application to exploit data." in IX Brazilian Symposium on Geoinformatics, 2007, pp. 61–72.

[29] Z. Yan, D. Chakraborty, C. Parent, S. Spaccapietra, and K. Aberer, "Semantic trajectories: Mobility data computation and annotation," ACM Trans. Intell. Syst. Technol., vol. 4, no. 3, pp. 49:1–49:38, Jul. 2013.

[30] C. Zhou, D. Frankowski, P. Ludford, S. Shekhar, and L. Terveen, "Discovering personally meaningful places: An interactive clustering approach," ACM Trans. Inf. Syst., vol. 25, no. 3, Jul. 2007.

[31] A. T. Palma, V. Bogorny, B. Kuijpers, and L. O. Alvares, "A clustering-based approach for discovering interesting places in trajectories," in Proceedings of the 2008 ACM Symposium on Applied Computing, ser. SAC '08. New York, NY, USA: ACM, 2008, pp. 863–868.

[32] S. Campora, J. A. F. de Macedo, and L. Spinsanti, "St-toolkit: A framework for trajectory data warehousing," in AGILE Conf. Lecture Notes in Geoinformation and Cartography, 2011.

[33] N. Arfaoui and J. Akaichi, "Modeling herd trajectory data warehouse," International Journal of Engineering Trends and Technology, vol. 1, pp. 1–9, 2011.

[34] L. Leonardi, S. Orlando, A. Raffaetà, A. Roncato, and C. Silvestri, "Frequent spatio-temporal patterns in trajectory data warehouses," in Proceedings of the 2009 ACM Symposium on Applied Computing, ser. SAC '09. New York, NY, USA: ACM, 2009, pp. 1433–1440.

[35] L. Wang, Z. Yu, D. Yang, H. Ma, and H. Sheng, "Efficiently targeted billboard advertising using crowdsensing vehicle trajectory data," IEEE Transactions on Industrial Informatics, pp. 1–1, 2019.

[36] S. Karagiorgou and D. Pfoser, "On vehicle tracking data-based road network generation," in Proceedings of the 20th International Conference on Advances in Geographic Information Systems, ser. SIGSPATIAL '12. New York, NY, USA: ACM, 2012, pp. 89–98.

[37] L. Gong, H. Sato, T. Yamamoto, T. Miwa, and T. Morikawa, "Identification of activity stop locations in gps trajectories by density-based clustering method combined with support vector machines," Journal of Modern Transportation, vol. 23, no. 3, pp. 202–213, 2015.

[38] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise," in Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, ser. KDD'96. AAAI Press, 1996, pp. 226–231.

[39] G. Huang, J. He, W. Zhou, G.-L. Huang, L. Guo, X. Zhou, and F. Tang, "Discovery of stop regions for understanding repeat travel behaviors of moving objects," Journal of Computer and System Sciences, vol. 82, no. 4, pp. 582 – 593, 2016, trajectory-based Behaviour Analytics.

[40] D. Wang, Q. Liu, Z. Xiao, J. Chen, Y. Huang, and W. Chen, "Understanding travel behavior of private cars via trajectory big data analysis in urban environments," in 2017 IEEE 15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing,

3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress, Nov 2017, pp. 917–924.

[41] L. Zhao and G. Shi, "A trajectory clustering method based on douglas-peucker compression and density for marine traffic pattern recognition," Ocean Engineering, vol. 172, pp. 456 – 467, 2019.

[42] Y. Han, R. Tse, and M. Campbell, "Pedestrian motion model using non-parametric trajectory clustering and discrete transition points," IEEE Robotics and Automation Letters, pp. 1–1, 2019.

[43] J. Gray, S. Chaudhuri, A. Bosworth, A. Layman, D. Reichart, M. Venkatrao, F. Pellow, and H. Pirahesh, "Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals," Data Mining and Knowledge Discovery, vol. 1, no. 1, pp. 29–53, Mar 1997.

[44] Y. Tao, G. Kollios, J. Considine, F. Li, and D. Papadias, "Spatio-temporal aggregation using sketches," in Proceedings. 20th International Conference on Data Engineering, April 2004, pp. 214–225.

[45] A. M. Hendawi and M. F. Mokbel, "Predictive spatio-temporal queries: A comprehensive survey and future directions," in Proceedings of the First ACM SIGSPATIAL International Workshop on Mobile Geographic Information Systems, ser. MobiGIS '12. New York, NY, USA: ACM, 2012, pp. 97–104.

[46] S. Lee, J. Lim, J. Park, and K. Kim, "Next place prediction based on spatiotemporal pattern mining of mobile device logs," Sensors, vol. 16, no. 2, 2016.

[47] W. Yang and T. Ai, "Poi information enhancement using crowdsourcing vehicle trace data and social media data: A case study of gas station," ISPRS International Journal of Geo-Information, vol. 7, no. 5, 2018.

[48] H. Li, L. Kulik, and K. Ramamohanarao, "Automatic generation and validation of road maps from gps trajectory data sets," in Proceedings of the 25th ACM International on Conference on Information and Knowledge Management, ser. CIKM '16. New York, NY, USA: ACM, 2016, pp. 1523–1532.

[49] C. Chen, C. Lu, Q. Huang, Q. Yang, D. Gunopulos, and L. Guibas, "City-scale map creation and updating using gps collections," in Proceedings of the 22Nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ser. KDD '16. New York, NY, USA: ACM, 2016, pp. 1465–1474.

[50] J. Huang, M. Deng, J. Tang, S. Hu, H. Liu, S. Wariyo, and J. He, "Automatic generation of road maps from low quality gps trajectory data via structure learning," IEEE Access, vol. 6, pp. 71 965–71 975, 2018.

[51] M. Ahmed and C. Wenk, "Constructing street networks from gps trajectories," in Proceedings of the 20th Annual European Conference on Algorithms, ser. ESA'12. Berlin, Heidelberg: Springer-Verlag, 2012, pp. 60–71. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-33090-2_7

[52] S. Karagiorgou and D. Pfoser, "On vehicle tracking data-based road network generation," in Proceedings of the 20th International Conference on Advances in Geographic Information Systems, ser. SIGSPATIAL '12. New York, NY, USA: ACM, 2012, pp. 89–98. [Online]. Available: http://doi.acm.org/10.1145/2424321.2424334

[53] J. Biagioni and J. Eriksson, "Map inference in the face of noise and disparity," in Proceedings of the 20th International Conference on Advances in Geographic Information Systems, ser. SIGSPATIAL '12. New York, NY, USA: ACM, 2012, pp. 79–88. [Online]. Available: http://doi.acm.org/10.1145/2424321.2424333

[54] L. Cao and J. Krumm, "From gps traces to a routable road map," in Proceedings of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ser. GIS '09. New York, NY, USA: ACM, 2009, pp. 3–12. [Online]. Available: http://doi.acm.org/10.1145/1653771.1653776

[55] K. Buchin, M. Buchin, D. Duran, B. T. Fasy, R. Jacobs, V. Sacristan, R. I. Silveira, F. Staals, and C. Wenk, "Clustering trajectories for map construction," in Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ser. SIGSPATIAL '17. New York, NY, USA: ACM, 2017, pp. 14:1–14:10. [Online]. Available: http://doi.acm.org/10.1145/3139958.3139964

[56] M. Ahmed, S. Karagiorgou, D. Pfoser, and C. Wenk, "A comparison and evaluation of map construction algorithms using vehicle tracking data," GeoInformatica, vol. 19, no. 3, pp. 601–632, Jul 2015. [Online]. Available: https://doi.org/10.1007/s10707-014-0222-6

[57] X. Liu, J. Biagioni, J. Eriksson, Y. Wang, G. Forman, and Y. Zhu, "Mining large-scale, sparse gps traces for map inference: Comparison of approaches," in Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ser. KDD '12. New York, NY, USA: ACM, 2012, pp. 669–677. [Online]. Available: http://doi.acm.org/10.1145/2339530.2339637

[58] J. Biagioni and J. Eriksson, "Inferring road maps from global positioning system traces: Survey and comparative evaluation," Transportation Research Record, vol. 2291, no. 1, pp. 61–71, 2012. [Online]. Available: https://doi.org/10.3141/2291-08

[59] A. Steiner and A. Leonhardt, "Map generation algorithm using low-frequency vehicle position data," in TRB 90th Annual Meeting of the Transportation Research Board. Transportation Research Board, 2011, pp. 1–17.

[60] S. Worrall and E. Nebot, "Automated process for generating digitised maps through gps data compression," 01 2007.

[61] B. Niehöfer, R. Burda, C. Wietfeld, F. Bauer, and O. Lueert, "Gps community map generation for enhanced routing methods based on trace-collection by mobile phones," in 2009 First International Conference on Advances in Satellite and Space Communications, July 2009, pp. 156–161.

[62] A. Fathi and J. Krumm, "Detecting road intersections from gps traces," in Geographic Information Science, S. I. Fabrikant, T. Reichenbacher, M. van Kreveld, and C. Schlieder, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 56–69.

[63] M. Ezzat, M. Sakr, R. Elgohary, and M. E. Khalifa, "Building road segments and detecting turns from gps tracks," Journal of Computational Science, vol. 29, pp. 81 – 93, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877750318302813

[64] X. Xie, W. Liao, H. Aghajan, P. Veelaert, and W. Philips, "Detecting road intersections from gps traces using longest common subsequence algorithm,"

ISPRS International Journal of Geo-Information, vol. 6, no. 1, 2017. [Online]. Available: http://www.mdpi.com/2220-9964/6/1/1

[65] R. W. Sinnott, "Virtues of the haversine," Sky and Telescope, vol. 68, p. 159, 1984.

[66] S. Dabiri and K. Heaslip, "Inferring transportation modes from gps trajectories using a convolutional neural network," Transportation Research Part C: Emerging Technologies, vol. 86, pp. 360 – 371, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0968090X17303509

[67] H. Qian and Y. Lu, "Simplifying gps trajectory data with enhanced spatial-temporal constraints," ISPRS International Journal of Geo-Information, vol. 6, no. 11, 2017. [Online]. Available: http://www.mdpi.com/2220-9964/6/11/329

[68] OpenStreetMap. (2013). [Online]. Available: http://www.openstreetmap.org

[69] S. Karagiorgou, D. Pfoser, and D. Skoutas, "Segmentation-based road network construction," in Proceedings of the 21st ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, ser. SIGSPATIAL'13. New York, NY, USA: ACM, 2013, pp. 460–463.

[70] Z. Wang, X. Liu, L. Liu, and J. Shi, "A method of road extraction from high resolution remote image based on delaunay algorithms," in 2018 International Conference on Robots Intelligent System (ICRIS), May 2018, pp. 127–130.

[71] H. M. Uwe Bacher, "Automatic road extraction from multispectral high resolution satellite images," IAPRS, vol. XXXVI, no. 16, pp. 29–34, 2005.

[72] Y. Zang, C. Wang, Y. Yu, L. Luo, K. Yang, and J. Li, "Joint enhancing filtering for road network extraction," IEEE Transactions on Geoscience and Remote Sensing, vol. 55, no. 3, pp. 1511–1525, March 2017.

[73] K. Buchin, M. Buchin, D. Duran, B. T. Fasy, R. Jacobs, V. Sacristán, R. I. Silveira, F. Staals, and C. Wenk, "Clustering trajectories for map construction," in Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, GIS 2017, Redondo Beach, CA, USA, November 7-10, 2017, 2017, pp. 14:1–14:10. [Online]. Available: http://doi.acm.org/10.1145/3139958.3139964

[74] C. Chen, C. Lu, Q. Huang, Q. Yang, D. Gunopulos, and L. J. Guibas, "City-scale map creation and updating using GPS collections," in Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016, 2016, pp. 1465–1474. [Online]. Available: http://doi.acm.org/10.1145/2939672.2939833

[75] S. Karagiorgou and D. Pfoser, "On vehicle tracking data-based road network generation," in SIGSPATIAL 2012 International Conference on Advances in Geographic Information Systems (formerly known as GIS), SIGSPATIAL'12, Redondo Beach, CA, USA, November 7-9, 2012, 2012, pp. 89–98. [Online]. Available: http://doi.acm.org/10.1145/2424321.2424334

[76] C. Poullis and S. You, "Delineation and geometric modeling of road networks," ISPRS Journal of Photogrammetry and Remote Sensing, vol. 65, no. 2, pp. 165 – 181, 2010. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0924271609001233

[77] A. Katartzis, H. Sahli, V. Pizurica, and J. Cornelis, "A model-based approach to the automatic extraction of linear features from airborne images," IEEE Transactions on Geoscience and Remote Sensing, vol. 39, no. 9, pp. 2073–2079, Sept 2001.

[78] J. Yang and R. S. Wang, "Classified road detection from satellite images based on perceptual organization," Int. J. Remote Sens., vol. 28, no. 20, pp. 4653–4669, Jan. 2007. [Online]. Available: http://dx.doi.org/10.1080/01431160701250382

[79] W. Shi, Z. Miao, and J. Debayle, "An integrated method for urban main-road centerline extraction from optical remotely sensed imagery," IEEE Transactions on Geoscience and Remote Sensing, vol. 52, no. 6, pp. 3359–3372, June 2014.

[80] J. B. Mena, "State of the art on automatic road extraction for gis update: A novel classification," Pattern Recogn. Lett., vol. 24, no. 16, pp. 3037–3058, Dec. 2003. [Online]. Available: http://dx.doi.org/10.1016/S0167-8655(03)00164-8

[81] Y. Z. F. W. T. C. Weixing Wang, NanYang and P. Eklund, "A review of road extraction from remote sensing images," Journal of Traffic and Transportation Engineering, vol. 3, pp. 271–282, June 2016.

[82] H. Y.-A. WU Liang, "A survey of automatic road extraction from remote sensing images," Acta Automatica Sinica, vol. 36, no. 7, p. 912, 2010. [Online]. Available: http://www.aas.net.cn/EN/abstract/article_13651.shtml

[83] TxDOT, "Texas department of transportation," 2016. [Online]. Available: https://www.txdot.gov

[84] N.-E. E. Faouzi and L. A. Klein, "Data fusion for its: Techniques and research needs," Transportation Research Procedia, vol. 15, pp.

495 – 512, 2016, international Symposium on Enhancing Highway Performance (ISEHP), June 14-16, 2016, Berlin. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S2352146516305749

[85] A. Essien, I. Petrounias, P. Sampaio, and S. Sampaio, "Improving urban traffic speed prediction using data source fusion and deep learning," in 2019 IEEE International Conference on Big Data and Smart Computing (BigComp), 2019, pp. 1–8.

[86] H. Nguyen, L. Kieu, T. Wen, and C. Cai, "Deep learning methods in transportation domain: a review," IET Intelligent Transport Systems, vol. 12, no. 9, pp. 998–1004, 2018.

[87] N.-E. E. Faouzi, H. Leung, and A. Kurian, "Data fusion in intelligent transportation systems: Progress and challenges - a survey," Inf. Fusion, vol. 12, no. 1, p. 4–10, Jan. 2011. [Online]. Available: https://doi.org/10.1016/j.inffus.2010.06.001

[88] W. Chen, J. An, R. Li, L. Fu, G. Xie, M. Z. A. Bhuiyan, and K. Li, "A novel fuzzy deep-learning approach to traffic flow prediction with uncertain spatial–temporal data features," Future Generation Computer Systems, vol. 89, pp. 78 – 88, 2018. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X18307398

[89] Y. Li, R. Yu, C. Shahabi, and Y. Liu, "Diffusion convolutional recurrent neural network: Data-driven traffic forecasting," 2017.

[90] X. Luo, D. Li, Y. Yang, and S. Zhang, "Spatiotemporal traffic flow prediction with knn and lstm," Journal of Advanced Transportation, vol. 2019, pp. 1–10, 02 2019.

[91] T. Bogaerts, A. Masegosa, J. Angarita-Zapata, E. Onieva, and P. Hellinckx, "A graph cnn-lstm neural network for short and long-term traffic forecasting based on trajectory data," Transportation Research Part C Emerging Technologies, vol. 112, pp. 62–77, 03 2020.

[92] S. Moosavi, M. H. Samavatian, S. Parthasarathy, R. Teodorescu, and R. Ramnath, "Accident risk prediction based on heterogeneous sparse data," Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, Nov 2019. [Online]. Available: http://dx.doi.org/10.1145/3347146.3359078

[93] Z. Zhao, W. Chen, X. Wu, P. C. Y. Chen, and J. Liu, "Lstm network: a deep learning approach for short-term traffic forecast," IET Intelligent Transport Systems, vol. 11, no. 2, pp. 68–75, 2017.

[94] X. Dai, R. Fu, Y. Lin, L. Li, and F.-Y. Wang, "Deeptrend: A deep hierarchical neural network for traffic flow prediction," 2017.

[95] M. Fouladgar, M. Parchami, R. Elmasri, and A. Ghaderi, "Scalable deep traffic flow neural networks for urban traffic congestion prediction," in 2017 International Joint Conference on Neural Networks, IJCNN 2017, Anchorage, AK, USA, May 14-19, 2017. IEEE, 2017, pp. 2251–2258. [Online]. Available: https://doi.org/10.1109/IJCNN.2017.7966128

[96] A. Koesdwiady, R. Soua, and F. Karray, "Improving traffic flow prediction with weather information in connected cars: A deep learning approach," IEEE Transactions on Vehicular Technology, vol. 65, no. 12, pp. 9508–9517, 2016.

[97] L. N. N. Do, N. Taherifar, and H. L. Vu, "Survey of neural network-based models for short-term traffic state prediction," WIREs Data Mining and

Knowledge Discovery, vol. 9, no. 1, p. e1285, 2019. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/widm.1285

[98] K. Guo, T. Xu, X. Kui, R. Zhang, and T. Chi, "ifusion: Towards efficient intelligence fusion for deep learning from real-time and heterogeneous data," Information Fusion, vol. 51, pp. 215 – 223, 2019. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1566253518304834

[99] F. E. White, "Data fusion subpanel of the joint directos of laboratories technical panel for c3," San Diego: Californie, 1991.

[100] F. Castanedo, "A review of data fusion techniques," The Scientific World Journal, vol. 2013.

[101] O. Sidek and S. Quadri, "A review of data fusion models and systems," International Journal of Image and Data Fusion, vol. 3, no. 1, pp. 3–21, 2012. [Online]. Available: https://doi.org/10.1080/19479832.2011.645888

[102] S. Ben Ayed, H. Trichili, and A. M. Alimi, "Data fusion architectures: A survey and comparison," in 2015 15th International Conference on Intelligent Systems Design and Applications (ISDA), 2015, pp. 277–282.

[103] F. Mastrogiovanni, A. Sgorbissa, and R. Zaccaria, "A distributed architecture for symbolic data fusion." 01 2007, pp. 2153–2158.

[104] B. V. Dasarathy, "Sensor fusion potential exploitation-innovative architectures and illustrative applications," Proceedings of the IEEE, vol. 85, no. 1, pp. 24–38, 1997.

[105] D. Zhang and M. R. Kabuka, "Combining weather condition data to predict traffic flow: A gru based deep learning approach," in 2017 IEEE

15th Intl Conf on Dependable, Autonomic and Secure Computing, 15th Intl Conf on Pervasive Intelligence and Computing, 3rd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress(DASC/PiCom/DataCom/CyberSciTech), 2017, pp. 1216–1219.

[106] S. Dunne and B. Ghosh, "Weather adaptive traffic prediction using neurowavelet models," IEEE Transactions on Intelligent Transportation Systems, vol. 14, no. 1, pp. 370–379, 2013.

[107] Y. Jia, J. Wu, M. Ben-Akiva, R. Seshadri, and Y. Du, "Rainfall-integrated traffic speed prediction using deep learning method," IET Intelligent Transport Systems, vol. 11, no. 9, pp. 531–536, 2017.

[108] X. Yang, Y. Yuan, and Z. Liu, "Short-term traffic speed prediction of urban road with multi-source data," IEEE Access, vol. 8, pp. 87 541–87 551, 2020.

[109] A. Ali, "Traffic forecasting applications using crowdsourced traffic reports and deep learning," Ph.D. dissertation, University of North Texas, 2020.

[110] N. G. Polson and V. O. Sokolov, "Deep learning for short-term traffic flow prediction," Transportation Research Part C: Emerging Technologies, vol. 79, pp. 1 – 17, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0968090X17300633

[111] R. Yu, Y. Li, C. Shahabi, U. Demiryurek, and Y. Liu, "Deep learning: A generic approach for extreme condition traffic forecasting," in SDM, 2017.

[112] C. D. of Transportation (Caltrans). Caltrans Peformance Measurement System (PeMS). (2020). [Online]. Available: http://pems.dot.ca.gov

[113] C. Chen, "Freeway performance measurement system (pems)," 2003.

[114] M. Stae Inc. (2020). [Online]. Available: https://municipal.systems/explore

[115] Waze. (2020). [Online]. Available: https://www.waze.com

[116] D. Sky. (2020). [Online]. Available: https://darksky.net/

[117] G. Van Rossum and F. L. Drake, Python 3 Reference Manual. Scotts Valley, CA: CreateSpace, 2009.

[118] W. McKinney et al., "Data structures for statistical computing in python," in Proceedings of the 9th Python in Science Conference, vol. 445. Austin, TX, 2010, pp. 51–56.

[119] M. Lippi, M. Bertini, and P. Frasconi, "Short-term traffic flow forecasting: An experimental comparison of time-series analysis and supervised learning," IEEE Transactions on Intelligent Transportation Systems, vol. 14, no. 2, pp. 871–882, 2013.

[120] C. Chen, K. Li, S. G. Teo, X. Zou, K. Wang, J. Wang, and Z. Zeng, "Gated residual recurrent graph neural networks for traffic prediction," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 485–492.

[121] S. Hochreiter and J. Schmidhuber, "Long short-term memory," Neural computation, vol. 9, no. 8, pp. 1735–1780, 1997.

[122] Y. Tian and L. Pan, "Predicting short-term traffic flow by long short-term memory recurrent neural network," in 2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity), 2015, pp. 153–158.

[123] K. Cho, B. van Merrienboer, Ç. Gülçehre, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for

statistical machine translation," CoRR, vol. abs/1406.1078, 2014. [Online]. Available: http://arxiv.org/abs/1406.1078

[124] Z. Cui, R. Ke, and Y. Wang, "Deep bidirectional and unidirectional LSTM recurrent neural network for network-wide traffic speed prediction," CoRR, vol. abs/1801.02143, 2018. [Online]. Available: http://arxiv.org/abs/1801.02143

[125] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.

[126] F. Chollet. (2015) Keras. [Online]. Available: https://github.com/fchollet/keras

[127] S. Guo, Y. Lin, N. Feng, C. Song, and H. Wan, "Attention based spatial-temporal graph convolutional networks for traffic flow forecasting," in Proceedings of the AAAI Conference on Artificial Intelligence, vol. 33, 2019, pp. 922–929.

BIOGRAPHICAL STATEMENT

Tariq Alsahfi is a PhD candidate in the Department of Computer Science and Engineering, at the University of Texas at Arlington and a lecturer at the Departments of Information System at the University of Jeddah. He completed his Bachelor in 2011 and his Master's in 2015 in Computer Science. His research interest is the geographical information systems, trajectory data, deep learning for transportation systems. Currently, he is working develop deep learning models for traffic flow forecasting.