Author/s:
Kannangara, Don Lashantha Sameera

Title:
Public movement analysis using location based social network data

Date:
2020

Persistent Link:
https://hdl.handle.net/11343/268136

Public movement analysis using location based social network data

Don Lashantha Sameera Kannangara

0000-0003-1030-9127

Submitted in total fulfillment of the requirements of the degree of

Doctor of Philosophy

School of Computing and Information Systems

THE UNIVERSITY OF MELBOURNE

October 2020

Abstract

Location Based Social Networks (LBSNs) provide an inexpensive source of data to analyse public movement. However, it is difficult to process LBSN to get useful information due to the sparsity and irregularity associated with data. These inherent properties of LBSN data are caused by the voluntary posting to the LBSN. Therefore we can say voluntary posting of data is both a strength and a weakness of LBSN data. In this thesis, we propose several methods to process LBSN data to extract useful information.

Information from LBSN data can be used to identify areas used for movement and the most popular paths. A clever way to extract these information is to first construct a graph structure based on the posted locations and query this graph. Neighbourhood graphs are useful structures to analyse the movement between a set of locations. More specifically we look at two candidates the Relative Neighbourhood Graph and the Shortest Path Graph, but both have weaknesses making them less suitable for our domain. By analysing the relationship between these two graphs, we propose the Stepping Stone graph with local criterion named the Diversion Neighbourhood which captures movement related information between two points. Due to the shape of the Diversion Neighbourhood, the Stepping Stone Graph is very effective in calculating movement related queries, and because the Diversion Neighbourhood is a local criterion, the Stepping Stone graph is efficient to create. In this thesis, we provide how the Stepping Stone Graph is related to other well-known graphs and empirically show how it is useful for LBSN data processing by answering a few spatial queries using real LBSN data.

Modern use cases of LBSN data processing need to generate results in real time. Even though the Stepping Stone Graph is suitable to process LBSN data sets with moderate number of post with locations, it takes a considerable time to process very large data sets. Continuing our research on neighbourhood graphs, in order to address this scalability issue we propose the Diversion Graph by relaxing the evaluation criteria of the Stepping Stone Graph. Due to this relaxed criteria, the Diversion Graph takes 10% of the time required to calculate the Stepping Stone Graph of the same location set. However, the

Diversion Graph contains a maximum of 2% additional edges than the Stepping Stone Graph, which is an undesirable effect. When considering creation time reduction against the number of additional edges, the Diversion Graph provides a favourable trade off for LBSN data analysis in numerous scenarios. We show this increased performance of the Diversion Graph against the Stepping Stone Graph by processing a few application queries that require to process a large number of locations filtered from real LBSN data and synthetic data.

Due to the sparsity and irregularity associated with LBSN data, there are use cases that cannot be solved by the data structures previously proposed. One such important use case useful for many fields is predicting group movement. Due to the lack of techniques to track multiple moving objects as a group using sparse irregular data, addressing this use case using LBSN data is difficult. To address this issue we turn our attention to algorithms in signal processing and time series analysis. LBSN trajectory of a single user can be considered a location signal collected over time. Group analysis relates to processing multiple of these signals together. We propose a new technique named the Group Kalman Filter, by extending the well-known technique named the Kalman filter which is used for processing multiple signals. When using LBSN data to track group movement, not all members of the group post continuously to the LBSN. Therefore we have to assume that locations of the members who are not posting are reflected by the members who are posting with the group. We develop a metric to quantify this behaviour and use it to detect complex group movement patterns such as group merging and group splitting. We propose four group movement models to track and predict the movement of detected groups with their advantages and disadvantages. Using real LBSN data and synthetic location data that mimic LBSN data distributions, we show that the Group Kalman Filter is both effective and efficient to detect and predict group movement using LBSN data as our final contribution to the area with this thesis.

Declaration

This is to certify that

1. the thesis comprises only my original work towards the PhD,
2. due acknowledgement has been made in the text to all other material used,
3. the thesis is less than 100,000 words in length, exclusive of tables, maps, bibliographies and appendices.

Don Lashantha Sameera Kannangara, October 2020

Acknowledgements

First of all, I would like to thank my principal supervisor, Prof. Egemen Tanin and my co-supervisor, Prof. Shanika Karunasekera for their continuous guidance and immense support throughout my PhD candidature. Also, I thank Dr. Aaron Harwood as his advice and research insights were key inputs to make my PhD journey a success. I am extremely grateful for their invaluable time, patience and encouragement which helped me grow as a researcher. I would also like to express my gratitude to A/Prof. Atif Ahmad, my advisory committee chair for his continuous support and words of encouragement.

I acknowledge the financial support provided by the Melbourne Research Scholarship and Defence Science and Technology Group. I am also grateful to the School of Computing and Information Systems for providing the facilities to pursue my research.

I am thankful to all my friends in the department, Sadegh Motallebi, Soheila Ghane, Mahtab Mirmomeni, Namrata Srivastava, Partha De, Nicholas Akinyokun, and Gayashan Amarasinghe for their support and company throughout my PhD journey. I am also grateful to Dr. Hairuo Xie for his support as a senior postdoctoral fellow.

I would like to thank my parents and my sisters for their love and support. It would not have been possible to pursue my PhD without their guidance and encouragement. I am also grateful to my cousin, Hashini and her family for all the love and comforting words. Last but not the least, I am thankful to my wife, Sachini, for always being there for me and for being my source of strength.

Preface

The main contributions of this thesis are discussed in Chapters 3, 4 and 5, which are respectively based on the following published or accepted papers. The thesis research has been carried out at The School of Computing and Information Systems, The University of Melbourne. I am the primary author of these papers and have contributed more than 50%.

- **Sameera Kannangara**, Egemen Tanin, Aaron Harwood, and Shanika Karunasekera. "Stepping stone graph for public movement analysis." In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pp. 149-158. 2018.
- **Sameera Kannangara**, Egemen Tanin, Aaron Harwood, and Shanika Karunasekera. "Stepping Stone Graph: A Graph for Finding Movement Corridors using Sparse Trajectories." *ACM Transactions on Spatial Algorithms and Systems (TSAS)* 5, no. 4 (2019): 1-24.
- **Sameera Kannangara**, Hairuo Xie, Egemen Tanin, Aaron Harwood, and Shanika Karunasekera. "Introducing Diversion Graph for Real-time Spatial Data Analysis with Location Based Social Networks." *11th International Conference on Geographic Information Science (GIScience 2021)*, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2020.
- **Sameera Kannangara**, Hairuo Xie, Egemen Tanin, Aaron Harwood, and Shanika Karunasekera. "Tracking Group Movement with Location Based Social Networks."

Proceedings of the 28th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems. 2020.

*To my **father**, **mother** and **wife***

Contents

List of Figures

Chapter 1

Introduction

Public movement analysis refers to analysing people's movement data, in order to gain understanding of public movement patterns. Public movement analysis is helpful in understanding regular movement patterns of the public, identifying anomalies in movement patterns and how some aspects of human behaviour propagates with movement [1]. Results from public movement analysis can be used for event detection, popular path detection, travel network extraction, disease propagation analysis and group movement tracking. Traditional method of data collection for public movement analysis is to use specialized mechanisms to collect data using autonomous GPS loggers, call data records and wifi access data records. Most of these techniques involve privacy concerns and/or low resolution of collected data.

Affordability of Global Positioning System (GPS) enabled devices has allowed the public to communicate their location to the world effortlessly. Location Based Social Networks (LBSN) application domain is one which utilizes people's location effectively, by enabling users to publish their location along with published content to the social network. Twitter, Flicker and Foursquare are some example LBSNs that allow users to publish their location. Since the user generated content in LBSN is often inspired by users' experiences, it is a valuable source of information to analyse user behaviour. Aforementioned LBSNs provide public Application Programming Interfaces (APIs) that release data for research purposes. In addition to being an inexpensive data source for public movement analysis, LBSNs can provide a more privacy aware way to access data as

users have full control of when and where to post about their activities. In this thesis, we aim to analyse the public movement using these publicly available LBSN data.

Depending on the way location data is collected for analysis, we can categorise data into two broad categories machine-generated data and user-generated data. Machine-generated data refers to dense location data sourced by autonomous systems associated with a moving object such as a vehicle or a mobile phone in small regular time intervals. Therefore machine generated data contain a detailed record of even a small movement. On the contrary, user-generated data is generated when a user decides to record their location to a system voluntarily. LBSN data and applications that allow users to record their locations when they want are examples of user-generated location data sources. Due to the voluntary generation of data, user generated data tend to be sparse and irregular, making only some of the location history of the reporting user available for analysis. Hence movement analysis techniques that work on machine-generated location data do not usually generate useful results from user-generated data. It is important to investigate and develop specialized movement analysis mechanisms to process sparse and irregular user-generated data.

Important information about different events and emergencies can be gathered in real-time using LBSN data. For example, by analysing movements of users using the LBSN data related to a particular event, we can infer information such as the common routes taken by the participants to come to the event or LBSN posts that mention keywords related to an emergency can be used for crisis management. The gathered content can be used to analyse different situations, identify irregularities in user movements and to enhance the understanding of situations. Analysing the spatial and temporal characteristics related to the day-to-day lives of the public is a crucial step towards establishing situational awareness. Since LBSN provides a voluntarily generated information about users, developing systems to support situational awareness using LBSN data is a trending research area [2, 3]. In these systems, it is noted that the visualizations generated to provide an overview of the spatial and temporal distribution of LBSN data relating to an event is specifically useful for behaviour analysis.

1.1 Location Based Social Network Data for Public Movement Analysis

A single element of user generated content in LBSN is called an LBSN post. An LBSN post can be viewed as a combination of four data items - identification of the user, voluntarily generated content, the location where the content is generated and the timestamp when the content is generated. Therefore, LBSN data can be analysed based on a single aspect of the data or a combination of aspects to reveal new information.

Since LBSN data is voluntarily generated users decide when and where to post content. Most users of LBSN have low and irregular posting patterns making it difficult to process LBSN data based on a single user trajectory. However, by combining this uncertain data we can come up with more certain information. The location of these LBSN posts come as a GPS point. When combining these point location data to extract movement information, mechanisms that connect these points using edges are useful for analysing movement. One such mechanism class is known as neighbourhood graphs. Neighbourhood graphs infer edges between two points based on the properties of the neighbourhood surrounding those two points. We can use a selected neighbourhood graph to create a structure between LBSN posts' locations in order to do further processing. However, it remains a problem as to which graph to select to generate this structure as there are many graphs with different properties. The selected graph should be effectively captured movement related information that is available through LBSN data and should be efficient to generate and process. Due to the availability to GPS enabled mobile phones we can expect high volumes of location data even for a small scale event.

Another interesting problem that can be investigated using LBSN data is group movement prediction. In order to detect groups in a traditional setting, potential group members must be equipped with specialized pieces of equipment to collect the location of the group member regularly with their identification. With the availability of LBSNs where any user can post their location freely, it is important to look into how LBSN data can be used in group movement detection and prediction. However, we have to be aware of

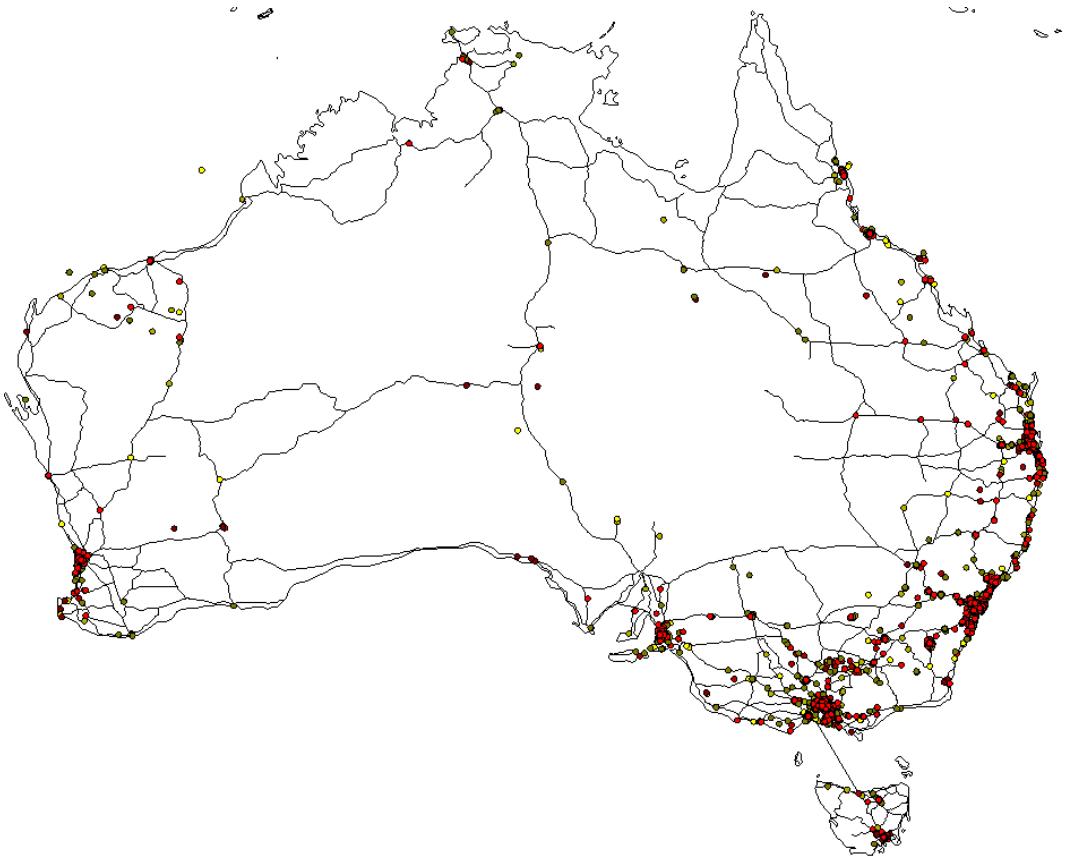


Figure 1.1: Distribution of LBSN posts from a Twitter data set over Australia.

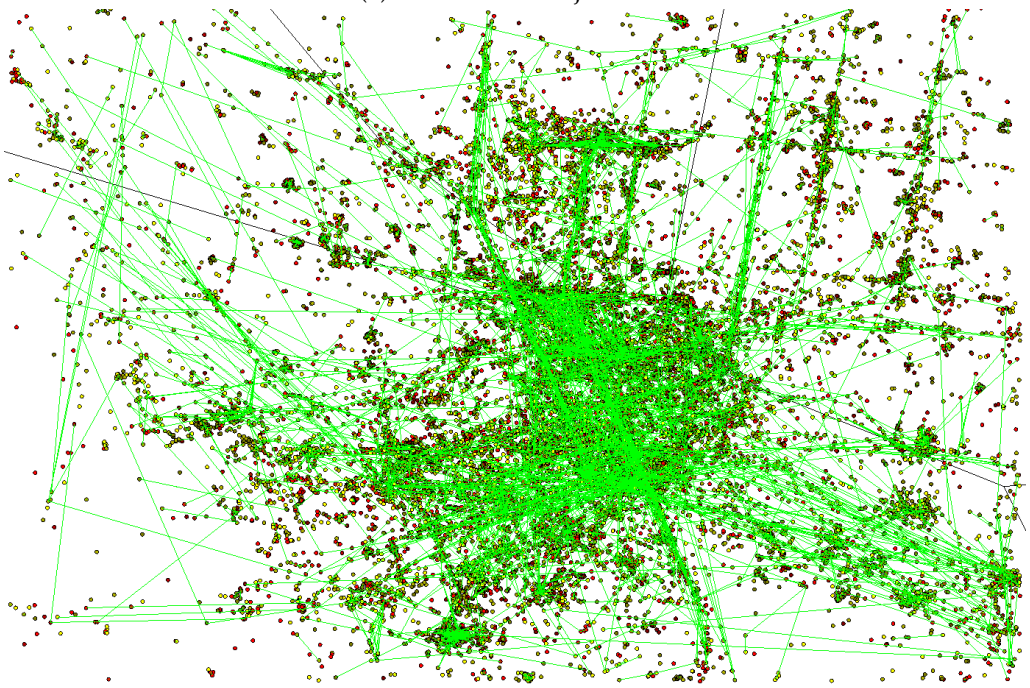
the negative impact created by the inherent properties of the LBSN data such as low and irregular posting frequencies.

1.2 Challenges of Public Movement Analysis using LBSN Data

1.2.1 Low and Irregular Posting Patterns

In LSBNs data generation is voluntary; users decide when to post content with a location. Most users are reluctant to post their location online due to privacy reasons. As a result, the number of geo tagged posts typically is relatively low compared to the total number of posts in social networks. For example in the Twitter data set we use, 70% of the users

(a) One minute trajectories



(b) Five minute trajectories

Figure 1.2: Trajectory trace distribution over Melbourne, Australia. Circles indicate LBSN posts, while green lines indicate trajectory traces.

have posted less than 10 geo-tagged posts in one month time period, which amounts to 11.5% of all records in the data set. Remaining 30% of the users have posted 88.5% of the whole data set. Another problem experienced in LBSN data is irregular posting patterns. Some users post with high frequency in a short period of time and do not post any LBSN posts for a long time. Even when users post with high frequency compared to the other users, it is not usually sufficient to analyse movement patterns of individual users [4]. Another form of irregularity observable in LBSN data is that posted locations are concentrated towards, roads and urban areas. Figure 1.1 depicts the distribution of posted locations in our Twitter data set. Note how location patterns highlight major highways and highly populated cities on the map. Irregularity of posting patterns of individual users are highlighted in Figure 1.2 which shows the trajectory traces taken from individual users. One minute trajectories shown in Figure 1.2 (a) are way less compared to the five minute trajectories shown in Figure 1.2 (b). Also, note how trajectory patterns are scattered making individual's movement analysis difficult. As a result of above posting patterns, it is hard to propose methods to process LBSN data.

1.2.2 Need for Real-time Processing

Most LBSNs have millions of users. With the availability of GPS enabled mobile phones these users can generate a high volume of data for even a small event. For example, 500 million tweets are posted on Twitter each day[1]. This may seem contradictory with the previously explained challenge; low and irregular posting frequencies. Data relating to individual users are sparse and irregular, but there are a lot of users. Also, this large number of users are spread across large areas, making the posted data collected with less density. This large amount of users generate a lot of data points which are sparse and irregular making it hard to process due to both volume and sparsity. In order to support modern use cases of processing, we need to process these increased amounts of data in a short amount of time. This processing includes both offline processing of collected data and online processing of streaming data. When proposing methods to process modern

[1]https://www.oberlo.com/blog/twitter-statistics

LBSN data we have to consider proposing methods to process in both situations.

1.2.3 Combining Data From Different Users

Due to the inherent properties of LBSN data such as low and irregular posting patterns, It is difficult to analyse movements of individual users. However, we can combine location data from different users and make a less uncertain picture of public movement. Also, it is hard to combine data from different users and process them due to the uncertainty associated with LBSN data. Due to these problems, it is hard to propose methods solve use cases such as group movement prediction and trajectory clustering using LBSN data. Even though we can detect a combined movement from time to time, it is less likely that all the members in the combined movement will post continuously to track and predict the movement. Therefore we need to compensate for that fact and we have to rely only on the portion of the locations from the combined movement and have to assume that the rest of the users who are not posting are represented by the available data.

1.3 Research Problems and Objectives

The focus of this research is to develop effective and efficient mechanisms for conducting public movement analysis using LBSN data. To address the challenges stemming from the properties of LBSN data, this thesis explores the following key research problems:

- **How to effectively and efficiently model LBSN data for movement analysis?** It is difficult to propose methods to process LBSN data to extract information relating to an event or a specific area due to the aforementioned properties such as sparsity and irregularity of LBSN data. We explore data structures to effectively capture the movement related aspects posted on LBSN data. Among available data structures, graphs that can capture and represent relationships between elements are especially useful in this use case. Our investigation leads us to explore the options with

neighbourhood graphs, which infer edges between location vertices based on the properties of the surroundings of locations. Most of the available neighbourhood graphs create a static graph structure, which cannot be configured to suit the location distribution relating to the situation. Therefore, most of the available neighbourhood graphs failed to process LBSN data effectively and efficiently. To make a configurable graph that can change the captured information based on the situation, we look at variable graphs. Variable graphs can change the created structure based on parameters to suit the distribution of analysed data. Since available variable neighbourhood graphs are not created with the focus of analysing movement data, we investigate the existence of a new variable neighbourhood graph focused on answering movement related queries.

- **How to process LBSN data in real time using neighbourhood graphs?** LBSNs generate data at very high speeds and modern use cases need to process this data in the least possible time. Therefore, our next research question investigates improving the processing time of the solutions we discovered relating to the previous research question. As mentioned in the previous research question we are investigating to propose new variable neighbourhood graphs with our main focus on increased effectiveness of the solution. Since we are aiming to reduce the processing time in this research problem, we are reducing our focus on effectiveness and increase the focus on efficiency. In other terms with this research question, we explore the possibility of developing techniques that create a useful trade off between effectiveness and efficiency. As the solution, we investigate to relax the evaluation condition on the graph data structure proposed for the previous problem and increase the efficiency of analysis. Due to the relaxation of the evaluation criteria, the graph data structure can contain additional impurities which affect negatively to the analysis. The objective here is to find a suitable balance between the existence of impurities against the increased performance to process increased amounts of data.

- **How can we predict the movement of groups using LBSN data?** LBSN provides an inexpensive source of data for group movement detection and prediction. This

use case involves combining incomplete movement data from different users. This is a difficult problem to solve using traditional methods available to process GPS data. These traditional methods can safely assume that all the group members continuously emit their location for the analysis, making them less effective when applied on LBSN data where the same processing needs to be done with incomplete data. Hence it is essential to investigate mechanisms that can combine incomplete data from different users to solve problems such as group movement prediction using sparse irregular data. Since this problem cannot be solved by using only proposing new data structures, we investigate algorithmic solutions. Furthermore, group movement analysis is difficult due to the increased dimensional space of the problem as there are multiple variable users. We investigate available literature to effectively deal with such problems and predict group movement using incomplete LBSN data. By considering user trajectories in LBSN data as signals generated by multiple users moving as a group, we explore solutions by extending algorithms from signal processing and time series analysis.

1.4 Thesis Contribution

In this thesis, we first explore data structures that can help us effectively process LBSN data for movement analysis. To mitigate the rigid nature of static neighbourhood graphs and poor performance of existing variable neighbourhood graphs, we investigate to link most suitable candidates from these two categories and propose a new graph data structure. In order to analyse movement we mainly process the location, the timestamp and the userID related to an LBSN post. Next, we work on improving the proposed data structures to increase their efficiency when processing increased amounts of LBSN data. Due to the inherent properties of LBSN data, not all movement related use cases can be addressed by proposing only data structures. Therefore we advance our research by proposing algorithms to predict group movement using LBSN data. For this use case, we successfully propose a solution using a well-known algorithm from signal processing

and time series analysis named the Kalman filter. Main contributions of this thesis can be summarized as follows:

1. **Introducing the Stepping Stone Graph for public movement analysis using LBSN data.** It is a new graph data structure that creates edges between points based on a local evaluation criterion capturing movement related information. New evaluation criterion named the Diversion neighbourhood is used to extract the Stepping Stone Graph. The planar Stepping Stone Graph can be efficiently extracted using the Delaunay triangulation. We have provided mathematical proofs relating the Stepping Stone Graph to well known graph structures such as the Gabriel graph, β - Skeletons and the relative neighbourhood graph. From our mathematical analysis, it is observed that the Stepping Stone Graph is more suitable for movement analysis than β - Skeletons due to the more even spread of the evaluation criteria Diversion neighbourhood. We have shown the usefulness of the Stepping Stone Graph by applying it for events analysis, solving nearest neighbour queries and most popular path finding. Using real world LBSN data we evaluated the effectiveness of the Stepping Stone Graph in the proposed use cases against the Shortest Path Graph. For all the use cases the Stepping Stone graph provides effective and efficient solutions compared to the Shortest Path Graph.

2. **Introducing the Diversion Graph for real time processing of LBSN data for movement analysis.** Even though the Stepping Stone Graph is efficient to process small scale data sets containing hundreds of locations, it takes longer to process larger data sets with thousands of locations. By relaxing the Diversion neighbourhood based evaluation criteria to focus less on effectiveness and focus more on efficiency we propose a new graph named the Diversion Graph for movement analysis. We show relationship between well known graph structures and the Diversion Graph through mathematical proofs. Our mathematical analysis shows that the Diversion Graph is not always equal to the Stepping Stone Graph and based on existing literature the Diversion Graph can contain maximally 2% of total edges in the Stepping Stone Graph as additional edges. Even though having additional edges is

disadvantageous for analysis, relaxation of evaluation criteria results in over 90% reduction of execution time. Therefore this trade-off is advantageous. We present some applications of the Stepping Stone Graph that can benefit from the reduced execution time of the Diversion Graph. In all the experiments comparing the Diversion Graph against the Stepping Stone Graph, the Diversion Graph takes less time to process the data set compared to the Stepping Stone Graph, while maintaining the same level of effectiveness as the Stepping Stone Graph.

3. **Introducing the Group Kalman Filter for group movement prediction using LBSN data.** Group movement prediction using LBSN data requires methods to combine incomplete movement information from different users. It is difficult to solve this research problem by only proposing data structures. Therefore we propose an algorithmic solution extending concepts from signal processing. We extend a generic signal processing technique named the Kalman filter which can process multiple signals. This technique is used to predict the next state of a process based on the previous state and the measurements of the current state. We propose to convert LBSN data into a format that can be processed by the Klaman filter. In order to apply Kalman filter on converted LBSN data we propose four movement models with increased complexity. Using real world data and synthetic data we validate the operation of the Group Kalman Filter. The architecture of the Group Kalman filter can be used to solve other location data combining problems such as trajectory clustering.

1.5 Thesis Organisation

The chapters of this thesis are organized as follows. Chapter 2 provides a literature review on Location data analysis, LBSN data processing, available neighbourhood graphs, and group movement analysis. Chapter 3 presents our first new graph data structure contribution, the Stepping Stone Graph with its application using LBSN data. Chapters 4 presents our second new graph data structure contribution, the Diversion graph and

validating the accuracy and performance benefits of the Diversion graph when applied to LBSN data analysis. Chapter 5 presents our third contribution, the Group Kalman Filter for group movement tracking with four group movement tracking models. Chapter 6 concludes the thesis with a summary of the thesis and discusses future research directions. The core chapters of the thesis are derived from publications completed during my PhD candidature, which are listed as follows:

- Chapter 3 proposes the Stepping Stone Graph and its applications. This chapter is derived from:
  - **Sameera Kannangara**, Egemen Tanin, Aaron Harwood, and Shanika Karunasekera. "Stepping stone graph for public movement analysis." In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pp. 149-158. 2018.
  - **Sameera Kannangara**, Egemen Tanin, Aaron Harwood, and Shanika Karunasekera. "Stepping Stone Graph: A Graph for Finding Movement Corridors using Sparse Trajectories." *ACM Transactions on Spatial Algorithms and Systems (TSAS)* 5, no. 4 (2019): 1-24.
- Chapter 4 introduces the Diversion Graph and confirms the performance benefits achieved using the relaxed evaluation criteria. This chapter is derived from:
  - **Sameera Kannangara**, Hairuo Xie, Egemen Tanin, Aaron Harwood, and Shanika Karunasekera. "Introducing Diversion Graph for Real-time Spatial Data Analysis with Location Based Social Networks." *11th International Conference on Geographic Information Science (GIScience 2021)*, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2020.
- Chapter 5 proposes the Group Kalman Filter and provides experimental results showing effectiveness and efficiency with real LBSN data. This chapter is derived from:
  - **Sameera Kannangara**, Hairuo Xie, Egemen Tanin, Aaron Harwood, and Shanika Karunasekera. "Tracking Group Movement with Location Based Social Net-

works." *Proceedings of the 28th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 2020.

Chapter 2

Literature Review

2.1 Introduction

The Location Based Social Network (LBSN) Data can be processed to reveal different kinds of useful movement related information, such as interesting events and most popular venues for a given activity. Public movement analysis is a selection of analysis techniques used to process location data for generating information related to movement patterns of the public. LBSNs provide rich and inexpensive data sources for public movement analysis. In this chapter, we will present existing literature relating to the LBSN data analysis focusing on public movement analysis using LBSN data.

Locations collected via LBSN data come in the form of GPS coordinates. When processing location data for movement analysis we need to build effective models of the data to process location data effectively and efficiently. As this is an important part of location data processing, next we present existing models used to represent location data for movement analysis. Among these existing models, we present existing neighbourhood graphs that can be used to model and process LBSN data. In this section, we present an existing gap between the Relative Neighbourhood Graph (a static neighbourhood graph) and the Shortest Path Graph (a variable neighbourhood graph). By analysing the relationship between these two graphs we propose the Stepping Stone Graph and the Diversion Graph to fill this gap. This section introduces various neighbourhood graphs that we are

going to mathematically relate to our newly created graphs. Content from this section will be referred to as needed in the later chapters when comparing our proposed graphs against existing graphs.

Since LBSN data is type of location data, processing techniques developed to process other types of location data to reveal movement patterns can be applied on LBSN data. Following this pathway, we then present existing literature on movement analysis using location data. Within the discussion of literature related to location data analysis, we present location data collection methods and processing of location data to generate public movement related information. We analyse group movement analysis as a separate section at the end of this section since we investigate this area separately and propose a new technique for group movement analysis using LBSN data.

After looking at the literature relating to location data processing and modelling location data, we present an analysis of existing literature relating to movement analysis using LBSN data. We focus on different types of analysis that can be performed using LBSN data and kinds of movement related information needs we can fulfil using LBSN data. Later in this section, we discuss combining individual movement information collected from LBSN data to create a collective understanding and finish with a discussion on the availability of LBSN data for processing.

In the next subsection, we present literature relating to combining individual location information from different users to analyse group movement. This section provides an analysis of existing literature relating to group movement tracking we conduct as our third research problem. We start this section by introducing the combination processing of GPS data. Later we shift the focus to discuss the usage of concepts from signal processing to analyse combined movement. This analysis highlights the lack of existing techniques for group movement tracking using sparse location data from sources such as LBSNs. To fill this gap in knowledge we propose the Group Kalman Filter by extending the well known signal processing method the Kalman filter. This chapter concludes with a summary of the discussed literature and a brief comparison of how our contributions relate to them.

2.2 Modelling of Location data for Movement Analysis

To process location data to obtain movement related information, we have to effectively and efficiently model location data. One of the useful modelling method of location data to analyse movement related aspects is graphs. In location data analysis, graphs represent locations as vertices and relationships among locations as edges. Among those existing graph structures, ones that can automatically create relationships between locations are specially useful for movement analysis. One such graph category is neighbourhood graphs. In the first subsection we discuss available neighbourhood graphs, their characteristics and existing gaps in graph knowledge. Another location data modelling technique useful for movement analysis is grid creation. The grids can be static and uniform or dynamic and variable. In the next subsection we discuss existing usage of grids in movement analysis.

2.2.1 Neighbourhood Graphs for Movement Analysis

Graphs are useful data structures to store and analyse relationships between data objects. Data objects that are subjected to relationships are usually called vertices, and created relationships between these data objects are represented by edges in a graph. When these edges do not signify any direction information between the vertices, they are called undirected graphs. On the other hand, when the direction information is associated with the edges, they are called directed graphs. To analyse location data both directed and undirected graphs are used. In this section, we discuss some available undirected graphs that are useful for LBSN data processing.

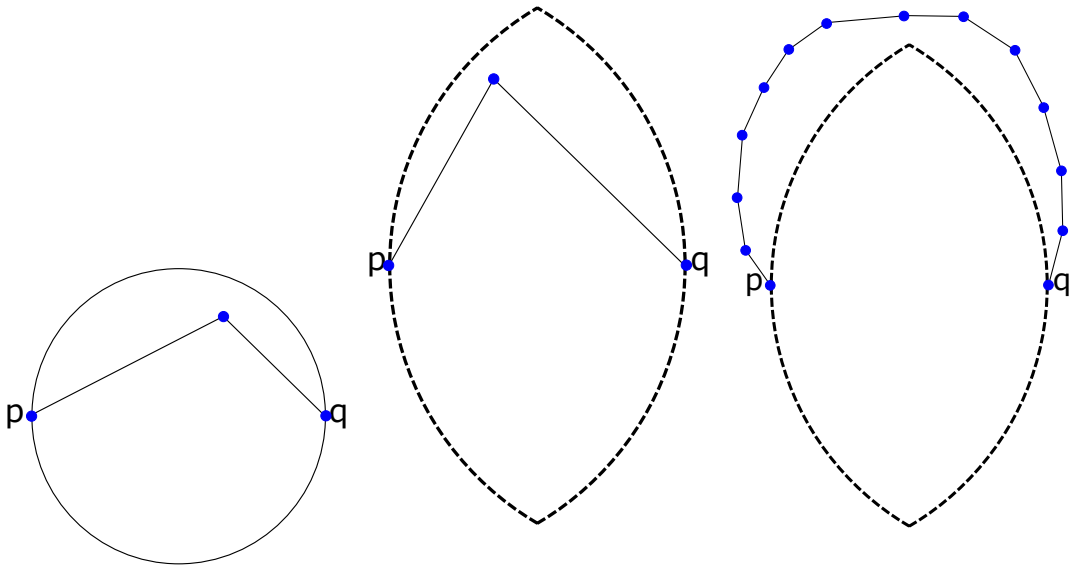
LBSN data can be seen as a set of positive locality samples collected over an imprecise region. In order to correlate these locality samples, neighbourhood graphs [5] can be used. Neighbourhood graphs infer edges between a set of given vertices based on pre-defined criteria. Graphical representation of a neighbourhood graph is called the neighbourhood skeleton [5]. Neighbour skeletons are used for studying the shape of a

point set. Connectivity and planarity are two properties necessary to make a graph representation intuitive. Connectivity refers to the ability to travel from one locality to any other locality using graph edges. Planarity is the property of not having overlapping edges when graph edges are projected on a plane.

In neighbourhood graphs, edge inferring neighbourhood can be defined per point, per point pair or per all points in the sample. Empty Region Graphs [6] study provides a comprehensive analysis of graphs that infer edges based on the absence of other vertices compared to a region surrounding the endpoints of the edge. When using an empty region as the evaluation criteria for edge inferring, it is important to note whether the boundary of the evaluation region is considered to be empty. If the boundary is considered to be empty then the evaluation region is referred to as a closed region and as an open region otherwise. It is a common practice to represent open and closed regions by dashed and solid lines when graphically representing a region.

Considering the inference of connections between locations, the Gabriel Graph (GG) [7] is a pioneering solution. GG was originally proposed as a tool for geographic variation analysis, to identify contiguous locality pairs. While discussing desirable properties of connecting regions for contiguous localities, authors emphasize that planarity and connectivity of inferred graph structure are crucial to making inferred connecting regions intuitive for geographic data. Authors use simple definition closed circle where given locality pairs are on the opposite ends of diameter as the empty region for GG as it ensures both planarity and connectivity of the inferred graph. Closed circle neighbourhood of GG is shown in Figure 2.1 (a). Later it was shown that the minimum spanning tree (MST) is a subgraph of GG and GG is a subgraph of DT [8].

In pattern recognition, Relative Neighbourhood Graph (RNG) [9] was first introduced as a mechanism that can infer a structure close to human perception of a point set. RNG achieved this due to the open lune shape it used for evaluating empty regions around two locations. Sketch of open lune neighbourhood is shown in Figure 2.1 (b). With the introduction of RNG, it was shown that MST is a subgraph of RNG and RNG is a subgraph of Delaunay Triangulation (DT). Given the DT of a point set, RNG can be found



(a) GG (b) RNG (c) SPG(2)

Figure 2.1: Examples of different neighbourhood graph on sample point sets

in linear time [10]. Since RNG is defined using distances between points, the definition of RNG can be extended to any dimension and for distance metrics other than Euclidean distance [9].

Urquhart Graph (UG) [11] was first proposed for fast construction of RNG. It was later proved that the UG is not always similar to RNG [12], but UG only differs from RNG by 2% maximally. Therefore UG can be seen as a faster method to approximate RNG [13]. We are combining the thought process behind UG creation and the Diversion neighbourhood of the Stepping Stone Graph to create the Diversion Graph.

β-Skeletons [14] is presented as a framework for neighbourhood graphs which infers edges based on a circle based empty regions. Authors present 2 categories of neighbourhoods. Lune based neighbourhoods are defined as the intersection of two circles and circle based neighbourhoods are defined as the union of two circles. With lune based definition, GG is $\beta = 1$ skeleton and RNG is $\beta = 2$ skeleton. β-Skeletons have been effectively used for reconstruction of curves given as point sets [15].

Delaunay Triangulation (DT) is a Triangulated Irregular Network (TIN) with many

benefits. It serves as a planar graph which has similar properties as the complete graph [16]. For this reason, it can be used as the starting graph for inferring many other planar graphs. Due to having a low spanning ratio and faster inferring it is used in many movement network analysis problems. Note that $MST \subseteq RNG \subseteq UG \subseteq GG \subseteq DT$.

Shortest Path Graph ($SPG(t)$) is first proposed as the base skeleton for inferring the boundary of an imprecise region, given as a positive locality sample [17]. The main idea of $SPG(t)$ is to infer the edges between points in such a way that the shortest path within the imprecise region roughly corresponds to the shortest path in the graph. In this work, authors discuss seven desirable properties of imprecise regions. When the weight of an edge is defined as its Euclidean length raised to the power of parameter $1 \leq t$, the edge is present in $SPG(t)$ if and only if it is the least weight path between its endpoints. Authors show using empirical results that $SPG(2)$ is better for delineating an imprecise region, compared to both KDE and GG. $SPG(2)$ performs better than GG and β-Skeletons because it compares the cumulative weight of the shortest path over inferred edges against the weight of the evaluating edge.

Presenting $SPG(t)$, authors highlight the connectivity of the resulting structure as one of the desirable properties of mechanisms used to analyze the boundary of an imprecise region. Open lune is the upper bound of the empty regions graphs to ensure a connected skeleton. Since RNG uses open lune as the empty region criteria, for the usage of delineating imprecise regions $SPG(2)$ should be compared against it. Figure 2.1 (c) depicts a situation where $SPG(2)$ perform better than RNG. In the discussion, the authors noted that the quality of result generated using $SPG(2)$ heavily depends on the selections for parameters. They have particularly highlighted outlier filtering which is an essential step to generate a quality result, as such an area. Later $SPG(t)$ skeletons with varying t are used as base structure for overlapping set visualization [18]. As a downside of using a global criterion, inferring algorithm results in undesirable running times such as $\mathcal{O}(n^2 \log n)$ to infer planar edges. Therefore generating $SPG(t)$ with varying t values is a computationally intensive task. Even though $SPG(t)$ provides a variable graph solution comparable to static RNG, due to increased running time of $SPG(t)$ creation algorithm

makes it less useful. Therefore we need to look at alternative graph solutions that exist with good properties of $SPG(t)$ and RNG, without being static or inefficient.

2.2.2 Spatial Grids for Movement Analysis

Spatial grids are useful and intuitive location data modelling technique useful for movement analysis. When modelling a location data set as a grid, total area where location points are spread are divided into smaller grid cells. These grid cells can be a static and uniform shape such as squares or rectangles, or variable and dynamic such grid cells based on location density. When using spatial grids for movement analysis, usually first the area under analysis is divided into grid cells and then movement patterns between grid cells are analysed.

In the operation of constructing popular routes from uncertain trajectories using RICK [19], analysed area is divided into a static grid of square cells. Next movement patterns between cells are analysed. Even though this approach generates comparable popular routes against other existing popular route finding methods, it is difficult to decide on a single grid size that is suitable for analysing all the data. Also based on the size of grid cells, the execution time of the solution varies. Selecting a smaller grid cells size improves the result quality, but at the same time increases the running time of the algorithm. For extraction of Geometries of Interest (GOIs) in [20], researchers have used static grids. Rather than analysing the movement between grid cells, they have linked the grid cells with similar locations where location posting entities have been stationary.

Another method for partitioning the analysis area is to use a variable grid. A popular method of identifying such grid cells is to use the Voronoi tessellation of a point set. To analyse movement patterns in large trajectory data sets this method is useful [21]. In this approach, first, the locations are grouped by proximity. Then, the Voronoi tessellation is created based on the group centroids. Next, movements between the Voronoi cells are analysed. Compared to the static grid cells, this method is more suitable for analysing location data sets with varying densities over the area. The Voronoi cell granularity can

be adjusted by changing the parameters used for grouping locations. The same approach of using the Voronoi cells of the location groups is used for analysing movement patterns using LBSN data [22].

When using grids to analyse movement, a set of locations are grouped into a grid cell. Therefore, movement analysis using grids usually present an aggregated view of location data. When using graphs directly on location data, movement analysis can be performed at the most descriptive granularity. Due to this reason, graphs are desirable when analysing movement within a smaller region or analysing data with less density. However, grid based methods are useful for analysing a larger area in a shorter amount of time or to analyse high density data.

2.3 Movement Analysis Using Location Data

Before discussing movement analysis using LBSN data, we will discuss movement analysis using location data in general. Location data can be collected using many different mechanisms including GPS enabled devices [23], cell phone record location data [24], Wifi access records, public transport card usage records [25], etc. These location data can be analysed to reveal different aspects of public movement patterns. Public movement analysis is an important use case where movements of the general public in a geographical area is analysed in various granularities.

GPS location data collected using machine-generated methods can be processed to uncover useful information. Geometries of interest (GOI) [26] is one such information type where area geometries of points of interests (POIs) are extracted using GPS trajectory data. To do this, locations of trajectories where users have stayed for more than a specific time are aggregated and outlier locations are filtered out using spatial indices. RoadRunner [27] is another system proposed to infer the road network using GPS trajectories, that is the connecting network between GOIs. Combining these approaches, researchers can uncover the popular areas of public visit and the travel network they use

to move from one area to another.

A research conducted on extracting regions based on what those regions means to users and analysing movement between these regions is available at [28]. Such meaningful regions are called semantic regions. This information is useful in characterizing users and in making friend suggestions based on semantic movement patterns. Location transition without semantic information can also be mined using GPS data [29]. Information uncovered in this analysis can be used for tour recommendation and to increase the usability of location based services such as congestion aware routing. On a similar note, machine-generated trajectories from vehicles can be used to understand urban recurrent congestion evolution patterns [23].

We can also observe that graph theory is used in analysing location data. [30] presents a graph-based approach to vehicle trajectory analysis. Proposed approach merges GPS trajectories based on their closeness. Rather than considering only spatial closeness researchers have developed a method to merge locations in the trajectory if trajectories passing through these locations are similar. This approach operates in two stages - reduction stage where trajectories are reduced to representative trajectories and are aggregated, and a second stage where partitioning and regionalisation occur where relationships between all trajectories are analysed to detect interesting regions and to define trajectory clusters connecting them. Neighbourhood graphs have been used to route planning of autonomous agents [31]. In this work *r*-limited Delaunay graph is successfully used in a robust rendezvous planning for several agents. Animal movement tracking can be performed using neighbourhood graphs as a base [32].

Other than GPS data, triangulated mobile phone signals can be used to process public movement [24]. Call Detail Records with locations can be used to profile user movement and predict their network usage [33]. The extracted information can be used for planning network bandwidth allocation based on user movement. Using locations obtained from cellular network data, [34] examines how diseases circulate around a country as people move between regions. Researchers uncover that restricting mobility does not delay the spread of the disease, but an information campaign relating to preventive mea-

sures might be an effective countermeasure against the spread. However, Call Detail Records are often characterized by spatial and temporal sparsity [35]. By using regularity in human movement mobility analysis results using Call Detail Records can be improved. Conducting a study on the limits of predictability in human mobility, Song et al. reveal that 93% potential predictability in user mobility using mobile phone records [36]. Also, they note that this predictability is less variable and independent of the distance users cover on a regular basis. Similar study [37] conducted on trajectories of 100,000 anonymized mobile phone users whose position is tracked for a six month period, come to a similar conclusion that the humans follow simple reproducible patterns despite the diversity of their travel history.

In the study [4], authors aim to answer the question "At what frequency should one sample individual human movements so that they can be reconstructed from the collected samples with minimum loss of information?". They find that the average error incurred by trajectories reconstructed from periodic samples scales linearly with the constant sampling interval. across their user base; depending on the individual, the error typically grows 1 to 4 meters when adding one minute to the inter-sample time. They propose that their finding is useful in energy-efficient mobile computing, location-based service operation, active probing of device position in mobile networks, and trajectory data compression.

2.3.1 Movement Analysis Using LBSN Data

As mentioned in Chapter 1, LBSN data mainly contains four different types of information relating to an LBSN post. They are namely, posting user's user id, authored content, posted time and posted location. Combining these data items researchers have developed mechanisms to analyse user mobility [38, 39], analyse crime distribution over area [40, 41], analyse disease infection risk [42], hazard mitigation [43] and situational awareness [2]. Figure 2.2 shows the typical architecture of LBSN processing systems. First, LBSN data is collected using APIs of LBSNs. Next, this data is pre-processed to cre-

ate the structure necessary to process LBSN data and to enrich LBSN data with additional information such as user profile information. The main processing of the systems happens after that, where high level information such as sentiment and semantics of LBSN post and their relevance to locations are extracted. Before releasing extracted crude information to the application layer, they are post processed to convert them into more understandable formats required by the application layer systems. Finally extracted information is published to the application layer to be used by the respective users. In this section, we will discuss a few such systems and compare them.

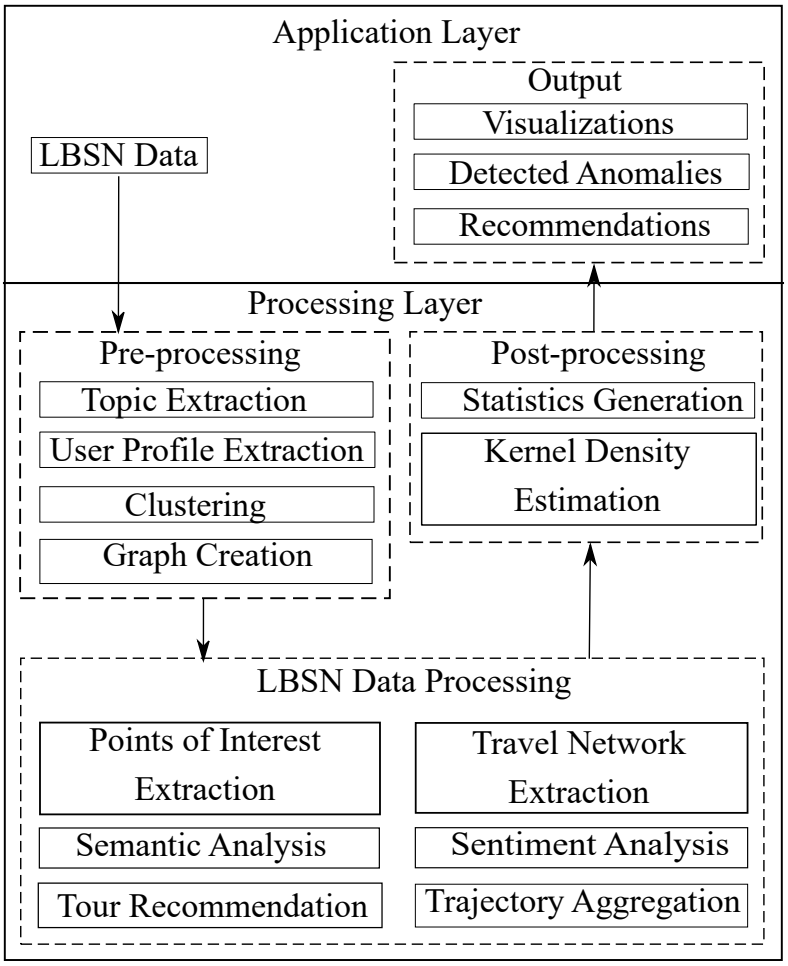


Figure 2.2: Example architecture of LBSN data processing systems.

2.3.1.1 Characteristics of LBSN Data

First, we are going to look into the characteristics of LBSN data. Study [44] analyses approximately 2.3 million photos posted on Instagram. However, the actual data was collected via Twitter posts of Instagram (a popular photo sharing service) users. Authors point out the highly unequal frequency of photos sharing, both spatially and temporally. High coverage for some cities where the population is dense is observed. When comparing the total number of photos shared over the analysed space, most of the areas have few shared photos, while there are few areas with hundreds. Authors note that a power law describes well this situation. When investigating the frequency in which users share photos on Instagram, we can observe bursts of activity and long periods of inactivity, such that "there are times when many photos are shared within a few minutes and there are times when there is no sharing for hours". Also, fewer users contribute more photos while more users contribute fewer photos. Authors observed that the frequency of photo sharing is spatio-temporaly very unequal and correlated with routine human activities. This data is useful in identifying POIs in a city, POI to POI transition analysis, and to analyse the Vibe of POIs. Another study [45], analyses check-in data published on several services such as Foursquare, Gowalla, Echofon, and Gravity. This study points out that most of the shared locations points are concentrated towards cities and highways. Also, they analyse user displacement and notice that user displacement compared against the total number of check-ins that exhibit that displacement follows a power law. This can be attributed to most users posting less to LBSN and less number of users post with high frequency to LBSN. These same characteristics of LBSN data can be observed in other research that study distribution of LBSN data [46, 47, 1, 48, 49]. In our data sets, we observe the same distributions, where posted locations are concentrated towards cities. Also, few of the LBSN users contribute most of the data while most of the LBSN users contribute less amount of data.

Since most of the publicly collected LBSN data is concentrated towards cities, data tends to be biased towards urban perspectives at the expense of rural ones [50]. This can be shown by comparing the spatial distributions of county-level summary statistics

with the spatial distribution of the urban/rural population ratio of countries. For example authors highlight that, popular LBSN data source Foursquare affords the study of urban mobility patterns, not mobility patterns in general (or rural mobility patterns specifically) [51, 52].

2.3.1.2 Movement Analysis Use Cases of LBSN Data

2.3.1.2.1 Event Analysis: Event detection and analysis is one such use case supported by LBSN data. In LBSN data, an event can be seen as a real world phenomenon that motivates LBSN users to get together to the physical area and post to the LBSN with an increased frequency with a set of keywords identifying the event. Event detection can be done by clustering the LBSN data stream or analysing the movements of users. Lee et al. [53] propose an approach to analyze public movement regularity and use that to detect events, utilizing publicly available LBSN data from Twitter. The proposed approach can be viewed as a two step process. Firstly, the analyzed area is partitioned. Next, user movements with respect to partitions are analyzed using the timestamps. In order to partition the area first, the LBSN posts are clustered using K-means and partitions are created around cluster centres. They have selected this method of partitioning mainly due to simplicity. However, they have highlighted the need for better partitioning mechanism. Authors present a methodology to measure the regularity of crowd behaviour by analyzing movement within the same partition. Focusing on the movement between different partitions, they present a methodology to discover irregularities in user behaviour [22]. They show the effectiveness of their method by analyzing areas influenced by popular events. Authors have pointed out the differences in posting frequency in different areas and developed a quadtree based data querying method to effectively query data from Twitter. Event detection based on analysing anomalies in LBSN data stream can be found in [54, 55, 56, 57]. Clustering based event detection [58, 59, 60] require researchers to clusters the LBSN data across multiple different dimensions including time, space and text. LBSN based event detection has been extended to create an interactive news portal in [61]. With these events and news extraction methods, the importance of spam detec-

tion also increases. Therefore research on spam detection is a trending research area [62].

2.3.1.2.2 Gaining Situational Awareness: Mostly social networks are used as a communication tool in the times of hazards and natural disasters [63, 64, 65]. However, there are research conducted on using LBSNs to gain situational awareness in a time of a disaster. In [2] MacEachren et al. present results of a survey regarding the usage of LBSN data in crisis management along with a prototype system that can be used to analyze LBSN data. The survey is conducted with the participation of crisis\emergency management professionals. Survey participants highlight the utility of map based visualizations for the analysis of LBSN data relating to crisis events. As for the data to be visualized, participants emphasize on locations where content relating to the crisis are generated and localities travelled by the generating users. Following a scenario-based design development approach, authors present a prototype system named SensePlace2. Using this system LBSN analysts can query data by manually providing a topic string. The system visualizes related LBSN post distribution as a grid based heatmap overlay of a geographic map. Discussing age and opinion based biases in Twitter data, authors highlight the necessity of incorporating other data sources for analysis. Chae et al. [3] propose a visual analytics approach for public behaviour analysis. Moreover, the authors emphasize on complexities associated with LBSN data. Initially, authors have tried to reveal public movement flows during the disaster events but, report that the movement patterns were too complicated to find meaningful flows due to the movement randomness and the visual clutter of the flows. They have used geospatial heatmap in their proposed approach, highlighting its usefulness for visualizing post distribution. To provide visualization of both spatial and temporal aspects of data, authors have suggested displaying hexagons on a geographic map colour coded by generated timestamp. Using Twitter data collected during Hurricane Sandy and a Tornado in Oklahoma City, they have discussed the effectiveness and limitations of the proposed analysis approach. The increase of tweets near areas affected by the above disasters are recognizable in the heatmap overlay provided. Authors emphasize that they did not observe any abnormal patterns of user distribution associated with both case studies. As future directions of this work, user flow analysis

and improving visualizations are highlighted.

2.3.1.2.3 Sentiment Analysis: With the rise of LBSN, usage of them in election campaigns and political endeavours is increasing [66]. This opens up new avenues of research such as election result prediction. The technique used to analyse LBSN data for election result prediction is known as sentiment analysis, where positivity or negativity related to the subject is extracted from voluntarily generated data. LBSN Twitter is used as a platform for political deliberation[67], and the number of tweets reflects voter preferences and comes close to traditional election polls. Election results predicted using LBSN data can be used to supplement the results obtained by surveying [68]. However, results from LBSN data analysis should be used with caution due to the aforementioned biases in LBSN data. One solution is to use LBSN data analysis results in correlation with census data [69].

2.3.1.2.4 Semantic Analysis: Similar to the application presented in the location data analysis section, LBSN data can be used for semantic analysis. This use case focuses on finding the meaning of the place users are visiting. In order to do this some LBSNs have categorized locations according their own taxonomy [70, 71]. When such a taxonomy is not present or such information are missing from a place, techniques from information retrieval are used to associate semantic information related to that place[72, 73, 49]. These location semantics can then be used to reveal a user's semantic trajectory [74]. Based on this semantic trajectory we can analyse user similarity, even though users are in different places of the world. Semantic trajectory similarity can be used to make friend suggestions and location(activity) suggestions based on other users semantic trajectories [75].

2.3.1.2.5 Recommendation Systems: LBSNs can be effectively used to develop recommendation systems as user travel patterns along with voluntarily generated content is available. The first step of creating a recommendation system is to extract patterns from LBSN data. This is usually done by spatially clustering posted locations and find-

ing frequent movements between those clusters [76, 77]. These extracted patterns can be used to create travel itineraries [78, 79, 80, 81]. Note that this travel itinerary creation problem is NP-Hard. Rather than planning an itinerary to visit tourist attractions, these methods can be further extended to recommend preferred paths [82] and to recommend preferred events to visit [83].

2.3.1.2.6 Aggregating information from different users: Since we propose methods to aggregate information from different users, let's look at an available work on aggregating information. RICK is a system that proposes a methodology to aggregate uncertain trajectories to construct popular routes to traverses a given set of locations [19]. In this work, timestamp ordered LBSN post sequence of a single user is considered as the trajectory. Due to the voluntary generation of LBSN data, posting frequencies tend to be low and irregular for most users. Therefore, it is hard to infer the popular routes from a single user trajectory. From a bird's eye view, the proposed trajectory aggregation mechanism has two steps. First, the analyzed area is divided into a static grid and densely populated grid cells are identified. Next, the posts' timestamps are used to calculate moving patterns between grid cells. The effectiveness of RICK is evaluated by comparing the inferred routes against the existing method MPR [84] and against well-known real-world routes. RICK requires several parameters to be passed to get the accepted results and the system's execution time increases with the data set size and the number of grid cells. Due to these reasons, developing a system using RICK to analyze LBSN data for real-time analysis is a challenging task. Hence, RICK can be improved by adding the capability of autonomously detecting some of the parameters. However, RICK's methodology for aggregating uncertain trajectories is useful for any other type of public movement analysis as well because irregular frequencies of LBSN posts is a common issue in LBSN analysis [57, 85, 86].

2.3.1.2.7 LBSN Users as a Sensor Network: With the increased affordability of GPS enabled mobile phones and the creation of LBSN, there is an opportunity to use LBSN users as a sensor network to sense various aspects of public's day-to-day lives [87]. This

approach has been used to analyse the distributions of noise experienced in cities [88, 89, 90], to map smell of the different parts of the cities [91], to track the alcohol consumption in regions [92], to analyse unemployment status of the public [93] and to analyse crime distribution over area [40, 41]. Most of these works use geo-spatial heatmaps to visualize and analyse the distribution of a particular type of LBSN content. Heatmaps are usually generated using mathematical techniques such as kernel density estimation. In order to identify boundaries of the necessary regions, graph based approaches may be better suited than heatmaps [17]. Also, LBSNs need to come up with a suitable incentive mechanism to motivate users to sense and publish data [94, 95].

2.3.1.3 LBSN Data Availability

Most LBSN provide only a small fraction of data for research purposes [96]. Therefore, it is important to verify the data density necessary to extract a meaningful result when using LBSN data to infer crowd movement patterns. Since LBSN data contain both spatial and temporal data about the user when a post is generated, methodologies used to process GPS trajectories can be applied for movement analysis. Such methodologies can be found in areas of animal movement analysis [21], vehicle trajectory clustering [97, 98] and inferring popular routes [84]. However, most of the available mechanisms need dense spatial data availability near the travel network and periodic location publishing by source. Due to the inherent properties of LBSN data such as data sparsity and irregular posting frequencies, it is hard to extract a meaningful result using these methods.

In summary, LBSN is a great source of information for user movement analysis. Most of the analysis approaches available can be viewed as two steps processes that analyze spatial and temporal aspects separately. The result of the two analysis steps is presented as the resulting movement pattern. To provide an intuitive graphical representation of inferred movement patterns, a geographic map based visualizations can be highlighted but, it is difficult to develop methodologies to infer meaningful user flows due to the voluntary generation of data. Considering the amount of information that can be uncov-

ered, developing methods for flow pattern mining from LBSN is an important research area. Investigating mechanisms to analyze sparse and irregular LBSN data, providing better methods to partition geographical areas based on LBSN posts and investigating on information that can be visualized using temporal and sentimental aspects of LBSN data are some of the areas that need to be further explored.

2.3.2 Group Movement Analysis

We can uncover important cumulative patterns by combining location information posted by multiple users. There are several works that combine the information generated by multiple individual users to generate a group. Flock analysis [99, 100] is a popular technique in this domain. Flock analysis aims to identify sets of location posting entities that remain within close proximity from each other. In flock analysis sources within the close proximity between each other are identified by making sure they remain within a spatial disc. Example moving flock is shown in Figure 2.3 within the circular area. Therefore, flock analysis can be considered a clustering operation on moving objects. Even though flock analysis is originally proposed as an off-line processing technique for artificially generated location data, methodologies were developed later to process spatio-temporal data and identify flocks on-line with vehicle and animal movement data [101] and applied to other location data sources such as pedestrians [102]. Also, there are other techniques that focus solely on finding clusters of moving objects [103, 104].

Due to the rigidity of spatial disc used in flock analysis, some of the entities travelling together will not be detected as belonging to the same flock. In Figure 2.3, the user outside the circle is intuitively moving with the other three users. However, the fourth user is not considered as a part of the flock due to the used disc. Making the disc larger will make flock absorb other users who are not in the flock. This problem is known as the lossy-flock problem. To mitigate this issue convoy [105] analysis is proposed. Rather than considering a shape like a disc to cover the travelling entities, convoy analysis considers a densely packed travelling entity clusters. This approach of density connected

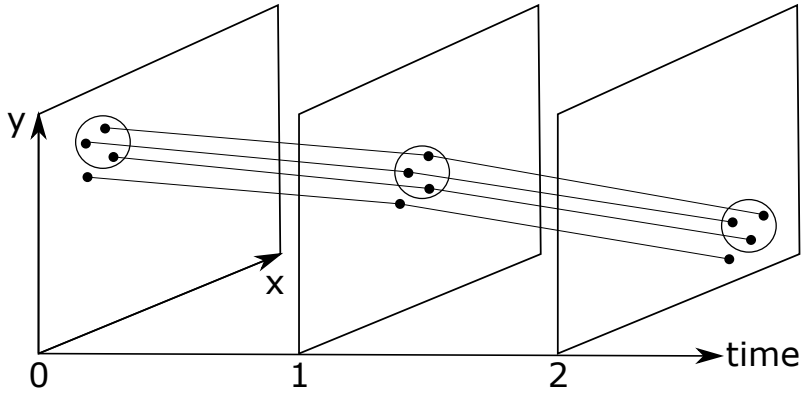


Figure 2.3: Example of a moving flock. Small black dots represent users. Circle that contain users is the sptial disc used to identify the flock.

group detection is preferable compared to group detection based on a static shape, in a situation where moving groups need to be tracked such as in emergency evacuation scenarios [106].

Trajectory clustering [97] is another popular technique used to combine information from different sources to create aggregated information. Focusing on a broader view of data, trajectory clustering aims to group trajectory traces based on their similarity and come up with representative trajectories for clusters of similar trajectory traces. All of the above techniques perform well with machine-generated dense data. It is hard to apply these techniques in LBSN domain due to the data sparseness and irregularities.

When monitoring group movement we have to look at the existing work on tracking moving objects. [107] provides an introduction to tracking multiple objects distributed over a geographical region using a wireless sensor network. The main difference in our study is that rather than sensing moving users, we are depending on moving users to give their locations voluntarily. [107] recommends developing a collaborative group abstraction to map the dynamically formed collaborative groups. We use this approach to handle complexities when tracking our moving groups, by rather than tracking individual users we track user groups to predict their next location.

The Kalman filter is a versatile method that can be used to combine several signals and reveal a more useful combined signal. Kalman filter works in two distinct steps,

namely the prediction step and update step. The output of one step can be fed in as input to the other step. Using these two steps the Kalman filter can predict the state of a process using the previous step's state and current state's measurements. By considering location data posted by moving entities as location signals we can use the Kalman filter to process them. There are many extended version of the Kalman filter more suitable for specific purposes [108]. The main thing when using the Kalman filter is to create a suitable model to capture the dynamics of the process we are analysing.

Multiple object tracking method using the Kalman filter is presented in [109]. In this approach, each object to be tracked is modelled separately from each other and applied Kalman filter for tracking. To address occlusion problem (several tracked objects merge into a single object) authors have introduced a new tracking object for merged object and continue tracking previous objects also. When merged object splits into several objects again, continuously tracked previous objects are matched to newly split objects and merged object tracking is stopped. Using image sequences depicting human and vehicle movement, the authors demonstrate the effectiveness of the proposed method. However, our LBSN data are too sparse to apply this method directly. Therefore we have developed a new model for tracking LSBN user groups.

Another method for tracking multiple moving objects is proposed in [110]. Unlike the model in [109], this model combines all the features relating to all objects into a single model. And instead of typical Kalman filter, authors employ unscented Kalman filter for tracking objects. Unscented Kalman filter is developed by applying the unscented transformation to pick a minimal set of sample points around the mean. This unscented transform makes the unscented Kalman filter applicable to non-linear systems. However, due to these additional processes unscented Kalman filter results in higher time complexity than Kalman filter. Using experimental results obtained by analysing human movement, authors demonstrate the proposed methodology detects and tracks multiple moving objects and works well in occlusion situations.

Note that the Kalman filter is a well-known filter which makes simple assumption of the process that its modelling in order to simplify the operation. Due to its simplicity

and the ease of explanation we have used the Kalman filter for developing our group movement prediction contribution. There are other filters such as the Weiner filter [111] which is the predecessor of the Kalman filter and the particle filter [112] along with many other filters that can be used for multiple signal filtering. General structure of our solution the Group Kalman filter will work with any of these multiple signal processing filters. However, based on the filter used for processing, inputs, outputs and process modelling needs to be updated to reflect the internal mechanics of the filter. For example using the Weiner filter will result in usage of continuous signals instead of discrete signals. Due to these complexities associated with other available filters we used the Kalman filter as the multiple single processing solution in our solution.

2.4 Summary

We started this chapter introducing modelling of location data for movement analysis. Modelling techniques of neighbourhood graphs and spatial grids were discussed in this section. In the neighbourhood graphs subsection, some of the well known neighbourhood graphs types were discussed. Most of the introduced graphs infer a rigid structure between analysed locations. Graphs that can vary the structure based on parameters are more useful as they can be configured as needed. Both our proposed graph structures are configurable based on a single parameter. In their relevant chapter, we mathematically relate newly proposed graph structures to the existing graph structures. Techniques used in the proof of previous work were very useful when writing mathematical proofs relating our proposed graph structures to existing work. Spatial grids divide the analysed area into grid cells and represent locations within a cell as a group. Due to this reason spatial grids usually provide an aggregated view of location data. At the end of this section we discuss the suitability of neighbourhood graphs and grids for analysing location data in different situations.

Next, we shifted our discussion to movement analysis using location data, and presented different sources of location data for movement analysis. In this section, we dis-

cussed different information that can be uncovered by processing location data such as areas of interest, movement network used to travel between these areas, the semantics of these locations and user mobility. Also, we presented how graphs are used as an effective tool to process location data. Our contributions the Stepping Stone Graph and the Diversion graph can be used as a strong base to analyse location data. As these graph structures are created with the focus of capturing movement related information, they are useful in all the use cases presented in the location data analysis section.

In the next subsection we discuss related literature on movement analysis using LBSN data. We started this section by introducing characteristics of LBSN data such as sparsity and irregularity of posting observed in other studies. Despite these negative characteristics LBSN data can be used to analyse events, to achieve situational awareness, sentiment analysis and semantic analysis of various aspects of public life. We concluded this section with reasoning why techniques applicable to machine-generated location data are not yielding good results on LBSN data. Newly proposed graph structures in later chapters are specially formulated to work with sparse and irregular data from LBSNs. As shown in applications in later chapters, these graph structures are suitable to act as a base for processing LBSN data. Furthermore, they can be used as a base structure to visualise various aspects of LBSN data.

In the last subsection, we reviewed literature relating to combining location data from multiple moving sources. We discussed flock analysis, location data stream clustering, convoy analysis, trajectory clustering and the Kalman filter based multiple object tracking methods. Our contribution, the Group Kalman filter is more similar to the convoy analysis than flock analysis in terms of the way groups are detected and analysed. However unlike both convoy and flock analysis we are not only trying to identify the LBSN users are moving as a group, but also we are modelling the group's movement to predict the next location of the group. This section shows the importance of combining location data and provides the building blocks for our third research question group movement prediction. In the following chapters, we propose new data structures and algorithms to process LBSN data. We try to address some of the aforementioned use cases using our

newly created data structures and algorithms. We will be referring to the information from this chapter as necessary.

Chapter 3

Stepping Stone Graph For Movement Analysis

This chapter is derived from:

- Sameera Kannangara, Egemen Tanin, Aaron Harwood, and Shanika Karunasekera. "Stepping stone graph for public movement analysis." In *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pp. 149-158. 2018.
- Sameera Kannangara, Egemen Tanin, Aaron Harwood, and Shanika Karunasekera. "Stepping Stone Graph: A Graph for Finding Movement Corridors using Sparse Trajectories." *ACM Transactions on Spatial Algorithms and Systems (TSAS)* 5, no. 4 (2019): 1-24.

3.1 Introduction

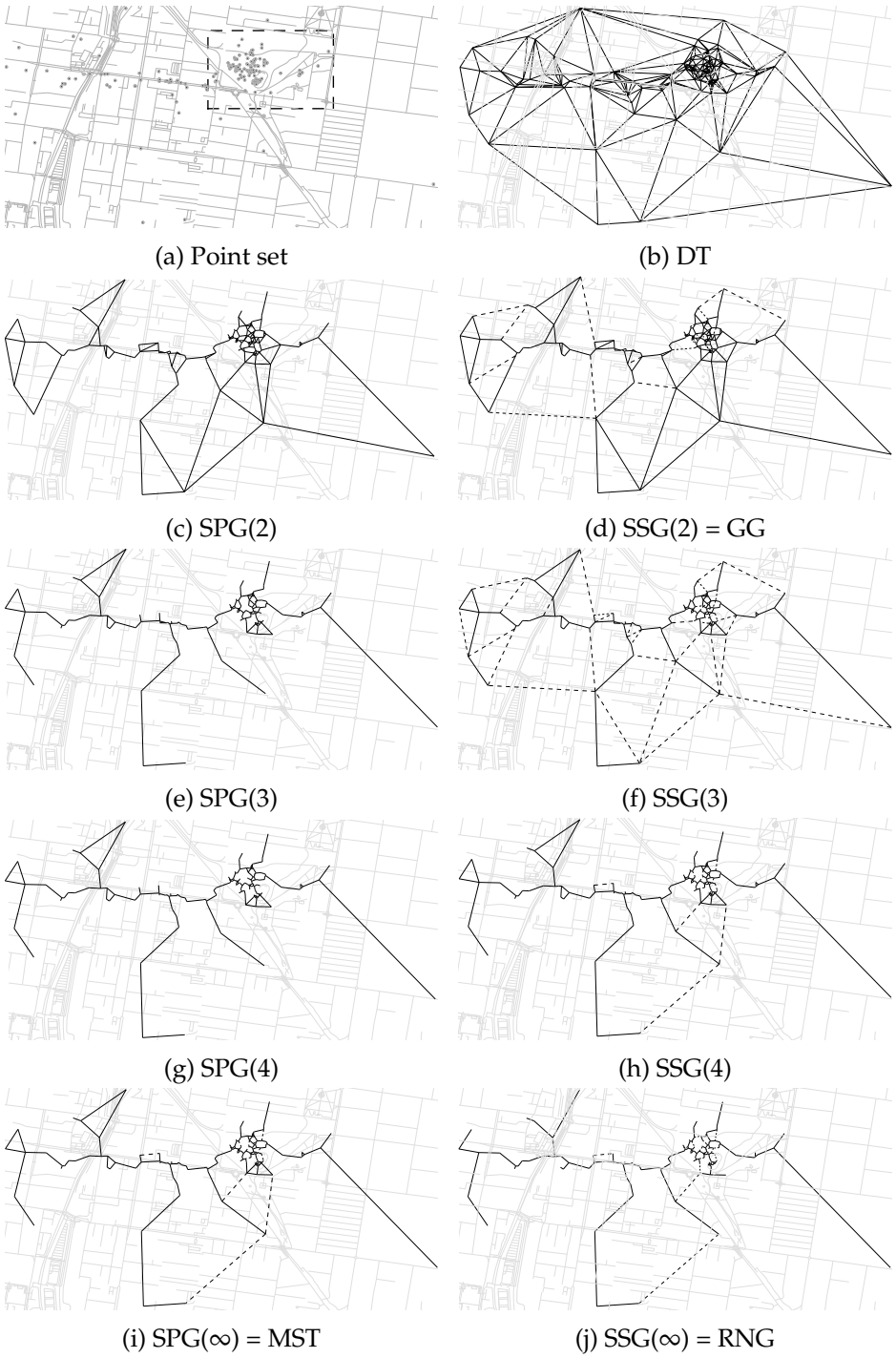
Often spatial data is provided as a distinct point set in LBSN domain. Graphs, in which the relationships between points can be represented as edges, are useful for representing and processing such point data. Neighbourhood graphs infer edges based on the relationships defined on the point set provided for analysis. Distance based connected neighbourhood graphs are useful for inferring and processing movement networks be-

tween a set of distinct points[31].

Delaunay Triangulation (DT), Minimum Spanning Tree (MST), Gabriel Graph (GG) and Relative Neighbourhood Graph (RNG) are a few distance based connected neighbourhood graphs that are used to infer and process movement network between point data. Edges inferred using these graphs do not vary based on parameters. Therefore, these graphs infer a static movement network between the point set. Figure 3.1 (a) shows a point set. Figure 3.1 (b) (d) (i) (j) represent DT, GG, MST and RNG skeletons, respectively. Neighbourhood skeleton which is obtained by connecting endpoints of the inferred edges using straight line segments is referred to as the geometric realization of a neighbourhood graph. Therefore, variations of neighbourhood skeletons are widely used to represent the geometric shape of a point set.

As opposed to static graphs, variable graphs have the ability to vary the inferred edge set based on parameters. Hence variable graphs are more versatile[113]. The Shortest Path Graph ($SPG(t)$)[17] is such a variable graph proposed with the idea of inferring edges between a point set such that the shortest path taken over the inferred edges will roughly align with the shortest path taken over the imprecise region represented by the point set. Characteristics of the inferred edge set vary based on a single user provided parameter $t \geq 1$. $SPG(t)$ become MST of the given point set as $t \to \infty$. $SPG(t)$ imposes more structure when inferring edges, due to its distance based global criteria. Since $SPG(t)$ imposes more structure when inferring edges, it is useful for use cases such as delineating imprecise regions[17] and overlapping set visualization[18]. Figure 3.1 (c), (e) and (g) represent skeletons generated using $SPG(t)$ with configuration parameter set to 2, 3 and 4, respectively. As the value of the configuration parameter t increases, the number of edges in $SPG(t)$ decreases. Therefore, paths taken over $SPG(t)$ edges become much longer compared to the direct distance between points as the value of configuration parameter increases.

In this chapter, we propose a new type of graph, which we refer to as the Stepping Stone Graph ($SSG(d)$). Like $SPG(t)$ converge to MST, $SSG(d)$ converges to RNG. RNG provides a structure more similar to human perception of a point set compared to



(a) Point set (b) DT

(c) SPG(2) (d) SSG(2) = GG

(e) SPG(3) (f) SSG(3)

(g) SPG(4) (h) SSG(4)

(i) SPG(∞) = MST (j) SSG(∞) = RNG

Figure 3.1: (a) A point set (with a rectangle in dashed lines indicating the location where a music festival held) with a set of connected neighbourhood skeletons. Dashed lines in bottom right hand side figures indicate additional edges inferred by the $SSG(d)$ compared to $SPG(t)$.

MST[114]. Therefore, structure inferred by $SSG(d)$ is more perceptually relevant compared to $SPG(t)$ when configuration parameters are equal. $SSG(d)$ infers edges between a point set using a distance based local criterion to mimic the movement network, due to which, $SSG(d)$ imposes less structure when inferring the edges. This local criterion varies based on a single user provided parameter $d \geq 1$. For all endpoint pairs, the value of d indicates the preference of inferring a longer alternative path with less distance between all point pairs on the path compared to the direct distance between the endpoint pair. Similar to $SPG(t)$, as the value of the configuration parameter d increases, the number of edges in $SSG(d)$ decreases. Therefore, the length of paths taken over $SSG(d)$ increases compared to the direct distance between locality pairs. Due to the local evaluation criteria, $SSG(d)$ infers more alternative paths compared to the $SPG(t)$. This effect makes $SSG(d)$ more suitable for analysing movement network between a given point set. In Figure 3.1, figure pairs (c) and (d), (e) and (f), (g) and (h), (i) and (j) depicts $SPG(t)$ and $SSG(d)$ with configuration parameter set to value 2, 3, 4, and ∞, respectively. Additional edges inferred by $SSG(d)$ which are not inferred by $SPG(t)$ are highlighted using dashed lines in Figure 3.1 (d) (f) (h) and (j).

We evaluate the utility of $SPG(t)$ and $SSG(d)$ for inferring a movement network between a set of related localities filtered from publicly available LBSN data. $SSG(d)$ can infer a planar edge embedding in $\mathcal{O}(n)$ time, whereas $SPG(t)$ takes $\mathcal{O}(n^2 \log n)$. Therefore, $SSG(d)$ can be computed faster compared to the $SPG(t)$ due to its local evaluation criteria. From our experiments, it is evident that $SSG(d)$ has a low and stable spanning ratio[115] compared to $SPG(t)$. Thus, $SSG(d)$ is more effective for inferring and processing the movement network compared to $SPG(t)$. We performed experiments on analysing movement network of locations influenced by an event and movement network between localities posted within a city. In both experiments, $SSG(d)$ inferred more alternative paths similar to real world paths compared to $SPG(t)$.

3.2 Stepping Stone Graph

We consider all location points provided for the analysis as stepping stones that need to be linked in a traversable manner. So we focus on connecting these stepping stones such that a travelling entity can find a path between two locations.

3.2.1 Definitions

We construct $SSG(d)$ in the form of an undirected graph $G(V, E)$ where $V \subseteq \mathbb{R}^2$ represents a given point set and E represents inferred edges between the points. An edge between two endpoints $p, q \in V$ is represented as $pq \in E$. Length l_{pq} represents the Euclidean distance between two points.

First, we define the *Diversion Neighbourhood* in Euclidean space, which is the main construct necessary for defining $SSG(d)$, an area between two points which varies based on a single parameter $d \geq 1$.

Definition 3.1 (Diversion Neighbourhood)**.** *For $p, q \in \mathbb{R}^2$, the Diversion Neighbourhood of pq at $d \in \mathbb{R}$ such that $d \geq 1$, denoted $DN(pq, d)$ or simply $DN(d)$, is defined as the region:*

$$DN(pq, d) = \left\{ z \in \mathbb{R}^2 \text{ such that } l_{pz}^d + l_{zq}^d \leq l_{pq}^d \right\}. \tag{3.1}$$

A graphical representation of $DN(pq, d)$ for varying d is shown in Figure 3.2. As the figure shows, $DN(d)$ increases from a straight line connecting the endpoints to the open lune neighbourhood between the endpoints. Note that we use $\leq$ instead of $<$ in Equation 3.1, making $DN(d)$ a closed region. Next, we define the Stepping Stone Graph using $DN(d)$.

Definition 3.2 (Stepping Stone Graph)**.** *For $V \subseteq \mathbb{R}^2$, the Stepping Stone Graph of V at $d \in \mathbb{R}$ such that $d \geq 1$, denoted $SSG(V, d)$ or simply $SSG(d)$, is defined as an undirected graph with*

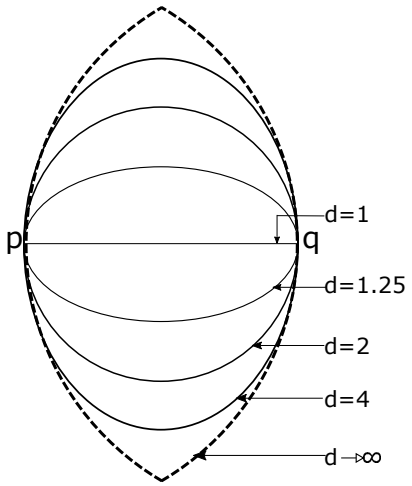


Figure 3.2: Graphical representation of the Diversion Neighbourhood

points V, such that for each point pair $p, q \in V$:

$$pq \text{ is an edge of } SSG(V, d) \text{ iff } DN(pq, d) \cap V \setminus \{p, q\} = \emptyset.$$

Intuitively, if $DN(d)$ for a given d and endpoint pair contains no other points then $SSG(d)$ will have an edge between that endpoint pair.

Since $DN(d)$ becomes an open lune as $d \to \infty$, $SSG(d)$ is a connected graph, due to theorem 25 in [6]. After creating $SSG(d)$, edges longer than a given threshold can be filtered out to accommodate travel distance requirements of a travelling entity [31]. The diversion neighbourhood is effectively a local criterion when selecting edges in $SSG(d)$, as opposed to the global criterion in $SPG(t)$ and therefore in some cases, $SSG(d)$ provides more alternative paths between a point set compared to $SPG(t)$, and $SSG(d)$ never provides less alternative paths than $SPG(t)$ when $d \leq t$.

By definition of $DN(d)$, as d increases, the evaluation area increases and that in turn reduces the number of edges in the final graph. Therefore the travel distances over the shortest paths taken from $SSG(d)$ between all endpoint pairs monotonically increase. As the path distances increases with the number of edges in a path, the number of points

on the shortest paths monotonically increases. Therefore, as d increases, the average distance to be travelled between a point pair over the shortest path taken from $SSG(d)$ monotonically decreases. Hence, $SSG(d)$ furnishes alternative paths where, for a given endpoint pair, each of the path edges in the alternative path is always shorter than the direct path between the endpoint pair.

3.2.2 Stepping Stone Graph Properties

As d increases, the area covered by $DN(pq, d)$ increases symmetrically around both the perpendicular bisector and the direct connecting line of pq. This leads to $SSG(d')$ being more restrictive than $SSG(d)$ for $d' > d$. Therefore, as d increases, by definition the number of edges in $SSG(d)$ monotonically decreases. Hence, the following theorem results.

Theorem 3.1. *For* $1 \leq d \leq d', SSG(d') \subseteq SSG(d)$

Proof. Define the *edge weight* of pq with respect to d' as $l_{pq}^{d'}$, for some $d' \geq 1$. Assume that for all $z \in V \setminus \{p, q\}$, $l_{pz}^{d'} + l_{zq}^{d'} > l_{pq}^{d'}$. In this case, pq is an edge in $SSG(d')$. Now we show that for $d \leq d'$, pq is also an edge in $SSG(d)$. Let us write $d = d'\,\epsilon$ where $\frac{1}{d'} \leq \epsilon \leq 1$. Then we need to show that:

$$
\begin{aligned}
l_{pz}^{d'\epsilon} + l_{zq}^{d'\epsilon} &> l_{pq}^{d'\epsilon} \qquad (3.2)\\
\frac{l_{pz}^{d'\epsilon} + l_{zq}^{d'\epsilon}}{l_{pq}^{d'\epsilon}} &> 1\\
\left(\frac{l_{pz}}{l_{pq}}\right)^{d'\epsilon} + \left(\frac{l_{zq}}{l_{pq}}\right)^{d'\epsilon} &> 1\\
\left(\left(\frac{l_{pz}}{l_{pq}}\right)^{d'\epsilon} + \left(\frac{l_{zq}}{l_{pq}}\right)^{d'\epsilon}\right)^{\frac{1}{\epsilon}} &> 1
\end{aligned}
$$

Since the function $x \mapsto x^{\beta}$ is subadditive for $\beta \geq 1$ then:

$$
\left(\left(\frac{l_{pz}}{l_{pq}}\right)^{d'\epsilon} + \left(\frac{l_{zq}}{l_{pq}}\right)^{d'\epsilon}\right)^{\frac{1}{\epsilon}} \geq \left(\frac{l_{pz}}{l_{pq}}\right)^{d'} + \left(\frac{l_{zq}}{l_{pq}}\right)^{d'}.
$$

We know the right hand side is greater than 1 due to our initial assumption and therefore Eq. 3.2 is true. Therefore pq is also an edge in $SSG(d)$ and this completes the proof. □

By definition of $DN(d)$ and by the Thales' Theorem, $DN(pq, 2)$ becomes a closed circumcircle of pq and as $d \to \infty$, $DN(d)$ becomes the open lune. Therefore, the following two theorems result:

Theorem 3.2. $SSG(2) \equiv GG$

Theorem 3.3. *As* $d \to \infty, SSG(d) \to RNG$

By Theorem 3.3 we refer to $\lim_{d\to\infty} SSG(d)$ as $SSG(\infty) \equiv RNG$ in later sections. The following lemmas are trivial due to Theorems 24 and 25 of the Empty Region Graph study [6]:

Lemma 3.1. *For* $d \geq 2, SSG(d)$ *is planar.*

Lemma 3.2. *For all* $d \geq 1, SSG(d)$ *is connected.*

By combining lemmas 3.1 and 3.2, the range of d where $SSG(d)$ is both connected and planar can be deduced.

Lemma 3.3. *For* $d \geq 2, SSG(d)$ *is both planar and connected.*

Now we compare $SSG(d)$ to $SPG(t)$.

Theorem 3.4. *For* $d \leq t, SPG(t) \subseteq SSG(d)$

Proof. Define the *edge weight* of pq with respect to t as l^t_{pq}, for some $t \geq 1$. Assume that for all $z \in V \setminus \{p, q\}$, $l^t_{pz} + l^t_{zq} > l^t_{pq}$. In this case, pq is an edge in $SPG(t)$. Now we show that for $d \leq t$, pq is also an edge in $SSG(d)$. Let us write $d = t\,\epsilon$ where $\frac{1}{t} \leq \epsilon \leq 1$. Then

we need to show that:

$$
\begin{aligned}
l_{pz}^{t\epsilon} + l_{zq}^{t\epsilon} &> l_{pq}^{t\epsilon} \\
\frac{l_{pz}^{t\epsilon} + l_{zq}^{t\epsilon}}{l_{pq}^{t\epsilon}} &> 1 \\
\left(\frac{l_{pz}}{l_{pq}}\right)^{t\epsilon} + \left(\frac{l_{zq}}{l_{pq}}\right)^{t\epsilon} &> 1 \\
\left(\left(\frac{l_{pz}}{l_{pq}}\right)^{t\epsilon} + \left(\frac{l_{zq}}{l_{pq}}\right)^{t\epsilon}\right)^{\frac{1}{\epsilon}} &> 1
\end{aligned}
\tag{3.3}
$$

Since the function $x \mapsto x^{\beta}$ is subadditive for $\beta \geq 1$ then:

$$
\left(\left(\frac{l_{pz}}{l_{pq}}\right)^{t\epsilon} + \left(\frac{l_{zq}}{l_{pq}}\right)^{t\epsilon}\right)^{\frac{1}{\epsilon}} \geq \left(\frac{l_{pz}}{l_{pq}}\right)^{t} + \left(\frac{l_{zq}}{l_{pq}}\right)^{t}.
$$

We know the right hand side is greater than 1 due to our initial assumption and therefore Eq. 3.3 is true. Therefore pq is also an edge in $SSG(d)$ and this completes the proof. □

Now we consider the relationship between $SSG(d)$ and β-Skeletons. The empty regions used for both the Lune based β-Skeleton [113] and $SSG(d)$ are symmetric around the perpendicular bisector of the straight line connecting point pair. Using this similarity, we analysed how these neighbourhoods relate to each other for $d \geq 2$. As depicted on Figure 3.3, we observed that the β-Skeleton touches $DN(d)$ only on four points in the range of $\beta = (1, 2)$. Therefore, we can say that $DN(d)$ is more robust compared to the lune based neighbourhood of β-Skeleton in the range $\beta = (1, 2)$. We used the fact that Euclidean distances between these four points should be equal, to analyse the relationship between β and d. As a result of this analysis, we arrived at the following theorem.

Theorem 3.5. $\forall d \geq 2 \wedge \beta \leq 2^{(1-2/d)}$,

$$
SSG(d) \subseteq \textit{lune based } \beta\textit{-skeleton}
$$

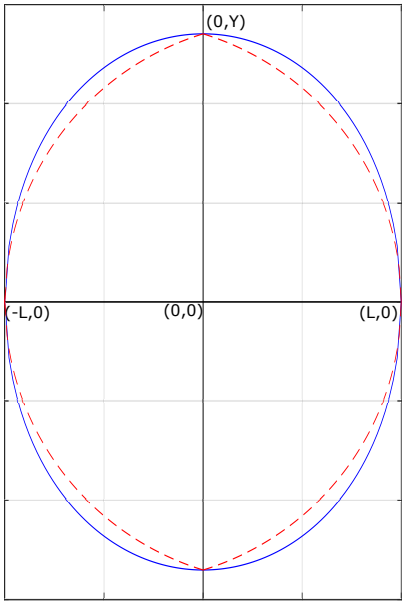


Figure 3.3: The lune based β skeleton neighbourhood in dashed lines and the Diversion Neighbourhoods in solid lines for planar and connected embedding

Proof. Figure 3.3 displays the schematic used to calculate Theorem 3.5. A Cartesian coordinate system is used with the inferring edge of length 2L as X axis. The midpoint of the inferring edge is taken as the origin. β neighbourhood is indicated by the dashed line plot and $DN(d)$ is indicated by the solid line plot.

The point where β neighbourhood touches $DN(d)$ above the edge is taken as $(0, Y)$. As boundaries of both neighbourhoods touch each other, Y calculated using both β neighbourhood (Y_β) and $DN(d)$ (Y_d) should be equivalent.

$$Y_\beta = Y_d$$
$$L\sqrt{2\beta - 1} = L\sqrt{(4^{(1-1/d)} - 1}$$
$$\beta = 2^{(1-2/d)}$$
$$\therefore \forall 2 \leq d \wedge \beta \leq 2^{(1-2/d)}$$

$$SSG(d) \subseteq \text{lune based } \beta\text{-skeleton}$$

□

Similar to the above analysis we can compare neighbourhoods of β-Skeleton in the range $\beta = (0, 1]$ and $SSG(d)$ in the range of $d = [1, 2)$. By analysing distances between points where two neighbourhoods touch each other we can derive the following theorem for non-planar regions of the $SSG(d)$ and lune based β-Skeleton.

Theorem 3.6.

$$\forall d \in [1, 2) \wedge \left(\frac{1}{\beta} - \sqrt{\left(\frac{1}{\beta^2} - 1 \right)} \right) \leq \sqrt{4^{(1-1/d)} - 1},$$

$$SSG(d) \subseteq \textit{lune based } \beta\textit{-skeleton}$$

Next we consider how $SSG(d)$ relates to DT.

Theorem 3.7. *For $d \geq 2, SSG(d) \subseteq DT$*

Proof. By theorems 3.3 and 3.2, $SSG(\infty) \subseteq SSG(2) \subseteq DT$. Let's assume DT excludes an edge which should be present in $SSG(d)$ for $d \geq 2$. For DT to exclude an edge, endpoints of that edge should not be Voronoi neighbors. An edge between a point pair which are not Voronoi neighbours will not create a planar graph. But, by Lemma 3.1 the excluded edge cannot be in $SSG(d)$ for $d \geq 2$. Thus, our initial assumption is wrong. Hence by proof of contradiction we establish that for $d \geq 2, SSG(d) \subseteq DT$. □

3.2.3 Algorithms

This section presents the algorithms to compute $SSG(d)$ and how to compute paths based on $SSG(d)$. Based on Theorem 3.7, we can use the complete graph (CG) and the DT

graph as starting graphs for extracting $SSG(d)$ for the ranges $2 > d \geq 1$ and $d \geq 2$, respectively. To make the extraction process more efficient, we label the edges spanned by these graph structures with the minimum value of d necessary for the edge to be excluded from $SSG(d)$. We define the starting graph structure, with edges labelled with these minimum d values, as the d-spectrum.

Definition 3.3 (d-spectrum)**.** *For an edge pq, the smallest real number $d \geq 1$ such that $DN(d) \cap V \setminus \{p, q\} \neq \emptyset$ is called the d-value for pq. The set of all edges spanned by V, each labelled by its d-value, is called the d-spectrum of V, denoted as d-spectrum(V) or simply d-spectrum.*

For $d \geq 1$, a naive algorithm can be formulated by evaluating all $z \in V \setminus \{p, q\}$ against each point pair $p, q \in V$ and comparing to find the d-value of pq. All edges which have other points on them will be discarded as they will never be present in $SSG(d)$. For the cases where an open lune neighbourhood is empty for an endpoint pair, the edge is labelled with ∞ as its d-value. All remaining edges will be labelled with d-values in the range of $1 \leq d < \infty$. For a V with n points, the number of edges that connect each point pair is in $\mathcal{O}(n^2)$. As each edge is compared against the other $(n-2)$ points, the time complexity of this naive algorithm is $\Theta(n^3)$.

3.2.3.1 Planar d-Spectrum(V)

For $d \geq 2$, DT can be used as a starting point due to Theorem 3.7. As a naive algorithm, for each point pair $p, q \in V$, where there is an edge between those points in DT, all $z \in V \setminus \{p, q\}$ can be evaluated to find the d-value of edge pq. For a V with n number of points, number of edges in DT is $\mathcal{O}(n)$. As each edge is compared against the other $(n-2)$ points, the time complexity of this algorithm is $\Theta(n^2)$.

A more efficient algorithm to obtain the d-Spectrum of edges spanned by $DT(V)$ is shown in Algorithm 1. In this approach, we are sweeping the triangles in $DT(V)$ that are intersecting with the open lune neighbourhood of each $p, q \in V$ such that $pq \in DT(V)$. The neighbouring triangles of $pq \in DT(V)$ are represented by the set $\Lambda(pq)$. $\Omega(T)$ denote

Algorithm 1: Planar d-Spectrum(V)

Input: V - Filtered locality set
Output: Every edge $pq \in DT(V)$ marked with d-value

```
1  DT ← create Delaunay Triangulation of V;
2  foreach (Edge pq such that pq ∈ DT) do
3      d ← ∞;
4      ME ← pq;
5      C ← Center of pq;
6      foreach (T ∈ Λ(pq)) do
7          MT ← T;
8          Z_max ← Z_max ∈ DN(d) such that max⌈l_{CZ_max}⌉;
9          while (MT ≠ ∅) do
10             if (Z_max ∈ Ω(MT)) then
11                 break;
12             end
13             MV ← Vertex Z ∈ MT such that Z ∉ ME;
14             if (MV ∈ I(DN(d))) then
15                 d ← d such that MV ∈ B(DN(d));
16                 update Z_max;
17             end
18             ME ← ∃edge E ∈ MT \ ME such that ME ∩ Z_max ≠ ∅;
19             if (ME = ∅) then
20                 break;
21             end
22             MT ← T1 ∈ Λ(ME) such that T1 ≠ MT;
23         end
24     end
25     set d as d-Value of pq;
26 end
```

the circumscribed circle of a triangle T. $I(DN(d))$ denote the interior of $DN(d)$, while $B(DN(d))$ denote the boundary of $DN(d)$. We start the process for each $pq \in DT(V)$ with one neighbouring triangle of pq (Line 7). And we continue sweeping the triangles intersecting with the line segment connecting the center point of pq and $Zmax$ which is the furthest away point in $DN(d)$ from pq. Given the Delaunay triangulation, this algorithm runs in $\mathcal{O}(n)$ time as it only examines the points within the open lune neighbourhood of each edge.

When presenting the time complexities of proposed algorithms above, we assume that the time taken to solve for the d-value of an edge is $\mathcal{O}(1)$, i.e. that we ask for constant

precision. Note that all of the above algorithms are readily parallelizable as there is no race condition between separate edge evaluations.

3.2.3.2 Computing $SSG(V, d)$

Once the d-spectrum of V is calculated for a given d range, extracting the edges of $SSG(d)$ is a straightforward process. As d-Spectrum indicates d-value of each edge, edges that have d-value greater than given d can be filtered out as the edge set of $SSG(d)$. This process will take $\mathcal{O}(m)$ time complexity where m is the number of edges resulting from the method used to calculate the d-Spectrum. For $1 \leq d < 2, m$ would be $\mathcal{O}(n^2)$, and for $2 \leq d, m$ would be $\mathcal{O}(n)$.

3.2.3.3 Improving Running Time of $SPG(t)$

Using the d-spectrum, we can exclude some of the edges from $SPG(t)$, without calculating the alternative shortest path, to improve the running time of $SPG(t)$ extraction. By Theorem 3.4, the edges of the d-spectrum that have a d-value less than or equal to a given t value, will not be in the $SPG(t)$. This property can be used to skip edges from the shortest path calculation part of $SPG(t)$ extraction algorithm [18]. It should be noted that even though this method improves the running time of the $SPG(t)$ algorithm, it does not reduce the time complexity of the worst case.

3.2.3.4 Calculating Paths Using $SSG(d)$

After calculating $SSG(d)$, Dijkstra's algorithm can be used to find the shortest paths between any two endpoints. As $SSG(d)$ contains more alternative paths compared to $SPG(t)$, paths taken over $SSG(d)$ will be shorter and more similar to real world paths.

3.2.4 Applications

As shown in the experiments section later, $SSG(d)$ skeleton itself can be used to represent and process the travel network between a set of related localities. In this scenario, inferred edges represent the space occupied by an entity when moving between locations. Following are some application scenarios that can benefit from $SSG(d)$.

3.2.4.1 Nearest Neighbour Queries

The created graph structure can be used to search nearest interesting locations from the current location and to get the path to travel in order to get there. For example, this kind of query can be used to find the nearest exit from an event happening in a park. We propose to use breadth first search starting from the query location and traverse the graph until a required interesting point is found.

Definition 3.4 (Nearest neighbour (NN))**.** *A location (L) from an interesting location set (IL) such that the shortest path distance from current location to that location is smaller compared to the shortest paths to all other locations,*

$$NN = \{\exists L \in IL \text{ such that } \forall x \in IL, length(path(L)) \leq length(path(x))\}$$

Similarly, we can calculate k-nearest neighbours. Instead of stopping breadth first search when the first interesting point is found, it can be continued until k interesting points are found. As for the edge weights, we can use weights calculated in the section 3.4.6 according to the usage of edges. This will make sure that the most popular path to the nearest neighbour will be found.

3.2.4.2 Reverse Nearest Neighbour Queries

$SSG(d)$ graphs structure can be used to perform reverse nearest neighbour queries. These queries focus on the inverse relation among a point and a point set. The objective of a reverse nearest neighbour query is to find the set of points which has a given point as their nearest neighbour. This kind of query can be used to evaluate locations to place attractions of an event relative to entrances of the event.

Definition 3.5 (Reverse Nearest neighbour (RNN)). *A set of locations (RNN) from an interesting location set (IL) such that the current location (x) is the nearest neighbour of all locations in RNN,*

$$RNN = \{\forall L \in IL \text{ such that } x = NN\}$$

We propose an algorithm to calculate RNN of a given location (x) utilising breadth first search. First, breadth first search is initialized and run from all locations in IL. Then all locations that have x as their nearest neighbour are filtered as the RNN of x.

3.2.4.3 Group Nearest Neighbour Queries

Another important query type that can be evaluated using $SSG(d)$ is group nearest neighbour queries. Given two sets of points $IL1$ and $IL2$, the objective of a group nearest neighbour query is to retrieve a point from $IL1$ with the smallest sum of distances to all points in $IL2$. For example, this kind of query can be used to find the nearest attraction of an event where a group of users in different locations can get together.

Definition 3.6 (Group Nearest neighbour (GNN)). *A location (GNN) from an interesting location set ($IL1$) such that, it minimizes the sum of distances to all locations in another location set $IL2$.*

$$GNN = \{\exists L \in IL1 \text{ such that } \forall x \in IL2, min(\Sigma length(path(x)))\}$$

Again we utilize breadth first search to propose an algorithm to calculate GNN. First,

we calculate k-nearest neighbour from all points of $IL2$ to all points in $IL1$. Then for each point in $IL1$ we calculate the sum of path distances from all points in $IL2$. Point or points in $IL1$ with the minimum sum of path distances is the GNN.

3.2.4.4 Refined Movement Corridors

Once $SSG(d)$ is created using posted localities, user trajectories can be used to refine the created travel network. For each consecutive location pair in user trajectories, the shortest path is determined using $SSG(d)$. For each $SSG(d)$ edge the number of trajectories passed through that edge is recorded. We represent movement corridors in the travel network based on edges that contain trajectory count higher than a given threshold.

Definition 3.7 (Usage counter)**.** *When path is a sequence of edges traversed by a trajectory trace, for all $pq \in E$, Usage Counter of pq (denoted $UC(pq)$), is defined as the trajectory count,*

$$UC(pq) = |\{path\ such\ that\ pq \in path\}|$$

One of the problems of using $SSG(d)$, as it is for movement network analysis, is that it does not consider the existence of obstacles. As trajectories do not appear on obstacles such as rivers, incorporating trajectory information into $SSG(d)$ allows filtering edges not used by the trajectories. By filtering edges not used by trajectories, we are able to eliminate edges that do not represent user movement information. In summary, refined movement corridors calculated using $SSG(d)$ represents the edge subset of $SSG(d)$ used by the trajectories for movement.

3.2.4.5 Inferring Road Network

Above refined movement corridors finding method can be used to infer the road network in an area where we do not have prior knowledge about the road networks. Ideally for this purpose, we need GPS locations published on the road network. The easiest way

to obtain such information is to collect LBSN post published while travelling in vehicles. Using these GPS data and associated trajectories, we can infer the refined movement corridors for the given area and visualize the road network.

3.2.4.6 Most Popular Paths

After calculating usage counters of $SSG(d)$ edges, they can be used to find the most popular path between locations. To find the most popular path, we calculate edge weight to reflect the popularity of the edge and use the shortest path algorithm to calculate the paths. To ensure edges with more usage have lower weights, we divide the length of the edge by usage counter of that edge. For edges with no usage, edge length multiplied by a constant greater than 1 is used as the edge weight. After calculating edge weights in this manner, the Dijkstra's shortest path algorithm is used to find the most popular path between two locations.

Definition 3.8 (edge weight)**.** *For all $pq \in E$, weight of pq is defined as,*

$$weight(pq) = \begin{cases} l_{pq}/UC(pq), & \text{if } 0 < UC(pq). \\ l_{pq} \times C \text{ such that } 1 < C, & \text{if } UC(pq) = 0. \end{cases}$$

3.2.4.7 Tour Recommendation

Above most popular path calculation mechanism can be used to calculate the most popular paths between tourist attractions. Given tourist attractions and user trajectories within a city, first we calculate $SSG(d)$ and edge weights based on usage. Then, we apply the most popular path calculation between any two selected attractions to find the path popular among tourists to travel between the selected attractions. The sequence of tourist attractions that appear on the calculated most popular path is reported as a recommended tour along with the path.

3.2.4.8 Trajectory Clustering

Trajectory clustering refers to the process of grouping similar trajectory traces together. This process involves deriving a representative trajectory for the identified trajectory group. We can use $SSG(d)$ to identify similar trajectory traces and to derive representative trajectory.

First, we align trajectory traces to be clustered along the edges of the created $SSG(d)$. Then for each trajectory trace, edge set that trajectory trace traverse through is recorded. If two trajectories have C ratio of similar edges, they are considered belonging to the same cluster. The subset of similar edges is taken as the representative trajectory of that cluster.

Definition 3.9 (Trajectory similarity)**.** *When path is a sequence of edges traversed by a trajectory trace, two trajectory traces are considered to be similar if they have C percentage of edges in common with each other,*

$$similarity(path_1, path_2) = \frac{|path_1 \cap path_2|}{|path_1|}$$

Both $similarity(path_1, path_2) \geq C$ and $similarity(path_2, path_1) \geq C$ where $0 \leq C \leq 1$ must be true, for $path_1$ and $path_2$ to be in the same cluster.

This method of trajectory clustering is developed without considering the trajectory directions. But this method can be easily changed to accommodate trajectory direction (i.e. cluster only the trajectories travelling in the same direction). For this, we have to consider the endpoint traversing order in addition to considering the edge similarity by endpoints.

3.2.4.9 Group Movement Detection

Above method proposed to perform trajectory clustering can be further extended to detect users moving together as groups. In order to detect this, we need to find trajectory traces that have traversed through same edges within the same time in the same direction. To accomplish this we propose to calculate the time when a user passed through each location in the path resulting from aligning the trajectory on the $SSG(d)$. After that similar to trajectory clustering method we detect not only trajectory traces travelled through the same edges, but also whether they have travelled around the same time and in the same direction. Trajectory traces that have travelled through same edges, around the same time and in the same direction are filtered as user groups travelled together. This proposed method to detect group movement is applicable in both sparse geo-located data such as LBSN data and dense data such as data from automated vehicles. But in our experiments, we are considering the situation where LBSN data is used to detect group movements.

One problem we have to deal with when using sparse data for group movement detection is the irregularity of group memberships. This happens because not all members of the group are posting their location at every time step. The proposed method mitigates this problem by calculating the time when each group member passed through each location in the group's path over the graph. Therefore, automatically we assume that a group member has passed through the group path location at the calculated time.

Note that in this method we consider only users travelled on the same path. This method can be further extended to detect user groups travelled not only on the same path but through nearby paths going in the same direction. This extension can be used to identify flock movement [116].

3.3 Experiments

We perform experiments to see the effectiveness of $SSG(d)$ against $SPG(t)$ on inferring movement networks.

3.3.1 Data Sets

We conducted experiments on two real world LBSN datasets and one synthetic data set. The first data set consists of Geo-located posts collected from Twitter. A post set collected from 06th March to 23rd April 2012, within a bounding box over the countries Australia and New Zealand is used as the data set for our analysis. It contains 724651 LBSN posts authored by 36639 users. For our analysis an LBSN post is defined as a tuple containing four elements - userID to identify authoring user, voluntarily generated textual content, the timestamp at which and the locality where the post was authored.

We used Yahoo! Flickr Creative Commons 100M (YFCC100M) dataset [117] as the second dataset. It contains meta data such as user information, timestamp and location where a photo was taken of 100 million photos and videos shared on site Flickr. Only the entries with point geo-locations were used for our experiments. For refined movement corridor and tour recommendation experiments localities in London, England is filtered out from the YFCC100M data set.

Synthetic data set for our experiments was generated using SMARTS simulator [118]. We simulated vehicle movement in the Melbourne central business district (CBD) and collected GPS locations of the vehicles every 0.5 seconds. Data set used for experiment contained 100000 GPS points. This is essentially a dense GPS data set. In order to make it sparse, we took a portion of this data set. Data point filtering is further explained in the road network inferring experiment.

3.3.2 Implementation

To visualize the inferred neighbourhood skeletons, a visualization tool was implemented utilizing GeoTools[1] Java libraries. All the skeleton visualizations presented in all figures were generated using this tool. Both $SSG(d)$ and $SPG(t)$ algorithms were implemented using Java 8. The data set was stored in a MongoDB database and unique locality set relating to a given event was queried from it as V. We used a travel network filtered from OpenStreetMap (OSM)[2] to get a better understanding of the area being analysed. When filtering travel network of an area from OSM data we took foot paths, cycle paths, roads and rail roads.

To infer planar $SSG(d)$, firstly, DT was created using SweepHull [119] algorithm in $\mathcal{O}(n \log n)$ time. Then, planar d-Spectrum is calculated using Algorithm 1. Finally, $SSG(d)$ is extracted from d-Spectrum created using DT. We used numerical analysis to calculate d-value of an edge. More specifically, a Java method was implemented to perform Secant method[3] to approximate the d-Value. For extracting planar $SPG(t)$, we used Algorithm 2 presented in [18] [4]. For varying $SSG(d)$ and $SPG(t)$ skeleton extractions, created DT was reused.

3.3.3 Event Analysis

An event can be viewed as an extraordinary spatiotemporal phenomenon that motivates LBSN users to generate posts within the spatial and temporal bounds of the event with content identifying the event. Following this perspective, we define an event as a tuple containing three elements - a spatial bound, a temporal bound and a description of content identifying the event. As the Twitter data we are using has textual content, we use a regular expression as the description of the event.

[1] http://www.geotools.org/
[2] https://www.openstreetmap.org/
[3] https://en.wikipedia.org/wiki/Secant_method
[4] However, we noticed that $\geq$ in line 5 violates the edge selection criteria proposed as equality allows paths with similar weights to exists between an edge pair. Therefore we removed equality in line 5 when creating our implementation.

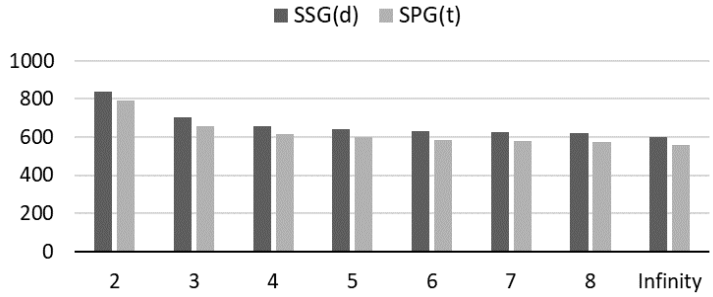
To discuss the properties of $SSG(d)$ and $SPG(t)$ skeletons we selected locality set relating to EasterFest 2012 [5] - an annual music festival held in Toowoomba, Australia from the Twitter data set. For temporal bound of EasterFest, we took the time period between 04th and 10th of April 2012. As for the spatial bound of EasterFest, we considered a bounding box (indicated by dashed lines in Figure 3.1 (a)) over the Queen's Park, Toowoomba and surrounding main streets where the event was held. We consider all users who have posted with "#EasterFest" hashtag within spatial and temporal bounds of the event as the user set influenced by the event.

The locality filtering process can be seen as a three step process. First, the users who have authored content matching the description of the event within the event's temporal and spatial bounds are filtered. Second, the LBSN trajectories within the event's temporal bounds generated by the users filtered in the first step are taken as the LBSN posts set influenced by the event. Third, unique locality set from posts filtered in the second step is taken as the V for variable skeleton generation. Relating to Easterfest, 183 unique points were filtered. Figure 3.1 (a) depicts the filtered locality set along with the travel network of the area analysed. We used the rectangle marked with the dashed lines as the spatial bound of the event. This rectangle encloses the event venue (Queens Park, Toowoomba) and the main streets around it as described above.

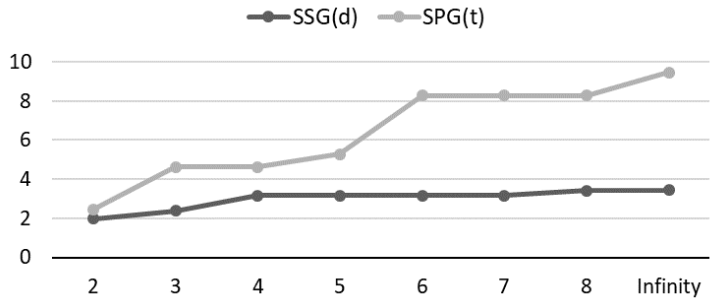
3.3.4 Comparison of Two Graphs

As experiments for filtered locality set (V) relating to EasterFest, we generated varying skeletons with $SSG(d)$ and $SPG(t)$ at $d = t$. After that, generated skeletons are analysed to compare the changing characteristics of the skeletons as parameters vary. We used a locality set relating to an event because all users participating in the event are there for a common reason and exhibit a similar movement pattern. To further understand how graph structures behaved, the number of edges and spanning ratio [115] of the graph structure were plotted with varying configuration parameters (Figure 3.4).

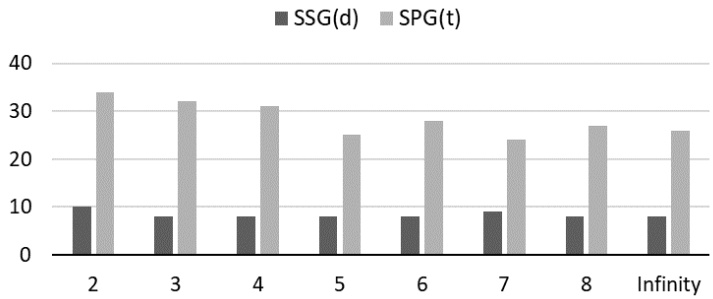
[5] http://www.easterfest.com.au



(a) Number of edges with varying parameter value



(b) Spanning ratio with varying parameter value



(c) Execution times with varying parameter value

Figure 3.4: Comparison of SSG(d) and SPG(t).

Figure 3.1 (a) and (b) present the locality set used as the V and the DT skeleton. Three skeletons on the middle left c, e and g represent the $SPG(t)$ with t set to 2, 3 and 4, respectively. While three skeletons on the middle right d, f and h represent $SSG(d)$ with d set to 2, 3 and 4, respectively.

At the first glance, we can see that $SPG(t)$ is a subgraph of $SSG(d)$ when $d \leq t$ (Theorem 3.4). Also, in both $SSG(d)$ and $SPG(t)$, edge counts are monotonically reducing (Figure 3.4 (a)). Further more, number of edges in $SPG(t)$ is always smaller than

that of $SSG(d)$. This is to be expected as $SPG(t)$ extraction uses a global criteria compared to $SSG(d)$ and infer less edges. Figure 3.4 (b) shows the variation of spanning ratio as configuration parameter varies to demonstrate how the shortest path distances between locality pairs change. Spanning ratio of a graph indicates the maximum ratio between the shortest path distance over the graph and direct distance between any point pair. Therefore, graphs with low spanning ratio are preferred to represent movement networks [115]. Since $SSG(d)$ has a low and stable spanning ratio compared to $SPG(t)$, $SSG(d)$ is suitable for movement analysis. Furthermore, when $SPG(t)$ and $SSG(d)$ are created with the same number of edges, $SPG(t)$ tends to have a higher spanning ratio compared to $SSG(d)$.

Next, we conduct an experiment to see how similar the paths from $SSG(d)$ and $SPG(t)$ are when compared to real world paths taken from OpenStreetMaps. We randomly selected 10 point pairs from the filtered point set and calculated path distances between them using OpenStreetMap paths, $SPG(3)$ and $SSG(3)$. We selected 3 as the configuration parameter for both $SSG(d)$ and $SPG(t)$ as it provides a general movement network between localities. For 9 point pairs, paths taken using $SSG(3)$ were different by 20% or less by length compared to real world path taken from OpenStreetMap. However, for paths taken using $SPG(3)$ only 6 pairs were in the same bracket. Therefore, we can see that paths inferred using $SSG(d)$ are more similar to real world paths compared to paths inferred using $SPG(t)$.

We also calculated nearest neighbour queries using created graph structures. As we used breadth first search for querying nearest neighbours, observed results were similar to results of the shortest path calculations above. Paths calculated to nearest neighbours using $SSG(d)$ were more similar to real world paths compared to paths calculated using $SPG(t)$.

When analysing the spectrum of graph skeletons presented in Figure 3.1, we can see that as configuration value increases, inferred $SSG(d)$ skeleton emphasizes more on tight locality clusters, by thinning longer edges. Relating to the selected event by analysing tweets, it can be seen that filtered localities are densely centered towards the Points of In-

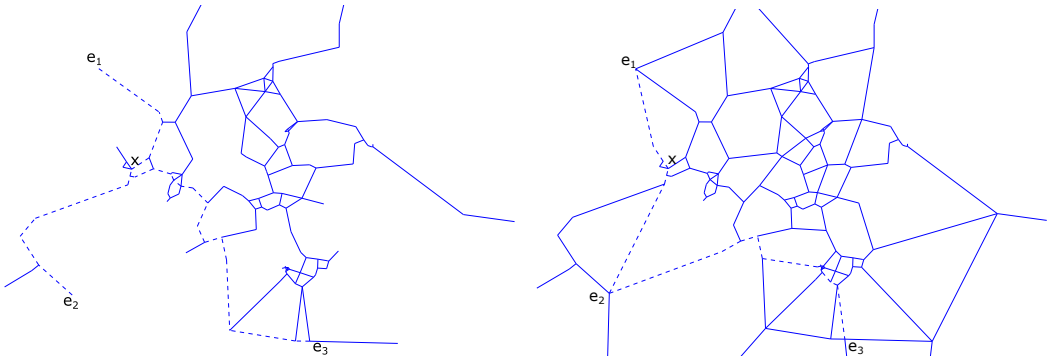
terest (POIs) such as event location and main streets in the area. This phenomenon makes it easier to visualize POIs clearer, along with the possible movement network between POIs. Furthermore, when comparing inferred skeletons, against the road maps available via public sources, it is observed that inferred edges of both $SSG(d)$ and $SPG(t)$ align better with road network as locality density increases in roads. But, as locality density varies, $SSG(d)$ skeleton highlights more alternative paths compared to $SPG(t)$ with the same configuration value.

In order to generate varying visualizations, the skeleton needs to be visualized with varying parameters manually. Time taken to calculate skeletons of $SSG(d)$ and $SPG(t)$ are shown in Figure 3.4 (c). Execution time for $SSG(d)$ calculation is less compared to $SPG(t)$ for all configuration values. To generate $SSG(d)$ skeleton of a given point set, we need to calculate the d-spectrum once and can extract skeleton for desired d from it. Once d-spectrum is calculated $SSG(d)$ skeleton can be rendered in $\mathcal{O}(m)$ time, where m is the number of edges spanned by V. Therefore, we can generate varying $SSG(d)$ skeletons efficiently. For $SPG(t)$, we have to execute iterative shortest path algorithm every time t changes, making analysis process inefficient. But, we can reduce this time using calculated d-Spectrum. However, this does not reduce the worst case time complexity of $SPG(t)$ algorithm.

3.3.5 Further Interesting Observations From Spatial Queries

3.3.5.1 Reverse Nearest Neighbour

Reverse nearest neighbour queries aim to find given a query location and a location set, locations of the set where query location is their nearest neighbour. When proposing applications of the $SSG(d)$ we proposed an algorithm to retrieve reverse nearest neighbours of a given query location utilizing breadth first search. In order to experiment the suitability of $SSG(d)$ for reverse nearest neighbour queries, we have selected an arbitrary location from the venue of the Easterfest 2012 as the query location and exits of the venue



(a) SPG(3) (b) SSG(3)

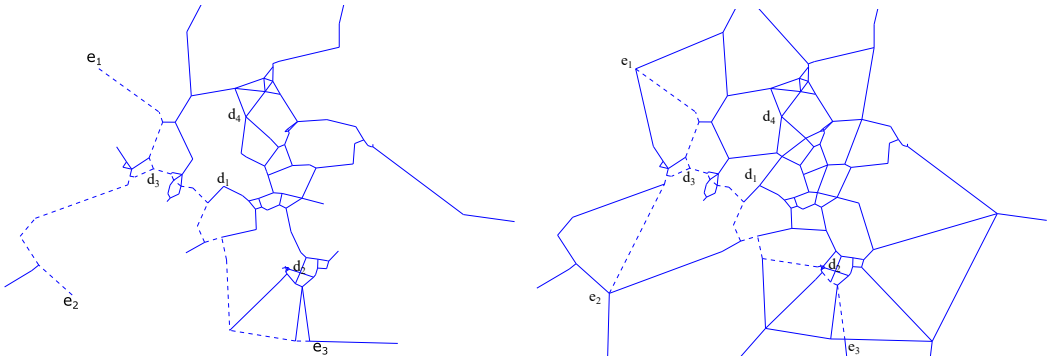
Figure 3.5: Dashed lines from e_1, e_2 and e_3 indicate their nearest neighbour calculated using each graph.

as the location set.

We executed the reverse nearest neighbour query on both $SPG(3)$ and $SSG(3)$. Results are shown in Figure 3.5. For both exits e_1 and e_2, query location x became the nearest neighbour in both $SPG(3)$ and $SSG(3)$. For e_3, nearest neighbour results were different for two graphs. With $SPG(3)$, x became the nearest neighbour for e_3. With $SSG(3)$, e_2 became the nearest neighbour for e_3. Therefore with $SPG(3)$, reverse nearest neighbours of x are e_1, e_2, e_3 ,while with $SSG(3)$ reverse nearest neighbours of x are e_1, e_2. In euclidean space nearest neighbour of e_3 is e_2. Therefore, we can see that $SSG(d)$ is more suitable for reverse nearest neighbour queries compared to $SPG(t)$ with the same configuration parameter. Also, note that paths selected from exits to query location are shorter in $SSG(3)$ compared to $SPG(3)$.

3.3.5.2 Group Nearest Neighbour

Group nearest neighbour queries are evaluated between two sets of locations. This query type aims to find a location or locations from one set such that cumulative travel distance from all locations in the other set is minimized. Our proposed algorithm to calculate group nearest neighbour is to calculate k-nearest neighbour from all locations in one set to find all locations in the other set and evaluate which location in the second location



(a) SPG(3) (b) SSG(3)

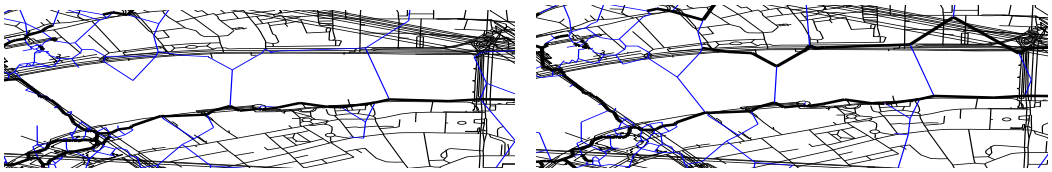
Figure 3.6: Dashed lines from e_1, e_2 and e_3 indicate their group nearest neighbour calculated using each graph, between destinations d_1, d_2, d_3 and d_4.

set results in the least cumulative travel distance from all locations in the first set. To evaluate group nearest neighbour queries we selected the same location set filtered for Easterfest 2012.

We experimented performing group nearest neighbour queries on both $SPG(3)$ and $SSG(3)$. Results are shown in Figure 3.6. We calculated group nearest neighbour from three exits (e_1, e_2 and e_3) to four arbitrary locations within the event venue (d_1, d_2, d_3 and d_4). Using both graphs $SPG(3)$ and $SSG(3)$, destination d_3 was calculated as the group nearest neighbour. Dashed lines in Figure 3.6 indicate the path calculated from an exit to the group nearest neighbour. Even though, both graphs resulted in same group nearest neighbour, note that paths calculated using $SSG(3)$ are shorter compared to paths calculated using $SPG(3)$ due to the availability of more alternative paths. Hence we can see that $SSG(d)$ is more suitable for group nearest neighbour calculations compared to $SPG(t)$ with the same configuration parameter.

3.3.5.3 Refined Movement Corridors

Refined movement corridors refer to edges of the graph that are used for movement. These edges are selected by aligning user trajectories along the graph edges using shortest path calculation. We analysed refined movement corridors relating to the trajectories



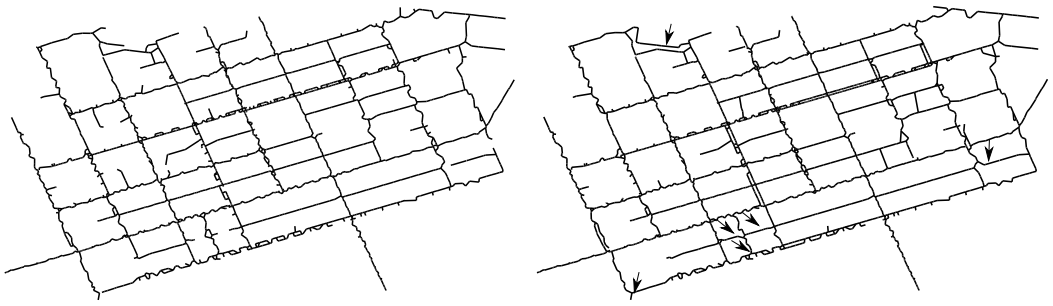
(a) SPG(3) (b) SSG(3)

Figure 3.7: Thick lines indicate refined movement corridors extracted using each graph.

filtered from YFCC100M dataset, around the Thames river in London. To represent the travel networks, $SPG(3)$ and $SSG(3)$ were used. This data set was selected because it had a lot of tourist movement compared to the Twitter data set. After that trajectories are aligned along both graph skeletons, and all the edges with usage counter more than 5 are filtered as refined movement corridors. Intuitively if an edge is used by trajectories 5 times, that edge is selected as a refined movement corridor. Visualization created using filtered edges are shown in Figure 3.7. In both refined movement corridors represented by thick edges appear along the banks of the river and on bridges where movements happen. Edges created across the river are filtered out as there is no movement happening. When all trajectories are aligned against the refined movement corridors calculated using $SSG(3)$ and $SPG(3)$, 14% more of trajectories appear on refined movement corridors calculated using $SSG(3)$ compared to $SPG(3)$. Hence refined movement corridors created using $SSG(d)$ are more similar to real world paths compared to refined movement corridors calculated using $SPG(t)$. As this method indicates the most used part of the travel network, it can be used to generate preferred part of a travel network used for calculating preferred paths [120].

3.3.5.4 Inferring Road Network

Refined movement corridor extraction can be used to infer the road network of an area. GPS locations posted on the road network are useful for this purpose. Using our synthetic data set we inferred the road network of the Melbourne CBD. In order to make this data sparse similar to LBSN data, we filtered out some of the points. Filtering process was to order all GPS points based on the timestamp and take every n^{th} point from the whole data



(a) SPG(3) (b) SSG(3)

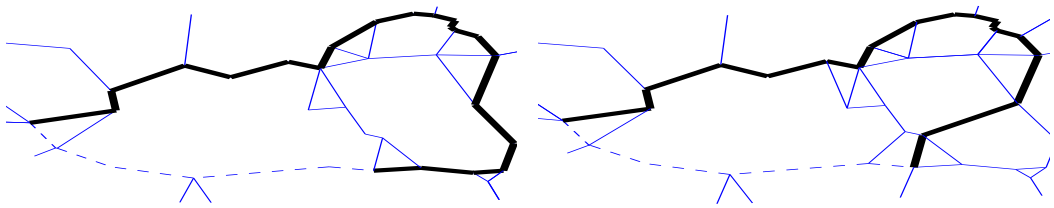
Figure 3.8: Road network (refined movement corridors) extracted using synthetic data on each graph. Arrows on (b) indicate additional movement corridors inferred using $SSG(3)$.

set. However, we used all the trajectories for the refined movement corridor calculation.

In Figure 3.8, we have shown road networks inferred using (a) $SPG(3)$ and (b) $SSG(3)$. As shown in the figure road network inferred using $SPG(3)$ breaks around some junctions, but $SSG(3)$ recreates the junctions correctly. Overall, 12% more road segments inferred using $SSG(3)$ aligns with real road network compared to $SPG(3)$. It should be noted that as n grows, the data set used to infer road network become more sparse, therefore results generated using this method degrades. Results heavily degrade when n reaches around 120.

3.3.5.5 Most Popular Paths

After calculating refined movement corridors over an area, edge weights are set to exhibit the usage of that edge as shown in section 3.2.4.6. Most popular paths between two locations can be found by calculating the shortest path using above calculated edge weights. In order to experiment with the effectiveness of the most popular path finding mechanism, we applied it to find a path around a junction in Melbourne Central Business District (CBD). Data for the experiment is taken from the Twitter data set. As twitter data set was restricted to the region Australia, it has a lot of movement information of Twitter users relating to their daily travels. One example of the most popular path is shown



(a) SPG(3) (b) SSG(3)

Figure 3.9: Paths calculated between two arbitrary locations using each graph. The dashed line indicates the shortest path and the thick line indicates the most popular path. Note that the most popular path calculated with $SSG(3)$ is shorter than that of $SPG(3)$

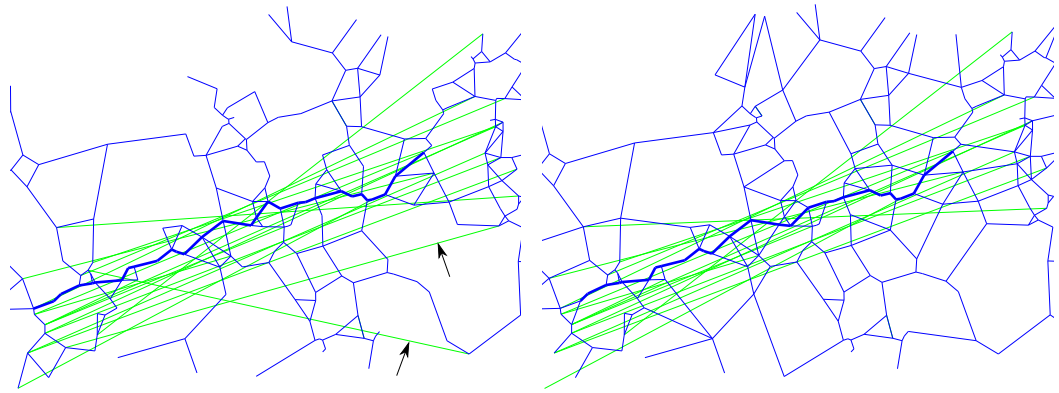
in Figure 3.9. $SPG(3)$ and $SSG(3)$ were used to represent travel networks, respectively. Calculated shortest paths are shown in broken line and most popular paths are shown in thick edges. As shown in the Figure 3.9 most popular path inferred using $SSG(3)$ is shorter compared to that of $SPG(3)$. In fact when most popular paths are calculated for all trajectories, most popular paths calculated using $SSG(3)$ were 18% shorter than most popular paths calculated using $SPG(3)$. In this particular case when selected most popular paths are compared with travel network taken from OpenStreetMaps, it is seen that there are roads that align with the most popular path whereas no actual road aligns with shortest paths.

3.3.5.6 Tour Recommendation

Using YFCC100M data set we experimented the effectiveness of tour recommendation in the city of London. We followed the experimental set up followed in [121]. Geo-located photos posted over two months time period were used for experiments. All tourist travel sequences set apart by one hour time intervals are taken as the query data set. For each tourist attraction sequence travelled by users, we calculated the most popular path between first and last tourist attractions they've been to and selected tourist attractions that coincide with the most popular path as recommended tour itinerary. For each real world travel sequence (P_v), a recommended tour itinerary (P_r) was calculated using both $SPG(3)$ and $SSG(3)$. For tours recommended by both graphs tour recall ($T_R = |P_r \cap P_v|/|P_v|$), tour precision ($T_P = |P_r \cap P_v|/|P_r|$) and tour F_1-score($T_{F1} =$

Table 3.1: Comparison of tour recommendations calculated using $SPG(3)$ and $SSG(3)$

Graph	Recall	Precision	F_1-score
$SPG(3)$	0.699 ± 0.050	0.681 ± 0.060	0.676 ± 0.050
$SSG(3)$	$\mathbf{0.777 \pm 0.050}$	$\mathbf{0.783 \pm 0.060}$	$\mathbf{0.774 \pm 0.059}$



(a) SPG(3) (b) SSG(3)

Figure 3.10: Thick lines indicate representative trajectory extracted for trajectory set shown as green (light coloured in black and white print) lines using each graph. Additional trajectory traces that belong to $SPG(3)$ cluster are marked with arrowheads.

$(2 \times P_r \times P_v)/P_v \times P_r)$ [121] were calculated and presented in Table 3.1. As shown for all three criteria $SSG(3)$ evaluates higher values than $SPG(3)$. Thus, we can say that the inferences using $SSG(d)$ are more close to real world user travels than $SPG(t)$.

3.3.5.7 Trajectory Clustering

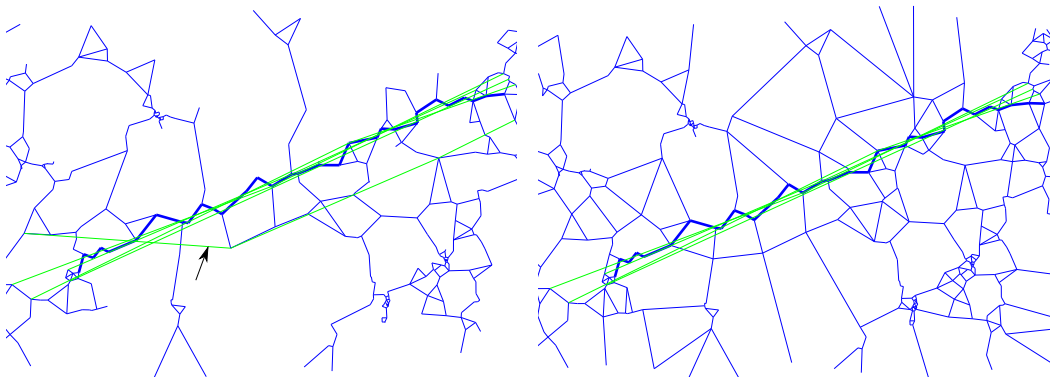
Proposed $SSG(d)$ can be used for trajectory clustering by aligning trajectories on the created graph and selecting trajectory traces that share the same edge sets from the graph. To evaluate $SPG(t)$ and $SSG(d)$ for trajectory clustering we used trajectory data collected in Melbourne CBD from Twitter data set. A trajectory cluster detected in the same area using both $SPG(3)$ and $SSG(3)$ is shown in Figure 3.10. We set the similarity threshold value C to 0.5. Green (light colour in black and white print) lines in the figures indicate trajectory traces included in the cluster, while thick lines indicate the representative trajectory calculated using graphs consisting of the shared edge set among trajectories. The

trajectory cluster calculated using $SPG(3)$ contains 15 trajectory traces while the cluster calculated using $SSG(3)$ only contains 13 traces. Two additional trajectory traces that belong to the cluster found using $SPG(3)$ are marked by arrows. Reason for $SSG(3)$ cluster leaving 2 traces out is due to the availability of alternative paths trajectory traces that deviate from other trajectory traces in the cluster can find paths other than the paths used by the trajectories in the cluster. Since $SPG(3)$ is more restrictive than $SSG(3)$, trajectory traces that deviate from most of the trajectory traces in the cluster have to travel in the same paths used by the clustered trajectory traces. Following the quality measures developed in [97] to measure the quality of trajectory clusters, we developed a quality measure. Once trajectory traces are aligned along the edges of the graph, we take the lengths of edges in the trajectory that are outside the representative trajectory of the cluster and divide the cumulative outside length of all the edges by the number of trajectories in the cluster. By definition of this measure lower the calculated value for this measure, better the cluster. With this measurement, the trajectory cluster is shown in Figure 3.10, clustered using $SSG(3)$ was 13% better than the cluster identified using $SPG(3)$. Therefore, we can say $SSG(d)$ is more suitable to identify similar trajectory clusters compared to $SPG(t)$.

3.3.5.8 Group Movement Detection

This application refers to the use of $SSG(d)$ to detect user groups moving together. This is an extension of trajectory clustering presented earlier. Instead of grouping trajectories based only on their spatial proximity, in the group movement detection we consider their direction and travelled time. Using this information we find trajectories travelled on the same path, in the same direction around the same time.

In both data sets we used, there were not any group movements that exhibit the difference between group movement detection using $SPG(t)$ and $SSG(d)$, due to the small size of data we have from real LBSN leading to very few group formation. Therefore, we created a synthetic data set using a geo-located Twitter post data set to exhibit this



(a) SPG(3) (b) SSG(3)

Figure 3.11: Thick lines indicate the path of group movement detected (from left to right) using each graph.

difference. In this synthetic data set, locations of the geo-located posts are not altered. However, timestamps and user ids are altered to make synthetic trajectories.

With this synthetic data set using both graphs $SPG(3)$ and $SSG(3)$, we detected a group movement occurred in Melbourne CBD. In the detected group movement with $SPG(3)$ there are trajectories of 5 users aligning with the same path of the graphs travelling in the same direction with a time difference of 10 minutes. However, in the group movement detected using $SSG(3)$ there are only 4 users. Detected group movement paths are shown in Figure 3.11. Thick lines in the graphs indicate edges of the path shared by the detected group. Lines in green colour (light colour lines in black and white print) indicate trajectories of the users travelled together. In the $SPG(3)$ case there is a trajectory marked with an arrowhead that is not travelling with other trajectories. This trajectory is clustered with other trajectories that exhibit group movement because of the restrictiveness on the $SPG(t)$. Therefore, group movement detection using $SPG(t)$ can suffer from the same weaknesses that would hinder its performance in trajectory clustering. Due to the more restrictiveness of $SPG(t)$, trajectories that are not supposed to travel in the same path may end up travelling in shared paths. This is avoided in $SSG(d)$ due to the availability of more alternative paths. Therefore, $SSG(d)$ is more suitable to detect group movements using the proposed method.

3.3.6 Configuration Value

We selected 3 as the configuration value to compare resulting graphs generated using $SSG(d)$ and $SPG(t)$. For configuration value 1 both $SSG(d)$ and $SPG(t)$ are the same and do not cull edges to cater for point density. And for configuration parameter value 2, $SSG(d)$ become GG which is a well known static graph. Configuration parameter value 3 gives interesting results so we have used it. Configuration values greater than 3 give similar results as the results generated with the configuration parameter set to 3.

3.4 Summary

We analysed the suitability of variable connected neighbourhood skeletons to visualize the possible movement network of LBSN users influenced by events. We presented the Stepping Stone Graph ($SSG(d)$) of a finite planar set, which is a connected graph that varies depending on a single parameter d. We analysed how $SSG(d)$ relates to some well-known graph structures, and we presented algorithms to calculate $SSG(d)$. In addition, we presented how $SSG(d)$ can be used to improve the running time of the Shortest Path Graph ($SPG(t)$). We have empirically shown that $SSG(d)$ is both efficient and effective to visualize the possible movement network using LBSN data due to its distance based local evaluation criteria. Furthermore, we have shown the usefulness of $SSG(d)$ in a few applications.

Even though $SSG(d)$ is suitable for effectively analyse moderate number of posts, the running time increases when we try to process very large data sets. In the next chapter, we are going to look at another graph structure named the Diversion graph ($DG(d)$) that can process this increased location data load. New $DG(d)$ is created by relaxing the evaluation criteria of the Stepping Stone Graph. Due to this reason, $DG(d)$ contains few additional edges (less than 2% of total edge count) that violate the empty Diversion Neighbourhood criteria. However, the running time reduction is over 90%. Therefore, $DG(d)$ is a suitable alternative to processing increased amounts of location data.

Chapter 4

Diversion Graph For Real Time Movement Analysis

This chapter is derived from:

- Sameera Kannangara, Hairuo Xie, Egemen Tanin, Aaron Harwood, and Shanika Karunasekera. "Introducing Diversion Graph for Real-time Spatial Data Analysis with Location Based Social Networks." *11th International Conference on Geographic Information Science (GIScience 2021)*, Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2020.

4.1 Introduction

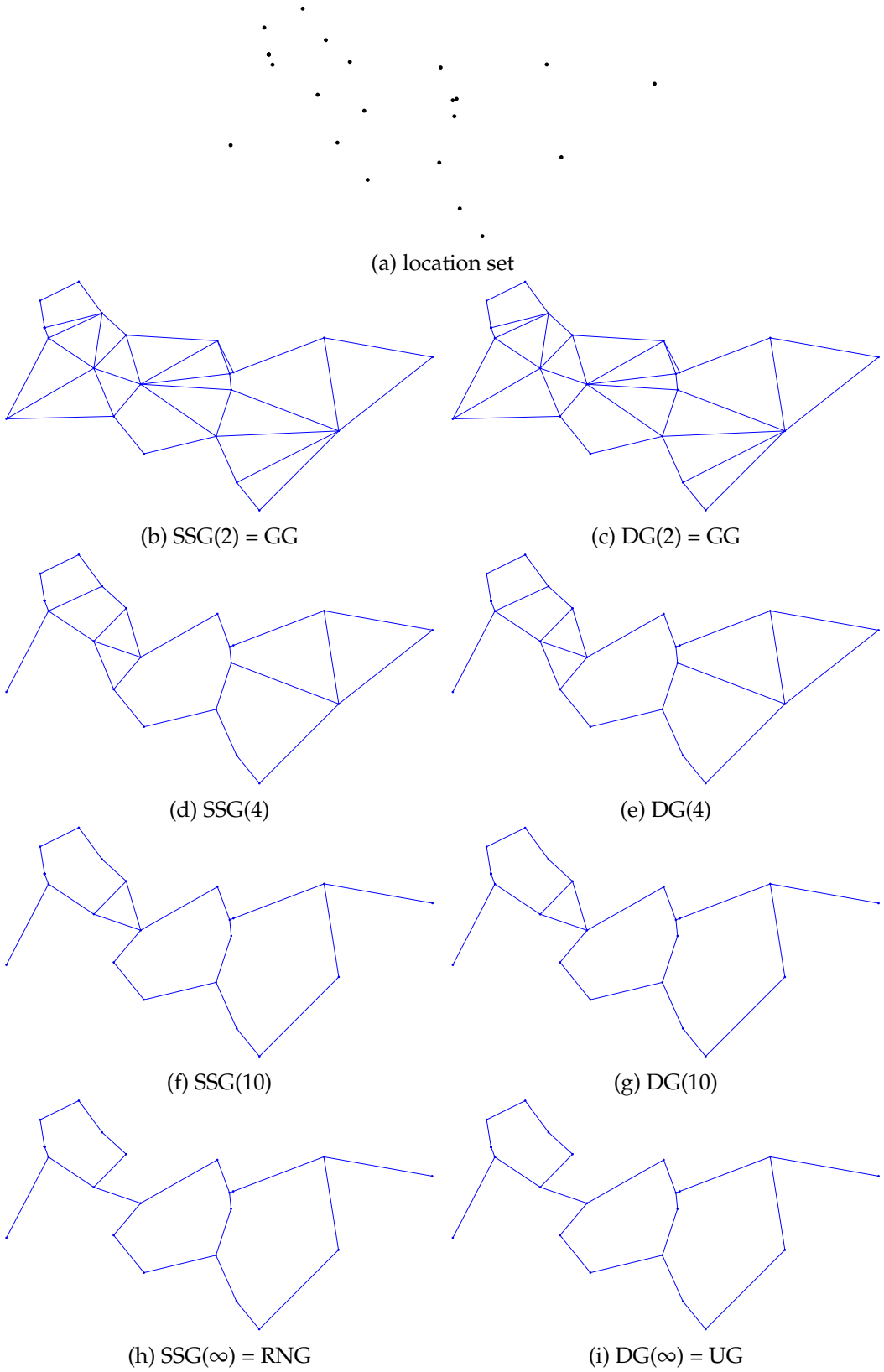
Modern LBSNs generate millions of data points for analysis[1]. Therefore when proposing methods to analyse LBSN data we need to think about the scalability of their executions. Although existing neighbourhood graphs such as the Stepping Stone graph proposed in the last chapter can process LBSN data, they are not suitable for large datasets due to the long running times. With the increase of availability of GPS enabled mobile phones the size of LBSN datasets will continue to be significantly large. Therefore we need to investigate efficient methods to process large amounts of location data collected from

[1]https://www.oberlo.com/blog/twitter-statistics

LBSNs.

In this chapter, we propose a new type of variable graph, which we refer to as the Diversion Graph ($DG(d)$). The skeletons inferred by $DG(d)$ are likely to be close to human perception of the corresponding point set. We show in our experiments that $DG(d)$ is easier and faster to build than $SSG(d)$, and gives similar results in processing movement analysis related queries. Similar to $SSG(d)$, $DG(d)$ is defined on top of DT and uses Diversion Neighbourhood ($DN(d)$) to cull edges from DT. Instead of checking all the points that lie in $DN(d)$ between two points $DG(d)$ only considers points in the neighbouring Delaunay triangles of the edge that is considered for culling. Thus, similar to $SSG(d)$ we assume that the social network data gives us partial data per individual user in terms of its path but with a good picture of where people could be in an event in a city from multiple users. As explained with the definitions of the $DG(d)$ below, for all endpoint pairs the value of d indicates the preference of inferring a longer alternative path with less distance between all point pairs on the path compared to the direct distance between the endpoint pair. This is useful when we have a very dense point set to cull some connections. Similar to both $SPG(t)$ and $SSG(d)$, as d increases the number of edges in $DG(d)$ monotonically decreases and therefore the path length between any two non-adjacent points in the skeleton monotonically increases. It is also important to note that GG is a special case of $DG(d)$ when $d = 2$.

We use publicly available LBSN data to evaluate the performance of $DG(d)$ and $SSG(d)$ for inferring movement networks. We show that $DG(d)$ performs as well as $SSG(d)$ in terms of the quality of the inferred network but $DG(d)$ achieves significantly faster execution times than $SSG(d)$. $DG(d)$ takes less than 10% of the time required to infer a movement network than $SSG(d)$. $DG(d)$ contains a few more edges (less than 2% of the total number of edges [13]) compared to $SSG(d)$. However, compared to the time advantage of $DG(d)$, the negative impact of the additional edges could be negligible. The resulting $DG(d)$ and $SSG(d)$ are very similar in terms of their shape and topology.



(a) location set

(b) SSG(2) = GG

(c) DG(2) = GG

(d) SSG(4)

(e) DG(4)

(f) SSG(10)

(g) DG(10)

(h) SSG(∞) = RNG

(i) DG(∞) = UG

Figure 4.1: SSG(d) and DG(d) skeletons created on a subset of locations from Twitter data set. Note that two skeletons in a row are created using two algorithms, but exhibit the same graph structure.

4.2 Diversion Graph

Given a set of points, a Diversion Graph ($DG(d)$) connects the points in a traversable manner.

4.2.1 Definitions

We construct $DG(d)$ in the form of an undirected graph $G(V, E)$ where $V \subseteq \mathbb{R}^2$ represents a given point set and E represents the inferred edges between the points. An edge between two endpoints $p, q \in V$ is represented as $pq \in E$. Length l_{pq} represents the Euclidean distance between the two points.

As $DG(d)$ is defined using Delaunay Triangulation (DT), let's briefly iterate a useful property of DT. DT is a triangulation of a point set, in which for each triangle's circumcircle does not contain any other points other than triangle's vertices. Also, DT is the dual of Voronoi diagram. Our proposed graph $DG(d)$ is evaluated by removing some edges from the DT according to the following criteria.

Definition 4.1 (Diversion Graph)**.** *For $V \subseteq \mathbb{R}^2$, the Diversion Graph of V at $d \in \mathbb{R} : d \geq 2$, denoted $DG(V, d)$ or simply $DG(d)$, is defined as an undirected graph with points V, subset of $DT(V)$ such that for each edge $pq \in DT(V)$:*

$$pq \text{ is an edge of } DG(V, d) \text{ iff } l^d_{pz} + l^d_{zq} > l^d_{pq}.$$

Where z is the other point in the Delaunay triangles where pq is an edge.

By this definition in common terms, $DG(d)$ is the graph created by removing edges pq from DT if and only if $l^d_{pz} + l^d_{zq} \leq l^d_{pq}$ where z is a vertex of a Delaunay triangle where pq is an edge. Therefore inherently $DG(d)$ is a subgraph of DT. Also due to the equation used to evaluate edges which are not used for movement for removal, there is a relationship between $DG(d)$ and $SSG(d)$. We explore these properties in the next section.

4.2.2 Diversion Graph Properties

Since $DG(d)$ is created by removing edges from DT, we can state the following theorem.

Theorem 4.1. *For* $2 \leq d, DG(d) \subseteq DT$.

Consider the case $DG(2)$ against GG. By comparing definitions of the two graphs we can state the following theorem.

Theorem 4.2. $DG(2) \equiv GG$.

Proof. Consider the definition of GG from [8]. The vertices $p, q \in V$ are least squares adjacent forming the edge $pq \in GG$ iff

$$l_{pq}^2 < l_{pz}^2 + l_{zq}^2 \text{ such that } \forall z \in V \setminus \{p, q\}.$$

Furthermore, in the same paper [8] it is proven that the GG can be extracted from DT by evaluating the above inequality on each triangle, for each edge. Now let's look at $DG(2)$ definition. It is extracted from DT by removing edges pq iff $l_{pz}^2 + l_{zq}^2 \leq l_{pq}^2$, for each z which are other points of the Delaunay triangles where pq is an edge. Since both GG and $DG(2)$ are evaluated from DT using the same inequality, they are equivalent. □

Since equation used in $DG(d)$ definition is same as the diversion neighbourhood definition, one may think $DG(d)$ and $SSG(d)$ are the same thing. Using a simple counter example we show that is not the case and later show that $SSG(d) \subseteq DG(d)$.

Theorem 4.3. *For* $2 \leq d, SSG(d) \subseteq DG(d)$.

Proof. Consider the five points p, q, r, s, t in Figure 4.2. Their Delaunay triangulation is shown in solid straight lines. Points p and q has equal y coordinates. Dashed line indicates diversion neighbourhood at $d = 4$ ($DN(4)$) between p and q. Both points r and s resides outside of shown $DN(4)$. Finally, t lies inside shown $DN(4)$. Let's consider inclusion of edge pq in $DG(4)$ and $SSG(4)$. Since t is within the $DN(4)$ of pq, it will not

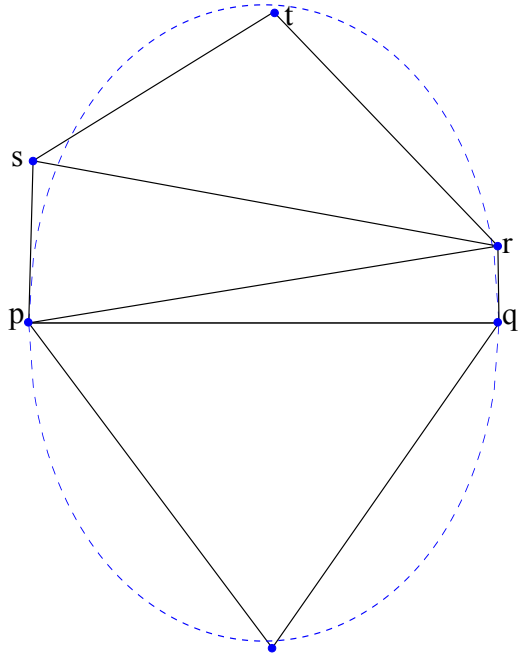


Figure 4.2: Counter example to show $DG(d)$ is not always equal to $SSG(d)$. Dashed line shows diversion neighbourhood at $d = 4$ ($DN(4)$). Points p and q has equal y coordinates. Both points r and s resides outside of shown $DN(4)$. Finally, t lies inside shown $DN(4)$.

be an edge of $SSG(4)$. However, in $DG(4)$ as it considers only points in neighbouring triangles of pq, it only considers point r when considering the inclusion of pq. Since r is outside the $DN(4)$ of pq, pq becomes an edge of $DG(4)$. Therefore, for $2 < d, DG(d)$ is not always equal to $SSG(d)$. □

In fact, $DG(d)$ can contain more edges than $SSG(d)$. Therefore we can state following corollary.

Corollary 4.1. *For $2 < d, DG(d)$ is not always equal to $SSG(d)$.*

Let's consider behaviour of $DG(d)$ when $d \to \infty$ and see how $DG(d)$ relates to UG.

Theorem 4.4. *As $d \to \infty, DG(d) \to UG$.*

Proof. When $d \to \infty$, by $DG(d)$ definition for an edge pq to be removed from DT, both other edges of the neighbouring Delaunay triangles only needs to be shorter than pq.

In other words, if pq is the longest edge in a Delaunay triangle it will be removed from $DG(d)$ when $d \to \infty$. UG is created from DT by removing the longest edge of each Delaunay triangle. Therefore, by the equality of definitions as $d \to \infty, DG(d) \to UG$. □

Theorem 4.5. *For $2 \leq d, DG(d)$ is planar and connected.*

Proof. Since for $2 \leq d, DG(d) \subseteq DT$, $DG(d)$ is planar. Inequality used in $DG(d)$ is the same as the diversion neighbourhood of $SSG(d)$. In chapter 3 section 3.2.2 it is shown that diversion neighbourhood does not get bigger than open lune neighbourhood. Open lune neighbourhood is proven as the tight neighbourhood that ensures a connected edge embedding in empty region graphs in [6]. Therefore $DG(d)$ is connected. Combining these arguments we can say for $2 \leq d, DG(d)$ is planar and connected. □

Next we consider the relationship between two $DG(d)$s as d increases.

Theorem 4.6. *For $2 \leq d \leq d', DG(d') \subseteq DG(d)$*

Proof. Define the *edge weight* of pq with respect to d' as $l_{pq}^{d'}$, for some $d' \geq 2$. Assume that for all $z \in \Lambda(pq) \setminus \{p, q\}$, $l_{pz}^{d'} + l_{zq}^{d'} > l_{pq}^{d'}$, where $\Lambda(pq)$ is the set of points in neighbouring Delaunay triangles of pq. In this case, pq is an edge in $DG(d')$. Now we show that for $d \leq d'$, pq is also an edge in $DG(d)$. Let us write $d = d'\epsilon$ where $\frac{2}{d'} \leq \epsilon \leq 1$. Then we need to show that:

$$
\begin{aligned}
l_{pz}^{d'\epsilon} + l_{zq}^{d'\epsilon} &> l_{pq}^{d'\epsilon} \qquad (4.1)\\
\frac{l_{pz}^{d'\epsilon} + l_{zq}^{d'\epsilon}}{l_{pq}^{d'\epsilon}} &> 1\\
\left(\frac{l_{pz}}{l_{pq}}\right)^{d'\epsilon} + \left(\frac{l_{zq}}{l_{pq}}\right)^{d'\epsilon} &> 1\\
\left(\left(\frac{l_{pz}}{l_{pq}}\right)^{d'\epsilon} + \left(\frac{l_{zq}}{l_{pq}}\right)^{d'\epsilon}\right)^{\frac{1}{\epsilon}} &> 1
\end{aligned}
$$

Algorithm 2: Create $DG(d)$

Input: V - Filtered locality set
d - Configuration parameter
Output: $DG(d)$

```
1  DT ← create Delaunay Triangulation of V
2  initialize DG(d) to empty set
3  foreach (Edge pq : pq ∈ DT) do
4      foreach (Point z : z ∈ Λ(pq) \ {p, q}) do
5          if (l^d_pq < l^d_pz + l^d_zq) then
6              Add pq to DG(d)
7          end
8      end
9  end
10 return DG(d)
```

Since the function $x \mapsto x^\beta$ is subadditive for $\beta \geq 1$ then:

$$\left(\left(\frac{l_{pz}}{l_{pq}}\right)^{d'\epsilon} + \left(\frac{l_{zq}}{l_{pq}}\right)^{d'\epsilon}\right)^{\frac{1}{\epsilon}} \geq \left(\frac{l_{pz}}{l_{pq}}\right)^{d'} + \left(\frac{l_{zq}}{l_{pq}}\right)^{d'}.$$

We know the right hand side is greater than 1 due to our initial assumption and therefore Eq. 4.1 is true. Therefore pq is also an edge in $DG(d)$ and this completes the proof. □

4.2.3 Algorithms

In this section, we present an efficient algorithm to compute $DG(d)$. Since $DG(d)$ is defined based on DT we can use DT as the starting graph to compute $DG(d)$. There are two approaches we can use to compute $DG(d)$ using DT. One approach is to process DT as triangles and check each edge of the triangle for removal from DT create $DG(d)$. The second approach is to process DT as a set of edges and evaluate each edge against the points in the neighbouring Delaunay triangles to check whether they belong in $DG(d)$. The approach we are presenting in this chapter is the second approach which evaluates edges to check their membership of $DG(d)$, as it can be easily compared with the d-spectrum algorithm of the $SSG(d)$.

We propose a simple and effective algorithm to calculate $DG(d)$. In the algorithm, $\Lambda(pq)$ is the set of points in neighbouring Delaunay triangles of pq. Each other point in the $\Lambda(pq) \setminus \{p, q\}$ are evaluated against pq to see whether pq is an edge of $DG(d)$. For simplicity, the condition in line 5 in Algorithm 2 is directly derived from the definition of $DG(d)$. However, it can be further improved by checking whether other edges connected with $\Lambda(pq)$ are longer than pq. The algorithm is readily parallelizable as there is no race condition between separate edge evaluations.

Let us consider the time complexity of the proposed algorithm for calculating $DG(d)$ provided DT for a set of n planar points. In line 3, as DT has $\mathcal{O}(n)$ edges, the code between line 4 and line 8 runs $\mathcal{O}(n)$ times. As each edge pq has at most two neighbouring triangles, the code between line 5 and line 7 runs at most two times per edge. Line 5 is assumed to run in $\mathcal{O}(1)$ time. Therefore, the whole algorithm runs in $\mathcal{O}(n)$ time.

4.2.3.1 Improving Running Time of $SSG(d)$

Introduction of $DG(d)$ allows us to efficiently calculate $SSG(d)$ for a specific $2 \leq d$ value without calculating d-Spectrum. Since $DG(d)$ is a super graph of $SSG(d)$ for $2 \leq d$, once $DG(d)$ is calculated we can use it to evaluate those edge using the triangle sweeping method presented in chapter 3 Algorithm 1. As later shown in the experiments $DG(d)$ only contain a very small percentage of additional edges compared to $SSG(d)$. Therefore this is a very efficient method of computing $SSG(d)$.

However, it should be noted that computing $SSG(d)$ from $DG(d)$ may be slower for varying skeleton generation compared to using d-Spectrum. Since d-Spectrum pre-computed the minimum d-value necessary for an DT edge to be in the $SSG(d)$, varying skeleton generation takes less time. But for generating $SSG(d)$ for a specific $2 \leq d$ using $DG(d)$ is faster than creating d-Spectrum.

4.2.4 Applications

The proposed $DG(d)$ can be used in many applications detailed as follows.

4.2.4.1 Nearest Neighbour Queries

The $DG(d)$ graph structure can be used to search for the path to the nearest interesting locations from a given location. For example, this kind of query can be used to find the nearest exit gate in a park. We can use breadth first search starting from the query location and traverse the graph until a required interesting point is found.

Similarly, we can use $DG(d)$ to search for the k-nearest neighbours. Instead of stopping breadth first search when the first interesting point is found, it can be continued until k interesting points are found. As for the edge weights, we can use weights calculated in the section 4.2.4.4 according to the usage of edges. This will make sure that the most popular path to the nearest neighbour will be found. This approach can be extended to solve reverse nearest neighbour queries and group nearest neighbour queries as suggested in chapter 3.

4.2.4.2 Refinement of Movement Corridors

Once $DG(d)$ is created using posted localities, user trajectories can be used to refine the created travel network. The approach for refining the travel network is as follows. For each consecutive location pair in user trajectories, the shortest path is determined using $DG(d)$. With each $DG(d)$ edge the number of trajectories passed through that edge is recorded. We can then represent movement corridors in the travel network based on the edges that contain trajectory count higher than a given threshold.

Definition 4.2 (Usage counter)**.** *When path is a sequence of edges traversed by a trajectory*

trace, for all $pq \in E$, Usage Counter of pq (denoted $UC(pq)$), is defined as the trajectory count,

$$UC(pq) = |\{path : pq \in path\}|$$

One of the problems of using $DG(d)$, as it is for movement network analysis, is that it does not consider the existence of obstacles. As trajectories do not appear on obstacles such as rivers, incorporating trajectory information into $DG(d)$ allows filtering edges not used by the trajectories. By filtering edges not used by trajectories, we are able to eliminate edges that do not represent user movement information. In summary, refined movement corridors calculated using $DG(d)$ is an edge subset of $DG(d)$ which are used by the trajectories for movement.

4.2.4.3 Inferring Road Network

The aforementioned approach for refining movement corridors can be used to infer the road network in an area where we do not have prior knowledge about the road networks. Ideally for this purpose, we need GPS locations published on the road network. The easiest way to obtain such information is to collect LBSN post published while travelling in vehicles. Using the GPS data in LBSN posts and associated the GPS points with trajectories, we can infer the road network in the given area.

4.2.4.4 Most Popular Paths

After calculating usage counters of $DG(d)$ edges, they can be used to find the most popular path between locations. To find the most popular path, we calculate edge weight to reflect the popularity of the edge and use the shortest path algorithm to calculate the paths. To ensure edges with more usage have lower weights, we divide the length of the edge by usage counter of that edge. For edges with no usage, edge length multiplied by a constant greater than 1 is used as the edge weight. After calculating edge weights in

this manner, the Dijkstra's shortest path algorithm is used to find the most popular path between two locations.

Definition 4.3 (edge weight)**.** *For all $pq \in E$, weight of pq is defined as,*

$$weight(pq) = \begin{cases} l_{pq}/UC(pq), & if\, 0 < UC(pq). \\ l_{pq} \times C : 1 < C, & if\, UC(pq) = 0. \end{cases}$$

4.2.4.5 Other Applications

$DG(d)$ can be used in other applications such as tour recommendation, trajectory clustering and group movement detections as mentioned in chapter 3. For all these applications we have to process user trajectories after creating the initial graph structure to incorporate additional movement related information to the created graph skeleton.

4.3 Experiments

4.3.1 Data Sets

We conducted experiments on two real world LBSN datasets and one synthetic data set. The first real data set consists of geo-located posts collected from Twitter. It is collected from 06th March to 23rd April 2012, within a bounding box over the countries Australia and New Zealand. It contains 724651 LBSN posts authored by 36639 users. For our analysis, an LBSN post is defined as a tuple containing four elements - userID, voluntarily generated textual content, timestamp and the location where the post was authored.

We used Yahoo! Flickr Creative Commons 100M (YFCC100M) dataset [117] as the second real dataset. It contains metadata such as user information, timestamp and location of 100 million photos and videos shared on Flickr. Only the entries with point

geo-locations were used for our experiments. For movement corridor experiment localities in London, England are filtered out from the YFCC100M data set.

The synthetic data set for our experiments was generated using SMARTS simulator [118]. We simulated vehicle movement in the Melbourne central business district (CBD) and collected GPS locations of the vehicles every 0.5 seconds. the data set used for our experiment contained 100000 GPS points.

4.3.2 Implementation

To visualize the inferred neighbourhood skeletons, a visualization tool was implemented utilizing GeoTools[2] Java libraries. All the skeleton visualizations presented in this chapter are generated using this tool. Both $DG(d)$ and $SSG(d)$ algorithms are implemented using Java 8. The datasets are stored in a MongoDB database.

To infer $DG(d)$, firstly, DT is created using SweepHull [119] algorithm. Then, $DG(d)$ is calculated using Algorithm 2. $SSG(d)$ is extracted from the planar d-Spectrum created using DT. We used numerical analysis to calculate d-value of an edge. More specifically, a Java method was implemented to perform Secant method[3] to approximate the d-Value. The same DT was reused for construction of $DG(d)$ and $SSG(d)$ with different d values. We selected 3 as the configuration value for d to compare resulting graphs generated using $DG(d)$ and $SSG(d)$ based on preliminary tests.

[2]http://www.geotools.org/
[3]https://en.wikipedia.org/wiki/Secant_method

4.3.3 Results

4.3.3.1 Event Analysis

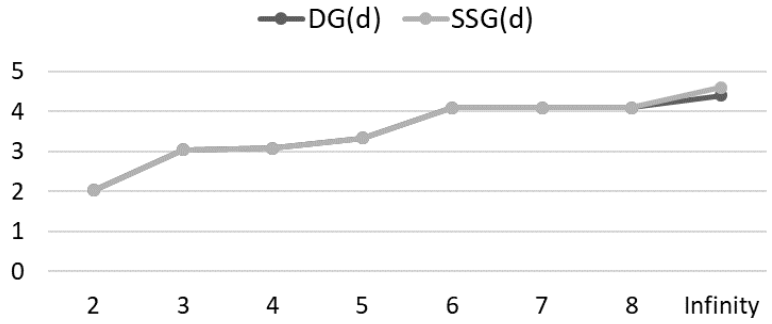
In this experiment, we use a Twitter dataset relating to Queensland state election 2012 [4]. All users participating in the event are there for a common reason and exhibit a similar movement pattern. We implement an LBSN post filtering technique used in chapter 3 experiments, to filter posts relating to the election. For temporal bound of the dataset, we took the time period between 23rd and 26th of March 2012. As for the spatial bound, we considered a bounding box over the Queensland state. We consider all users who have posted with "#qldvote" hashtag within spatial and temporal bounds of the event. The data set contains, 1270 unique points after filtering the original Twitter data.

We generate skeletons with $DG(d)$ and $SSG(d)$ with different settings of d. The number of edges and the spanning ratio [115] of the graph structure are plotted with varying configuration parameters (Figure 4.3). Spanning ratio of a graph indicates the maximum ratio between the shortest path distance over the graph and direct distance between any point pair. Therefore, graphs with low spanning ratio are preferred to represent movement networks [115].

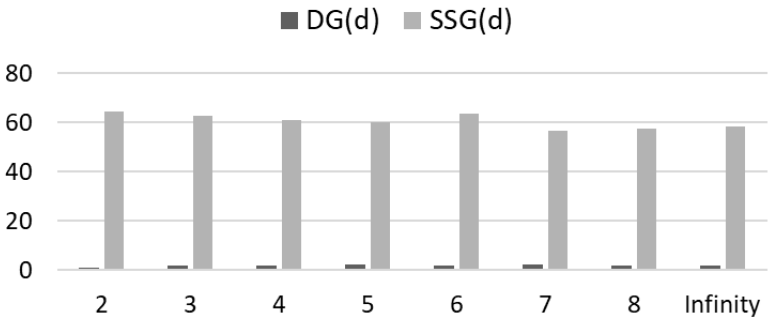
Figure 4.3 (a) shows the variation of spanning ratio as configuration parameter varies to demonstrate how the shortest path distances between locality pairs change. Both $DG(d)$ and $SSG(d)$ has a low and stable spanning ratio, making them suitable for movement analysis. Furthermore, both $DG(d)$ and $SSG(d)$ have the same spanning ratio when d is less or equal to 8.

The time taken to calculate skeletons of $DG(d)$ and $SSG(d)$ are shown in Figure 4.3 (b). Execution time for $DG(d)$ calculation is around 95% less compared to $SSG(D)$ for all configuration values. Relaxed nature of $DG(d)$ algorithm gives it a significant advantage in computation time.

[4] https://en.wikipedia.org/wiki/Queensland_state_election,_2012



(a) Spanning ratio with varying configuration parameter d

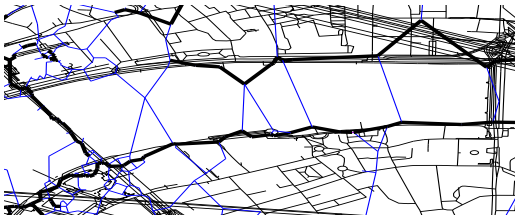


(b) The time to generate the graphs in mili seconds with varying d

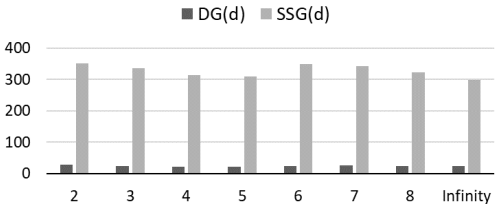
Figure 4.3: Graphs depicting different properties between $DG(d)$ and $SSG(d)$.

4.3.3.2 Movement Corridors Refinement

The refined movement corridors refer to the edges of the graph that are used by the users on the move. These edges are selected by aligning user trajectories along the graph edges using shortest path calculation. We analyse the refined movement corridors relating to the trajectories filtered from YFCC100M dataset, which are around the Thames river in London. We take locations posted over a month. The total number of locations is 6318. To represent the travel networks, $DG(3)$ and $SSG(3)$ are used. This data set is selected because it has higher randomness in tourist movement compared to the Twitter data set. After that trajectories are aligned to both graph skeletons, and all the edges with usage counter more than 5 are filtered as refined movement corridors. That is, if an edge is used by 5 or more trajectories, the edge is selected as a refined movement corridor. Both graphs resulted in the same refined movement corridor structure (Figure 4.4 (a)). Edges



(a) Thick lines indicate refined movement corridors extracted using $DG(3)$ and $SSG(3)$



(b) Execution time in mili seconds to generate each graph with varying d

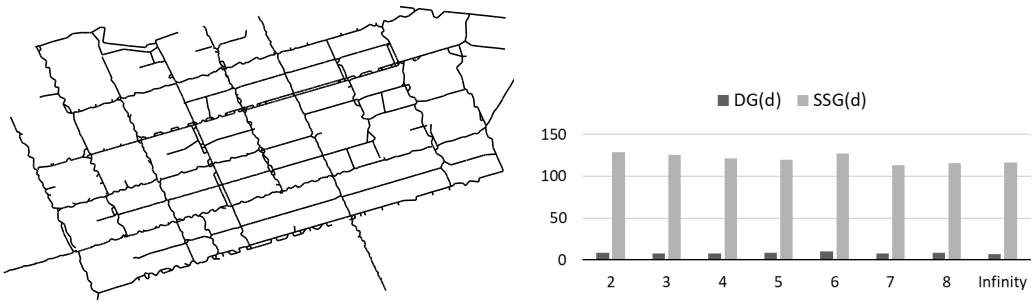
Figure 4.4: Results of movement corridor refinement.

created across the river are filtered out as there cannot be any movement on the edges. Figure 4.4 (b) shows the execution times taken to calculate the graph structures. We can see that the time for creating $DG(d)$ is only 8% of the time taken to create $SSG(d)$.

4.3.3.3 Road Network Inference

Refined movement corridor extraction can be used to infer the road network of an area. Using our synthetic dataset we infer the road network of the Melbourne CBD. In order to make this data sparse similar to LBSN data, we filtered out some of the points in the synthetic dataset. The filtering process first sorts all GPS points based on the timestamp and then takes one point for every n points from the sorted set. There are 2763 locations collected for this experiment.

In Figure 4.5 (a), we show the road networks inferred using $DG(3)$ and $SSG(3)$. The two road networks almost totally overlap with each other. It should be noted that as the filtering parameter n grows, the data set used to infer road network become more sparse, degrading the result road network. Results heavily degrade when n reaches around 120. Figure 4.5 (b) shows the execution times taken to calculate the graph structures. We can see that $DG(d)$ creation takes 7% of the time that is used for creating $SSG(d)$.



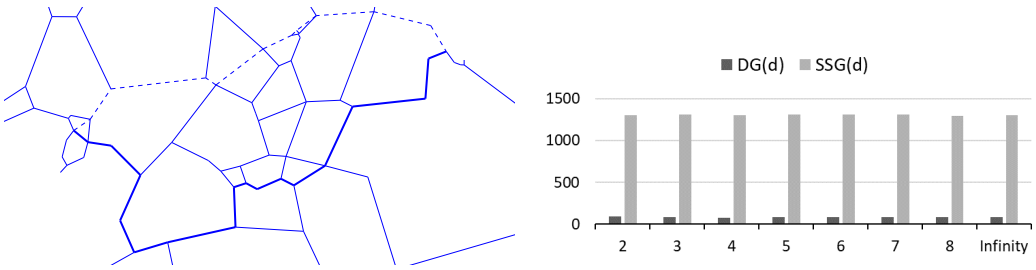
(a) Road network extracted using $DG(3)$ and $SSG(3)$

(b) Execution time in mili seconds to generate each graph with varying d

Figure 4.5: Results of road network extraction.

4.3.3.4 Most Popular Path Finding

By calculating edge weights to reflect the popularity of edges we can use the resulting graph to calculate the most popular paths. We used Twitter data set to run experiments on extracting most popular paths. This data set was used because it contains users with regular movement patterns. We executed the experiment in Melbourne city area where we found 28431 locations. Figure 4.6 (a) shows a most popular path found between two locations. Dashed lines indicate the shortest path between the two locations while the thick lines indicate the most popular path. When comparing this result with a map there are roads along the most popular path detected while there are building on top of the shortest path. Overall 78% of the edges selected for most popular path calculation were sitting on the road network of the Melbourne city. Figure 4.6 (b) shows the executions times taken by $DG(d)$ and $SSG(d)$ to create graphs. Due to the size of the dataset $SSG(d)$ has resulted in running times longer than one second. However, $DG(d)$ has generated the graph in 7% of the time taken by the $SSG(d)$, making it suitable for processing large data sets in real time.



(a) Most popular path extracted using $DG(3)$ and $SSG(3)$ is shown in thick lines

(b) Execution time in mili seconds to generate each graph with varying d

Figure 4.6: Results of most popular path extraction.

4.4 Discussion

From our experiments, it is clear that given a point set $DG(d)$ creation takes less time compared to $SSG(d)$ creation. Also, created $DG(d)$ can be used with similar effectiveness as the $SSG(d)$. It is worthwhile to consider the reason behind $DG(d)$ improved creation time. Firstly, per considering edge $DG(d)$ creation algorithm (Algorithm 2) only process two triangles. However, for $SSG(d)$ it can be empirically found out that per edge at least three triangles are processed. This effect should give a $2:3$ advantage to $DG(d)$ creation compared to $SSG(d)$. However time difference we see in above experiments are $1:10$ between $DG(d)$ and $SSG(d)$. Reason for this massive difference is behind the numerical analysis method used to evaluate the d-value of an edge for $SSG(d)$. In $DG(d)$'s case only a simple inequality is evaluated based on Definition 4.1, per processing triangle. For d-spectrum calculation method of $SSG(d)$, per processing triangle numerical analysis method runs to determine the least empty diversion neighbourhood around the edge. In our implementation, it is the Secant method and it is executed loop that results in hundreds of iterations. Therefore we experience this massive creation time difference between $DG(d)$ and $SSG(d)$. Due to this reason, it is advantageous to use $DG(d)$ in applications where very little processing time is available to generate results. This also highlights the need for looking into faster ways to locate the d-value for $SSG(d)$, as future work.

In the above experiments, we have used data sets with few thousands of locations.

As the dataset size increases one may think we can apply techniques applicable on dense GPS data. Even though these datasets are large their locations are spread across large areas making their density sparse. Due to this reason techniques applicable on dense GPS data does not generate meaningful results on these sparse datasets.

4.5 Summary

We presented the Diversion Graph ($DG(d)$), a connected graph that varies depending on a single parameter d. We analysed how $DG(d)$ relates to some well-known graph structures, and we presented how $DG(d)$ can be used to improve running time of the state-of-the-art graph, the Stepping Stone Graph ($SSG(d)$). We have empirically shown that $DG(d)$ is both efficient and effective to analyse LBSN data due to its distance based local evaluation criteria.

In the last two chapters, we proposed graph data structures to analyse LBSN data. Next problem we are going to look at is combining location data from different users. This is a difficult problem as different users have different posting frequencies. The use case we are going to look at for combining location information is group movement prediction. To do group movement prediction using LBSN data we propose a technique named the Group Kalman Filter, by extending the well known signal processing technique Kalman filter.

Chapter 5

Group Movement Analysis And Prediction

This chapter is derived from:

- Sameera Kannangara, Hairuo Xie, Egemen Tanin, Aaron Harwood, and Shanika Karunasekera. "Tracking Group Movement with Location Based Social Networks." *Proceedings of the 28th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 2020. (Accepted).

5.1 Introduction

In this chapter, we are exploring techniques to combine location information available from different LBSN users for group movement analysis. This is a difficult task as different users have different posting patterns. The use case we are going to look at regarding location data combining is group movement prediction. Group movement prediction is important to many application domains such as crowd management and law enforcement. Tracking the movement of groups involves estimating the current position of the groups and predicting the future position of the groups. The effectiveness of group tracking depends on the availability and accuracy of location data. The traditional manner of collecting location data relies on application-specific monitoring systems, such as GPS

recorders on taxis. However, due to the high cost of installing the devices and privacy concerns, using monitoring systems can limit the scalability of data collection. Location Based Social Networks (LBSNs) are becoming an important source for collecting location data. Compared to purpose-built monitoring systems, LBSNs can be more privacy-aware as the users have full control of when and where to post about their activities. In addition, the cost of data collection can be significantly lower with LBSNs. We are interested in tracking the movement of groups by analysing the trajectory data obtained from LBSNs, which shows the location of individual users at specific times.

Group tracking with LBSN data can be challenging as the quality of the data is affected by various dynamic factors. One dynamic factor is **group entity**, which is the set of group members that may change frequently as people join or leave a group. The **spatial extent** occupied by a group is also dynamic as the group may change its moving direction, speed and spread over time. The quality of LBSN data can also be affected by the randomness in **posting time** as the data is generated voluntarily at irregular times, making data processing more difficult with traditional trajectory processing mechanisms [122].

To effectively track the movement of groups based on LBSN data, a tracking system needs to handle the noise in the location data caused by the aforementioned dynamic factors. We use the well-known Kalman filter algorithm [123, 124] for this purpose. Kalman filter is commonly used for estimating the state of a system under the influence of linear processes in discrete time. Kalman filter has been used in tracking individual moving objects. By executing Kalman filter at consecutive steps, the estimation of the location becomes more accurate if the object moves at a stable velocity. If the object makes a sudden change of its velocity, Kalman filter is able to make estimations that follow the change in future steps.

Although Kalman filter has been used for tracking individual moving objects [125], it has not been used for tracking groups with the aforementioned dynamic factors. We propose a first-of-its-kind solution for tracking the movement of groups based on crowd sourced trajectory data. Our solution, named Group Kalman Filter (GKF), can deal with

the randomness caused by all the aforementioned dynamic factors. GKF can predict the group entity, which allows a tracking system to focus on the locations submitted by the users who are likely to stay in a group during a specific period. GKF has the ability to predict the space covered by the groups. GKF can also predict the posting time, which can help capture the changes of group movement in a timely manner.

A complete GKF-based solution would have three sub-solutions, each addresses one of the aforementioned dynamic factors. Due to the increase complexity of the solution, it is not possible to show it in its entirety. In this chapter, we focus on the second dynamic factor, the spatial extent of groups, which we think is the most important one among the three factors for group tracking. Although the presented implementation does not predict group entity and posting time, it considers the most recent changes related to these two factors to mitigate their impact on predicting the spatial extent of groups. The sub-solutions that are catered for predicting group entity and posting time can be addressed in an extension to the presented implementation.

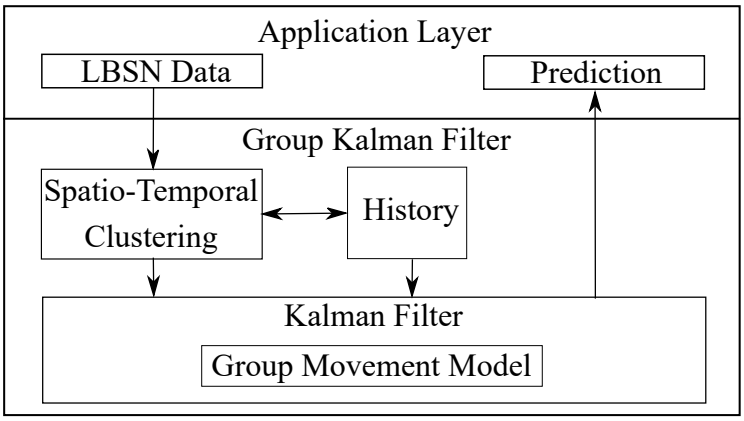
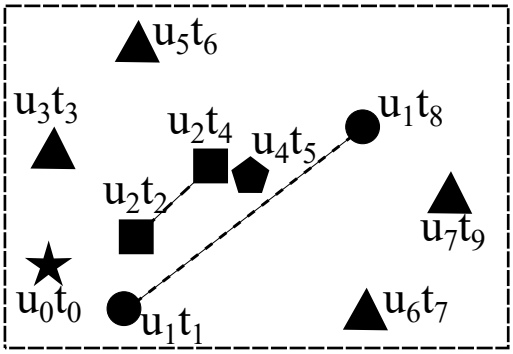


Figure 5.1: High-level architecture of the proposed solution for tracking groups based on LBSN data.

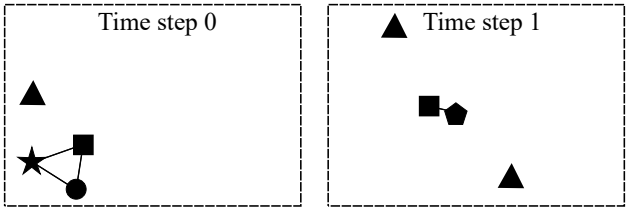
The architecture of our proposed solution is illustrated in Figure 5.1. GKF has a spatio-temporal clustering module for identifying the existence of groups within specific time windows. There is also a history module that keeps certain information about group entities at different time steps. These two modules work together to help mitigate the impact of the uncertainty in the group entity. For example, they can identify com-

plex movement scenarios such as group-merging and group-splitting. They also help mitigate the impact of the uncertainty in posting time in LBSN data. The Kalman filter module predicts the group locations, which are the minimal circular areas covering individual groups, based on a group movement model. This module addresses the uncertainty in the spatial extent of groups by modelling bounds of noises associated with the location data from LBSNs. GKF is inspired by the traditional Kalman filter but it has a unique characteristic. The traditional Kalman filter is stateless, which means predictions are made based on the most recent measurement. GKF can use the states from an arbitrary number of consecutive steps in the past, thanks to the history module. With this characteristic, GKF can perform better than the regular Kalman filter in tracking the movement of groups.

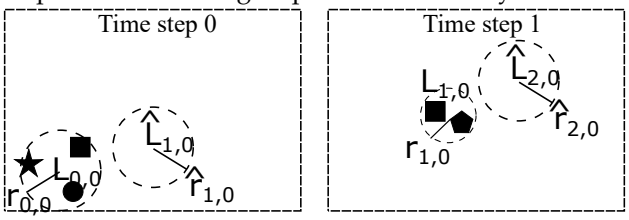
The workflow of tracking groups with GKF can be shown with the following example (Figure 5.2). Figure 5.2 (a) shows the locations that individual users posted to an LBSN. The locations are marked as $u_i t_j$, where u_i represents a user ID and t_j represents the time when a user submitted a location to the LBSN. We can see that there are eight unique users, u_0 to u_7, who reported their locations in ten timestamps, t_0 to t_9. Given the example data set, GKF first divides the whole period that covers the timestamps of all the records into multiple consecutive time steps in equal length. Let us assume that time step 0 includes four timestamps (t_0 to t_3), time step 1 includes four timestamps (t_4 to t_7) and time step 2 includes two timestamps (t_8 to t_9). Figure 5.2(b) shows the reported locations in the first two steps. GKF identifies the groups that present at a step based on the posted locations at the step. For example, in step 0, GKF finds a group with three users. In step 1, GKF finds a group with two users. Although the two groups are of different size, they are the variations of the same group as both groups contain user u_2 (represented as a square). At each time step, GKF makes an estimation of the current group location and predicts the group location for the next step. This process is shown in Figure 5.2(c), where an estimated group location is shown as a circle that is centred at $L_{k,i}$, where k indicates the time step and i indicates the group ID. The radius of the circle is $r_{k,i}$. Based on the estimation of the current group location, GKF predicts the centroid and the radius of the circle for the next time step. The predicted information is shown by $\hat{L}_{k,i}$ and $\hat{r}_{k,i}$,



(a) The location data obtained from an LBSN. The locations are marked as $u_i t_j$, where u_i represents a user ID and t_j represents the time when the user posted the location to the LBSN. The locations posted by the same user are connected with a dashed line.



(b) The detected groups after spatio-temporal clustering of the locations. Time step 0 covers four posting times, t_0, t_1, t_2 and t_3. Time step 1 covers four posting times, t_4, t_5, t_6 and t_7. One user group is detected at each time step by clustering the locations at the time step. The users of a group are connected by lines.



(c) The estimated group location and the predicted group location. An estimated group location is shown as a circle that is centred at $L_{k,i}$, where k indicates the time step and i indicates the group ID. The radius of the circle is $r_{k,i}$. A predicted group location is shown as a circle centred at $\hat{\mathrm{L}}$ with a radius $\hat{\mathrm{r}}$.

Figure 5.2: High-level process of identifying and predicting the location of a user group with GKF based on LBSN data. Triangles represent users who are not in any group.

respectively. For example, at step 0, GKF predicts that the group will move to the right at step 1 without a change of the radius. By combining the prediction from step 0 (the circle with centroid $\hat{L}_{1,0}$ and radius $\hat{r}_{1,0}$ in the left sub-figure of Figure 5.2(c)) and the location data posted at step 1 (u_2t_4 and u_4t_5 in the right sub-figure of Figure 5.2(b)), GKF makes an estimation of the group location at step 1 (a circle centred at $L_{1,0}$ with a radius $r_{1,0}$). As we can see that the estimated group location at step 1 is different to the predicted location from step 0 for this example. At step 1, GKF predicts that the group will move in the upper-right direction and expand in the next step, shown with a circle centred at $\hat{L}_{2,0}$. The prediction will be considered when estimating the group location at step 2.

User locations represented by the triangles in Figure 5.2 (a) and (b) are from the users of the groups that exist in only one time step. We do not show these groups in Figure 5.2 (c).

The contribution of this chapter includes: 1) We proposed a first-of-its-kind concept, GKF, for addressing the impact of noisy crowd sourced location data in tracking group movements; 2) We extended the traditional stateless Kalman filter by considering the states in multiple time steps in GKF; 3) We implemented several variations of GKF and evaluated their effectiveness in tracking group movements.

5.2 Problem Definition

Definition 5.1. *An **LBSN post set** is a data set $S = \{tp_1, ..., tp_n\}$, where tp_i is a tuple $< u, t, lat, lon >$, where u is a user ID, t is a timestamp, lat is a latitude and lon is a longitude.*

Definition 5.2. *The length of a time step is **time step size**, $\Delta t = T/k$, where T is the whole period covering the posts in S and k is the number of time steps. We denote the subset of S at time step ts as S_{ts}. The relationship between k, S_{ts} and S is as follows: $\forall ts_1, ts_2 \leq k \wedge ts_1 \neq ts_2, S_{ts_1} \cap S_{ts_2} = \emptyset$, and $\bigcup_{ts=1}^{k} S_{ts} = S$.*

Definition 5.3. *A **post group** is a cluster of LBSN posts submitted within one time step, where the location of a post (the GPS coordinates in the post) is within a distance threshold of d to the*

location of any other post in the group. A post group within a time step is denoted as $G_{ts,i}$ where ts is the time step and i is the group ID. Assuming the total number of mutually exclusive groups at time step ts is g, we have $\forall i, j \leq g \wedge i \neq j, G_{ts,i} \cap G_{ts,j} = \emptyset$ and $\bigcup_{i=1}^{g} G_{ts,i} = S_{ts}$.

Definition 5.4. *The **location of a post group** $G_{ts,i}$ at time step ts is the circular area that is centred at a point $L_{ts,i}$ with a radius $r_{ts,i}$, which is the maximum distance between the centroid and the location of any post in the group. In other words, the location of a group is the minimal circular area covering the whole group.*

Definition 5.5. ***Group entity** $U_{ts,i}$ is the set of user IDs in the posts from the post group $G_{ts,i}$.*

Definition 5.6. ***Group entity similarity** (GES) is a measure of the similarity between the entity of two groups, each of which presents in one of two consecutive time steps. Let $G_{k-1,i}$ be a post group i from time step $k-1$ and $G_{k,j}$ be a post group j from time step k. The GES between the two groups is defined as follows:*

$$GES(G_{k-1,i}, G_{k,j}) = \frac{|U_{k-1,i} \cap U_{k,j}|}{|U_{k,j}|} \tag{5.1}$$

Definition 5.7. ***Group entity similarity threshold** (GES threshold) controls the minimum level of similarity between two related groups in two consecutive time steps. Given a GES threshold C, where $0 < C \leq 1$, we say that $G_{k-1,i}$ and $G_{k,j}$ are related to each other if $GES(G_{k-1,i}, G_{k,j}) \geq C$.*

Definition 5.8. ***Entity change interval**, $I \geq \Delta t$, where Δt is the time step size (Definition 3.2), is the average time interval at which group entity changes for one group, that is, $U_{ts,i} \neq U_{ts+I,i}$.*

Problem Statement. Assume there are n users that move as groups and post to an LBSN. These groups can change entities at an average time interval of I. Assume that the LBSN posts made by the users are collected in consecutive time steps of size Δt. Assume that users may not post their locations at all consecutive time steps. Given the posts made at two time steps, $k-1$ and k, the problem is to find the groups that exist in both steps based on a group entity similarity threshold C, find the new groups that form at step k, estimate the locations of all the groups at step k and predict the locations of the groups

for step $k+1$ such that the difference between the predicted locations for step $k+1$ and the actual locations at step $k+1$ is minimized.

5.3 Group Kalman Filter

In this section, we present Group Kalman Filter (GKF) for solving the tracking problem described in Section 5.2. GKF is tailored for tracking the movement of groups based on the crowdsourced trajectory data from LBSNs. GKF consists of three modules, a spatio-temporal clustering module, a history module and a Kalman filter module. The detailed architectural overview of GKF is shown in Figure 5.3. We describe each module in the rest of the section. An example of the workflow of GKF is shown in Figure 5.2.

5.3.1 Spatio-Temporal Clustering Module

Given a LBSN post set (Definition 5.1), GKF first needs to find the groups that travel together before estimating the location of the groups. The spatio-temporal clustering module finds the groups in three steps. We detail each of the steps in this subsection.

5.3.1.1 Time Discretizing

To detect groups using LBSN data, we first divide the period, which covers the timestamps of all the LBSN posts, into k equal-sized time steps (an example is shown in Figure 5.2 (b)). The **time step size**, Δt (Definition 5.2), should be adjusted based on specific application scenarios. Even though we divide time into equal-sized time steps for this work, it should be noted that a solution can be developed with a dynamic division of time.

5.3.1.2 Group Detection using Spatial Clustering

In order to detect the post groups (Definition 5.3) at any time step ts, we use SMTIN algorithm proposed in [126] to divide the locations posted during ts into a set of mutually exclusive groups. The algorithm works by generating a Delaunay triangulation of all posted locations in the same step and filtering out the edges that are longer than a **distance threshold**, d. As shown in the experimental results (Section 5.5.1.2), choosing a suitable value for d can help improve the effectiveness of clustering.

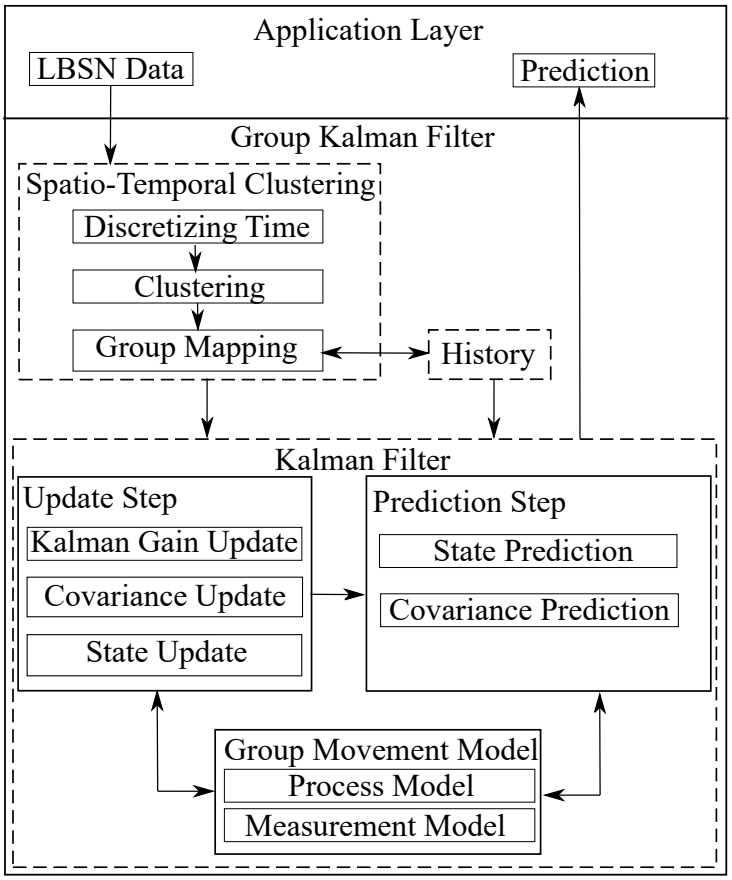


Figure 5.3: Detailed architecture of Group Kalman Filter.

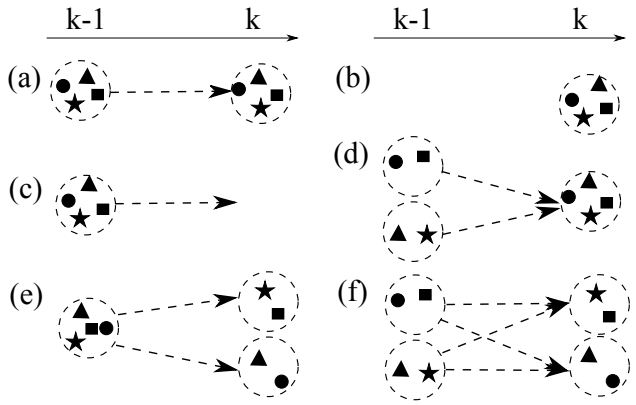


Figure 5.4: Group entity changes from time step $k-1$ to k: (a) no change (b) group appearing (c) group disappearing (d) merging (e) splitting (f) splitting and merging at the same time. Group members are shown as various shapes.

5.3.1.3 Group Mapping between Consecutive Time Steps

To track the movement of groups, we need to identify the groups that persist in consecutive time steps. Figure 5.4 illustrates various situations that can occur to the group entities (Definition 5.5) between two consecutive time steps $k-1$ and k. Figure 5.4(a) shows an occasion where a group remains unchanged across the two time steps. We need to keep tracking the group at time step k. Figure 5.4(b) depicts a situation where a group forms at step k. Figure 5.4(c) shows a situation where a group disappears. Figure 5.4(d) illustrates a merging situation where two user groups become one group. Figure 5.4(e) shows a group-splitting situation. The change of group entities can be complex in the real world when more than two groups are involved. For example, Figure 5.4(f) shows that two groups are transformed into another pair of groups. In all the situations except the situation in Figure 5.4(a), we stop tracking the groups that present in time step $k-1$ and start tracking the groups that present in time step k.

In order to find the groups that persist across consecutive steps (Figure 5.4(a)), we first need to narrow down the search to the groups that present at different time steps and are related to each other. Given a pair of related groups, we can find out whether the two groups are likely to be the same group by comparing their entity sets. The relationship between the groups that are mapped between consecutive time steps can be

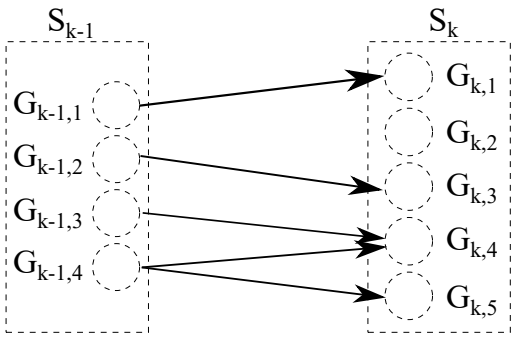


Figure 5.5: Group mapping between time steps can be considered as a bipartite graph. Circles indicate groups detected in each time step. Directed edges indicate that $GES(G_{k-1,i}, G_{k,j}) \geq C$ for endpoint groups.

depicted as a bipartite graph (Figure 5.5), where groups that exist at step $k-1$ are in the partition S_{k-1} and the groups that exist at step k are in the partition S_k. The relationship between groups that are present at different time steps can be identified based on the *GES* metric (Algorithm 3). Given a group i at step $k-1$, $G_{k-1,i}$, a group j Given a group i at step $k-1$, $G_{k-1,i}$, a group j at step k, $G_{k,j}$, and a **GES threshold** C (Definition 5.7), we build a mapping between the two groups if $GES(G_{k-1,i}, G_{k,j}) \geq C$. A higher *GES* threshold implies that a larger portion of the later group comes from the former group. We set the default value of *GES* threshold C to 0.2 in our experiments, based on parameter sensitivity tests. Note that the *GES* measurement is not symmetric as it is focused more on the later state of the group. Our earlier tests with symmetric functions such as Jaccard index show that the symmetric functions lead to small and inconsistent values when mapping groups over the time steps. Therefore we do not use symmetric functions to measure the group similarly in this work. The mapping process compares each group from step $k-1$ with each group in step k. This takes $\mathcal{O}(g_{k-1}g_k)$ time to run as it compares each group in time step $k-1$ with each group in time step k, where g_{k-1} is the number of groups present in time step $k-1$ and g_k is the number of groups present in time step k. Although our current implementation only considers two consecutive steps in group mapping, it is straightforward to extend the process such that the mapping can be done between step $k-n$ and step k, where n is an arbitrary number. In order to compute *GES* in the mapping process, we need to keep the group entities for different time steps. The following section details the history module, which can be used for this purpose.

Algorithm 3: Map(S_k, S_{k-1}, C)

Input: S_k - groups in step k
S_{k-1} - groups in step $k-1$
C - threshold value for GES
Output: Mappings between S_{k-1} and S_k

```
1 foreach (Group G_{k-1,i} ∈ S_{k-1}) do
2     foreach (Group G_{k,j} ∈ S_k) do
3         ges ← GES(G_{k-1,i}, G_{k,j})
4         if (ges ≥ C) then
5             Mappings.add(G_{k-1,i}, G_{k,j},ges)
6         end
7     end
8 end
9 return Mappings;
```

5.3.2 History Module

The history module stores the group entity information for one or more time steps. The information is used in calculating GES to map groups between consecutive time steps. The information is also used in calculating the weighted centroid of groups (Section 5.3.3.2 and Section 5.3.3.4). In our current implementation, the history module maintains group membership information for the previous time step, but this can be easily extended to store the membership information for an arbitrary number of steps if the group mapping process (Section 5.3.1.3) needs to consider more than two consecutive steps. Also, for any group member, the history module keeps a count showing the number of consecutive steps that the user posted locations.

5.3.3 Kalman Filter Module

After the groups are detected, GKF uses the Kalman filter to estimate the location of the groups. We follow the formulation proposed in [123] when developing the Kalman filter module for GKF.

To describe the mechanism of the module, let us start with a simple case, where we

track a single LBSN user u_i who posts locations from time to time. The Kalman filter module maintains the state $x_{k,i}$, a state vector which represents the dynamic behaviour of the object (in our case the LBSN user u_i) at time step k. The objective is to estimate the state $x_{k,i}$ based on the measurement of u_i's posted locations, $z_{k,i}$, which may contain a certain level of noises. Since we are interested in the movement of the user, the state vector $x_{k,i}$ should include the user's location and velocity. Therefore, $x_{k,i}$ can be expressed as:

$$x_{k,i} = \begin{bmatrix} lat_{k,i} & lon_{k,i} & V_{lat,k,i} & V_{lon,k,i} \end{bmatrix}^T$$

Here, $lat_{k,i}$ and $lon_{k,i}$ are the latitude and longitude of user u_i, respectively. $V_{lat,k,i}$ and $V_{lon,k,i}$ are the velocity of the user in the latitudinal direction and the longitudinal direction, respectively.

The Kalman filter makes estimations based on a system of equations. The first equation is referred to as the **process model** (Equation 5.2), which is used to compute a prediction of u_i's state at time step k, $\hat{x}_{k,i}$, based on the user's estimated state at the previous time step, $x_{k-1,i}$. In this equation, A is a matrix called the state transition model. B is a matrix called the control input model. c_k is the control signal that is given to the system to change its behaviour. As we do not control the movement of the user, the Bc_k vector component can be ignored for our analysis.

$$\hat{x}_{k,i} = Ax_{k-1,i} + Bc_k + w_k \tag{5.2}$$

$Ax_{k-1,i}$ takes the following form, assuming that a time step covers a period of Δt.

$$Ax_{k-1,i} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} lat_{k-1,i} \\ lon_{k-1,i} \\ V_{lat,k-1,i} \\ V_{lon,k-1,i} \end{bmatrix}$$

The multiplication between A and $x_{k-1,i}$ produces a column matrix with 4 rows as

follows. The first row of the resulting matrix is $lat_{k-1,i} + V_{lat,k-1,i} \times \Delta t$. This is the new latitude value of the user ($lat_{k,i}$ of $\hat{x}_{k,i}$). Similarly, the second row of the resulting matrix gives the new longitude value of the user. As we assume that the speed of the user does not change in one time step, the latitudinal velocity (the third row) and the longitudinal velocity (the fourth row) are the same as in $x_{k-1,i}$.

Vector w_k represents Gaussian process noise with covariance Q. In our case, w_k captures all the noises associated with GPS locations that are posted by the users. Since we do not have means to capture these noises, w_k is ignored for $\hat{x}_{k,i}$ estimation.

However, the noise covariance matrix Q is used when calculating error covariance matrix $P_{k,i}$. This error covariance matrix determines the impact of the components in $x_{k-1,i}$ vector on the components in $x_{k,i}$. It can be computed using Equation 5.3. For a newly detected group, we set $P_{k-1,i} = I_4$. For the sake of simplicity, we use $Q = 4.9 \times I_4$ in our implementation, which means that each component of $x_{k,i}$ vector depends only on the previous state of their own and value of this dependence is equal to typical GPS accuracy in metres.

$$P_{k,i} = AP_{k-1,i}A^T + Q \tag{5.3}$$

When the measurement of the user's location at time step k becomes available, the Kalman filter module updates the user's state, $\hat{x}_{k,i}$, and the error covariance matrix, $P_{k,i}$. In order to do this, we need to model the relationship between the measurement vector ($z_{k,i}$) and the state vector ($x_{k,i}$). The measurement vector of our model takes the following form, where $lat_{k,i}$ and $lon_{k,i}$ are the latitude and longitude of the location posted by user u_i at time step k.

$$z_{k,i} = \begin{bmatrix} lat_{k,i} & lon_{k,i} \end{bmatrix}^T$$

The measurement is affected by a number of factors as shown in the **measurement**

model (Equation (5.4)).

$$z_{k,i} = Hx_{k,i} + v_k \tag{5.4}$$

In the measurement model, H is an observation model that maps the process state space into a measured space. In our case the component $Hx_{k,i}$ takes the following form.

$$Hx_{k,i} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} lat_{k,i} \\ lon_{k,i} \\ V_{lat,k,i} \\ V_{lon,k,i} \end{bmatrix}$$

Vector v_k represents Gaussian process noise with a covariance R. For the sake of simplicity, we set $R = 4.9 \times I_3$, with the value of the typical GPS accuracy in metres.

The following three equations are used in the **update step** of the Kalman filter.

$$K_{k,i} = P_{k,i}H^T(HP_{k,i}H^T + R)^{-1} \tag{5.5}$$

$$\hat{x}'_{k,i} = \hat{x}_{k,i} + K_{k,i}(z_{k,i} + H\hat{x}_{k,i}) \tag{5.6}$$

$$P'_{k,i} = (1 - K_{k,i}H)P_{k,i} \tag{5.7}$$

The update step starts with calculating $K_{k,i}$ (Equation 5.5), called Kalman gain, which acts as a weight for calculating the updated state vector ($\hat{x}'_{k,i}$) and the updated error covariance matrix ($P'_{k,i}$). The state vector and the error covariance matrix are updated using Equation 5.6 and Equation 5.7.

The updated state vector and the updated error covariance matrix can be used to

predict the state and error covariance matrix for the next time step. That is, we can use $\hat{x}'_{k,i}$ in place of $\hat{x}_{k-1,i}$ in Equation 5.2 to predict $\hat{x}_{k+1,i}$. Similarly, we can use $P'_{k,i}$ in place of $P_{k-1,i}$ in Equation 5.3 to predict $P_{k+1,i}$. We call this the **prediction step**. This process can be repeated for any number of time steps. For example, when the next measurement $z_{k+1,i}$ for time step $k+1$ becomes available, the Kalman filter module can update $K_{k+1,i}$, $\hat{x}'_{k+1,i}$ and $P'_{k+1,i}$.

As mentioned earlier, a group location is represented as a circle with a centroid and a radius. Using the Kalman filter module, we estimate the position of the centroid and the size of the radius for any detected group. When estimating the location of the groups, we still use the prediction step and the update step as shown with the single-user scenario. However, we need to extend the process model and the measurement model so that they work for the group-based scenarios. The extended models are detailed in Section 5.3.3.1 to Section 5.3.3.4.

5.3.3.1 Basic Centroid Model

The state vector ($x_{k,i}$) for a group $G_{k,i}$ is similar to the vector in the single-user scenarios with two differences. First, the latitude, longitude and velocity are based on the centroid of the group. Second, the vector has an additional parameter of $r_{k,i}$, which represents the radius of the circle that centred at the centroid. Hence $x_{k,i}$ takes the following form.

$$x_{k,i} = \begin{bmatrix} lat_{k,i} & lon_{k,i} & V_{lat,k,i} & V_{lon,k,i} & r_{k,i} \end{bmatrix}^T$$

The process model needs to accommodate the extended state vector. Different to the single-user case, the state transition model (A) for the group-based case should be a 5×5 matrix with an additional row to accommodate $r_{k,i}$. The component, $Ax_{k-1,i}$, in Equation 5.2 takes the following form.

$$Ax_{k-1,i} = \begin{bmatrix} 1 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} lat_{k-1,i} \\ lon_{k-1,i} \\ V_{lat,k-1,i} \\ V_{lon,k-1,i} \\ r_{k-1,i} \end{bmatrix}$$

The noise covariance matrix Q and the error covariance matrix $P_{k,i}$ also need to be extended. In our implementation, we set $Q = 4.9 \times I_5$ which is the typical value of GPS accuracy and $P_{k,i} = I_5$. The measurement vector, $z_{k,i}$, takes the following form.

$$z_{k,i} = \begin{bmatrix} lat_{k,i} & lon_{k,i} & r_{k,i} \end{bmatrix}^T$$

The $lat_{k,i}$ and $lon_{k,i}$ are the latitude and longitude of the centroid of $G_{k,i}$ at time step k, respectively. The distance between the centroid and the furthest group member is measured as $r_{k,i}$. The measurement model is extended from the single-user case to accommodate the additional $r_{k,i}$. The component $Hx_{k,i}$ in the model takes the following form while the covariance of the Gaussian process noise for vector v_k is set to $R = 4.9 \times I_3$, where 4.9 metres is the typical accuracy of GPS enabled devices.

$$Hx_{k,i} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} lat_{k,i} \\ lon_{k,i} \\ V_{lat,k,i} \\ V_{lon,k,i} \\ r_{k,i} \end{bmatrix}$$

Now we have all the components for tracking a detected group using the Kalman filter. We should note that the position of the centroid is computed by simply averaging the positions of all the group members in the basic centroid model. We also present other ways to compute the centroid in the following sections.

5.3.3.2 Weighted Centroid Model

To improve the accuracy of location estimation, we develop a weighted centroid model, which gives more weights to the locations of users who post more frequently than other users. The rationale behind the model is that the users, who post more frequently than others, tend to be the leaders of their groups. Hence, their locations are more indicative of group locations. The model computes the centroid using the following formula.

$$Centroid(z_{k,i}) = \frac{\sum_{j=1}^{n}(m_j + 1/(1+d_j))l_j}{n + \Sigma m_j + \Sigma(1/(1+d_j))}$$

In this formula, m_j refers to the number of consecutive time steps where a user u_j posted to the LBSN, d_j refers to the distance from the basic centroid (the average of the posted locations) to the location posted by u_j, n is the number of group members who posted at time step k and l_j refers to the location in a post. In our implementation, the maximum value of m_j is the maximum number of consecutive steps a user has posted continuously, and this value resets to zero if the user does not post for a single time step. The radius measurement $r_{k,i}$ of $G_{k,i}$ should be made after the weighted centroid is computed. The weighted centroid and the updated radius are used as input to the measurement vector $z_{k,i}$. It is possible to incorporate other user behaviours [106] into the model.

5.3.3.3 Basic Cumulative Model

In the basic centroid model and the weighted centroid model, we calculate the group centroid outside the Kalman filter and feed it into the Kalman filter as $z_{k,i}$. In this section, we present a model that allows us to feed the locations of group members into the Kalman filter directly. To do this we need to first modify our measurement vector $z_{k,i}$ matrix such that the matrix can take the locations from an arbitrary number of group members. Assuming n is the number of locations posted from group i in time step k, the modified

$z_{k,i}$ takes the following form, where i_1 to i_n are the user IDs in group $G_{k,i}$.

$$z_{k,i} = \begin{bmatrix} < lat_{k,i_1} & lon_{k,i_1} > ... < lat_{k,i_n} & lon_{k,i_n} > & r_{k,i} \end{bmatrix}^T$$

In this model, $z_{k,i}$ is a column matrix with $2n+1$ rows. In order to map this extended $z_{k,i}$ to $x_{k,i}$, which is a 5×1 matrix, we need to extend the observation model H to a variable $H_{k,i}$, where k refers to the time step and i refers to the relevant group, as follows.

$$H_{k,i} = \begin{bmatrix} F_1 \\ : \\ F_j \\ : \\ F_n \\ D \end{bmatrix}$$

In this equation, F_j where $j = \{1, 2, ..., n\}$ and D are defined as follows,

$$F_j = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}, D = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

We should note that the size of $H_{k,i}$ can change over time. When this variable $H_{k,i}$ matrix is in use, the R matrix should also be a variable to accommodate the changes in $H_{k,i}$. Since the size of $H_{k,i}$ becomes $(2n+1) \times 5$, we set $R = 4.9 \times I_{(2n+1)}$.

5.3.3.4 Weighted Cumulative Model

The basic cumulative model can be further extended with the weights that are used in the weighted centroid model. More specifically, we can incorporate the m_j component of the weighted centroid calculation, which gives higher importance to the locations from users

Algorithm 4: Group Movement Prediction (S_k, S_{k-1}, C)

Input: groups in step k (S_k), groups in step $k-1$ (S_{k-1}), threshold value for GES (C)
Output: Predicted locations of groups at step $k+1$

```
1  Mappings ← Map(S_k, S_{k-1}, C) ;                    // Section 5.3.1.3
2  foreach (Mapping e ∈ Mappings) do
3      G_{k-1,e} ← e.getGroups(k − 1)
4      G_{k,e} ← e.getGroups(k)
5      if (G_{k-1,e} and G_{k,e} have a 1-to-1 mapping) then
6          Apply Kalman update step
7          Predict next locations using predict step
8      end
9      else
10         Stop tracking G_{k-1,e}
11         Initialize tracking G_{k,e}
12     end
13     Update group members for G_{k,e}
14 end
15 return predictions
```

who are continuously posting with the group. To achieve this, we duplicate the location of the users, who make posts in consecutive time steps, in the measurement vector $z_{k,i}$ used by the basic cumulative model. The number of duplications for a user is equal to the number of consecutive time steps that the user posts to the LBSN.

It should be noted that the size of the matrices used by the cumulative models can be considerably larger than the size of the matrices used by the centroid based models. However, we think that the additional computation cost with the cumulative models can be worthwhile due to the increased accuracy in tracking the groups.

5.3.3.5 Group Movement Prediction Algorithm

The whole process of predicting the movement of groups is shown in Algorithm 4. We assume that the groups at time step $k-1$ and k are all detected based on spatial clustering. At first, the groups at step k are mapped to the groups from the previous time step $k-1$ in order to find the groups that are related to each other at the two steps (Line 1). The

mapping is based on the explanation provided in section 4.1.3. If a group at step k has the same entity as a group at step $k-1$ (Figure 5.4(a)), the group's location at step k is updated and the group's location at step $k+1$ is predicted using the Kalman filter (Line 6-7). For the situations shown in Figure 3(b)-(f), we stop tracking the groups that present at step $k-1$ and start tracking the groups that present at step k (Line 10-11).

5.4 Experimental Settings

We implemented the proposed solutions in Java 8. For the Kalman filter implementation, we used the existing implementation from Apache Commons 3.6.1.

5.4.1 Baseline Solutions

We compare GKF against two baseline solutions for tracking the movement of groups. The first baseline solution includes Dead Reckoning for centroid prediction and Radius Dead Reckoning for radius prediction. The second baseline solution includes Graph Movement for centroid prediction and Constant Radius Rates for radius prediction. The algorithms used by the baseline solutions are detailed as follows.

5.4.1.1 Baseline Algorithms for Centroid Prediction

5.4.1.1.1 Dead Reckoning When predicting the centroid of the area of a group, dead reckoning assumes that a group does not change its direction and speed between two consecutive time steps [127]. Hence, the direction of group movement is determined by linear extrapolation based on the centroid at the current step and the centroid at the previous step. At each time step, the dead reckoning solution uses the observed locations of the group members to compute the centroid of the group. Then, it computes the velocity of the group based on the centroid at the current step and the centroid at the previous

step. Based on the velocity and the centroid at the current step, dead reckoning computes the predicted centroid for the next time step.

5.4.1.1.2 Graph Movement The second algorithm is inspired by the work presented in [29]. This algorithm first generates a network graph that contains all the potential centroids of all the groups. We assume that any centroid found by this algorithm is a location that has been posted by one or more users to LBSNs. The generation of the graph is based on a data structure called the Gabriel graph [7]. To predict the next centroid of a group, the algorithm finds all the potential paths that start from the current centroid of the group. All the paths have the same distance, which is the distance travelled by the group in the previous time step. We use Dijkstra's algorithm to calculate the potential paths. Finally, based on the group's direction of movement, which is found based on the centroid at the current time step and the centroid at the previous time step, we select the path that closely follows the direction of movement as the predicted path. The endpoint of the path is the predicted centroid of the group.

5.4.1.2 Baseline Algorithms for Radius Prediction

5.4.1.2.1 Radius Dead Reckoning This algorithm assumes that a group does not change its radius from the previous time step. Therefore, the radius measured in the current time step is used as the predicted radius for the next time step.

5.4.1.2.2 Constant Radius Rate Different from Radius Dead Reckoning, which assumes that the radius of a group does not change from the previous time step, Constant Radius Rate assumes that the radius can change at a constant rate. For example, if a group is shrinking between the previous time step and the current time step, we assume that the group will shrink further in the next time step. We calculate the radius change rate based on the radius at the previous time step and the radius at the current time step.

5.4.2 Evaluation Metrics

We evaluate the accuracy of prediction based on two metrics. One is centroid error distance, which is the distance between the predicted centroid and the measured centroid. For a given time step k, the value is the distance between the prediction made at step $k-1$ and the measurement made at step k. The other metric is the radius error distance, which is the difference between the predicted radius and the measured radius. We should note that the measured centroid and the measured radius for the baseline solutions are computed based on the basic centroid model.

We also measure the computation time for making predictions using the solutions. The running times are measured for individual time steps separately. We compute the average running time per step for each of the solutions. We should note that the time for radius prediction is negligible compared to the centroid prediction. Therefore, we only report the time for centroid prediction.

5.4.3 Group Movement Scenarios

Due to the lack of group movement data that shows complex movement patterns, we created synthetic scenarios to simulate the movement of groups. Our synthetic data sets contain the movement of several groups in Melbourne CBD area during a 30-minute window. All the simulated users post their locations at a specific frequency. A record in the data sets contains a location, an artificial user ID and an artificial time stamp. The data sets exhibit complex group movements at specific intervals. In addition to the synthetic data sets, we created two data sets based on real LBSN data. Both data sets exhibit the movement of a small group of people during a 12-15 minute period. The real data sets are detailed in Section 5.5.2.

5.4.4 Parameter Settings

We evaluate the effects of two sets of parameters. The first set, group movement parameters, affects the movement patterns shown in the synthetic LBSN data. The set includes the number of groups (g), the number of users (n) and the interval between group-entity-changes such as merging and splitting (I). The parameter settings of this set are shown in Table 5.1. The second set, GKF parameters, only affect the performance of GKF. The second set includes time step size (Δt), distance threshold in clustering (d) and GES threshold (C). Their settings are shown in Table 5.2. All six parameters are included in the definition of our research problem (Section 5.2). The GKF parameters, Δt, d and C, are used in the spatial-temporal clustering module of GKF (Section 5.3.1).

We conduct comparative tests with all the solutions based on the first set of parameters. For each parameter, we create multiple sets of synthetic LBSN data with different settings of the parameter. The settings of other movement parameters are kept to their default values. For each set of synthetic data, we use the variations of GKF and the baseline solutions to predict the location of groups at each time step. We should note that the parameter settings for GKF are based on the default values as shown in Table 5.2. We should also note that the group movement settings are only applied to the synthetic data. We use fixed parameter settings in the experiments with real LBSN data.

Table 5.1: Group movement settings

	No. of Groups (g)	No. of Users (n)	Entity Change Interval (I)
1	3-20	100	3min
2	10	30-200	3min
3	10	100	1min-5min

We evaluate the effects of the second set of parameters using the four variations of GKF. Similar to the experiments on group movement parameters, we run three sets of tests. In each set of the tests, we vary the value of a specific parameter while keeping other parameters at their default values.

Table 5.2: GKF parameter settings

	Time Step Size (Δt)	Distance Threshold In Clustering (d)	GES Threshold (C)
1	1min-5min	5m	0.2
2	3min	1m-20m	0.2
3	3min	5m	0.1-0.3

We conduct comparative tests with all the solutions based on the group movement parameters. For each parameter, we create multiple sets of synthetic LBSN data with different settings of the parameter. The settings of other movement parameters are kept to their default values. For each set of synthetic data, we use the variations of GKF and the baseline solutions to predict the location of the groups at each time step.

We evaluate the effects of GKF parameters using the four variations of GKF. Similar to the experiments on group movement parameters, we run three sets of tests. In each set, we vary the value of a specific GKF parameter while keeping other GKF parameters at their default values.

In addition to the evaluation with synthetic LBSN data, we evaluate the solutions with real LBSN data. Due to the small size of real LBSN data exhibiting group movements, we do not vary parameter values with the real data. The settings used with the real data are detailed in Section 5.5.2.

5.5 Experimental Results and Discussion

Before presenting the results on individual parameters, we show the results from a simplified scenario, where two user groups merge into one group, which splits into two groups again later. The scenario is illustrated in Figure 5.6, where a group location is denoted as $G_{(TimeStep),(GroupIndex)}$. We observe that all the four variations of GKF, which use the same group detection procedure, can detect the two groups at time step 0 and initialize Kalman filter to track the groups from the step. The GKF solutions also correctly

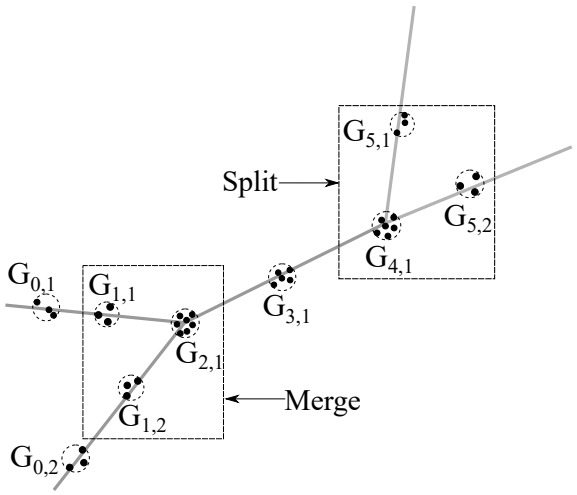


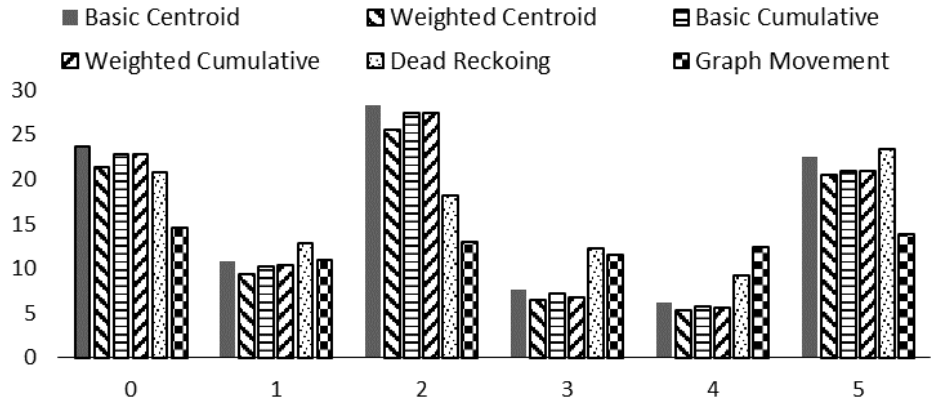
Figure 5.6: The simplified synthetic scenario with a group-merging (time step $1 \rightarrow 2$) and a group-splitting (time step $4 \rightarrow 5$). Notation $G_{k,i}$, indicates a group with index i in time step k.

detect the merging and splitting events in the following time steps.

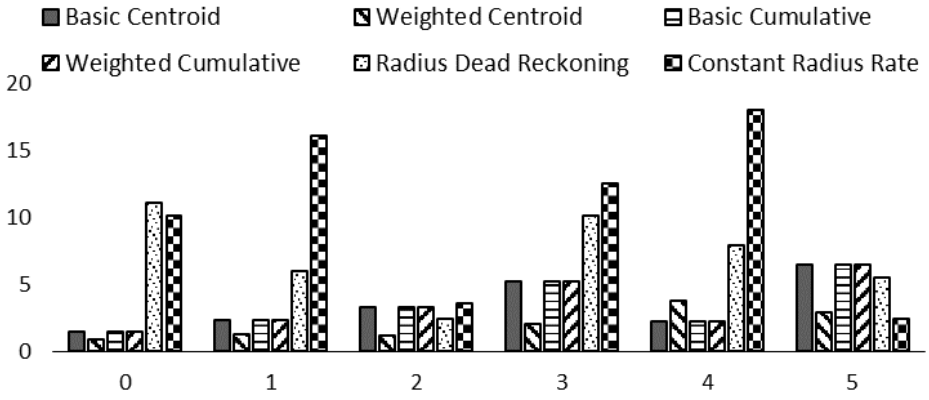
GKFs show relatively large errors in predicting centroids when it needs to start tracking new groups at time step 0, 2 and 5 (Figure 5.7(a)). This is due to the fact that the Kalman filter is not stabilized at the exact moment that a group event happens. Once the Kalman filter is stabilized, as shown in other time steps, GKFs achieve lower errors compared to the baseline methods. For example, the variations of GKF can improve the prediction of the centroid location by an average of 12% from dead reckoning and by 18% from graph based movement (Figure 5.7(a)). The variations of GKF improve the radius prediction by more than 41% over Radius Dead Reckoning and by more than 57% over Constant Radius Rate (Figure 5.7(b)).

5.5.1 Results on Synthetic Scenarios

In the synthetic scenarios of group movement in Melbourne CBD, we discovered that when there is a continuously posting user in the group, the two group movement models that use the history module (the weighted centroid model and the weighted cumulative model) exhibit an accuracy gain of 36% over the models that do not use the history



(a) Centroid error distance (in metres) at different time steps



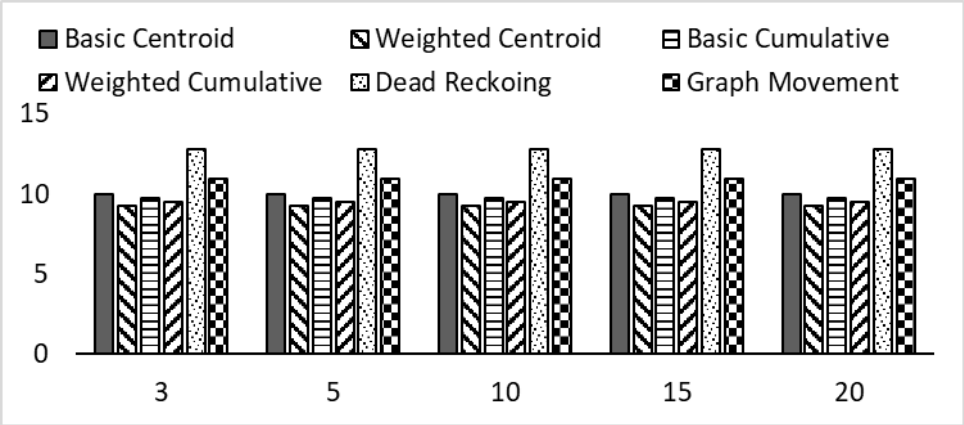
(b) Radius error distance (in metres) at different time steps

Figure 5.7: Prediction errors in the simplified synthetic scenario at 5 time steps.

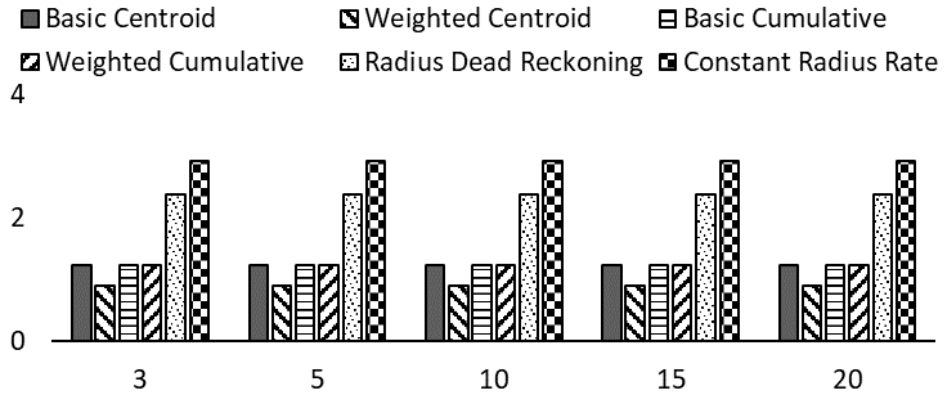
module (the basic centroid model and the basic cumulative model). These results show the positive impact of introducing the history module in GKF. The effects of individual parameters are detailed in Section 5.5.1.1 and Section 5.5.1.2.

5.5.1.1 Effects of Group Movement Parameters

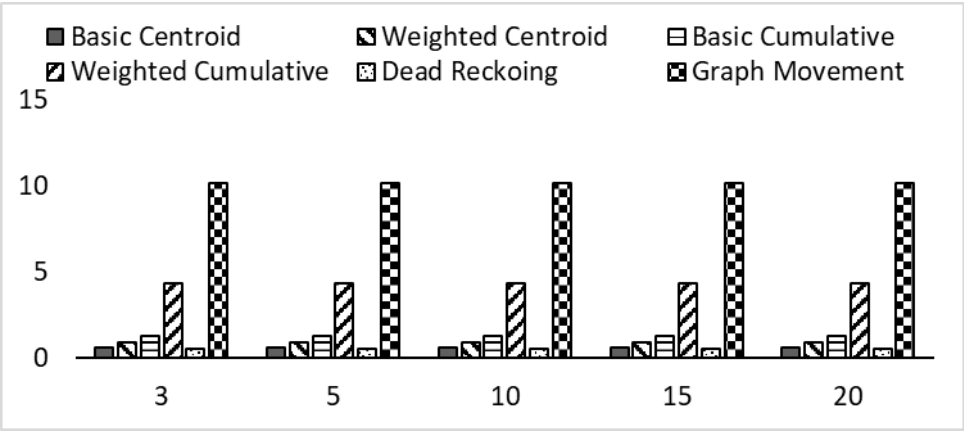
The effects of the number of groups are shown in Figure 5.8. The variations of GKF reduce errors in predicting the centroid of groups by an average of 15% from Dead Reckoning and 10% from Graph movement (Figure 5.8 (a)). All the GKF variations achieve an even larger reduction of errors in predicting the radius of group coverage. They reduce the



(a) Average centroid error distance in metres



(b) Average radius error distance in metres



(c) Average computation time for centroid prediction in milliseconds

Figure 5.8: Results on the number of groups.

prediction errors by an average of 50% from Radius Dead Reckoning and 60% from Constant Radius Rate (Figure 5.8 (b)). Among the variations of GKF, the Weighted Centroid

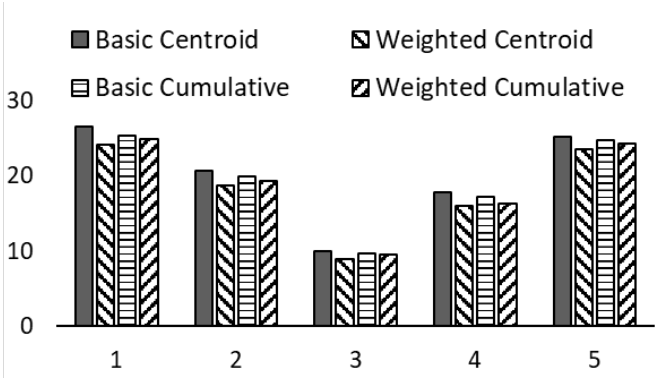
variation performs better than other variations as it considers the effects of group leaders on the group locations.

For centroid prediction, GKFs take similar computation times as Dead Reckoning except for the Weighted Cumulative variation, which needs to perform a large number of matrix multiplications (Figure 5.8 (c)). All the variations of GKF run significantly faster than the baseline solution that uses graph-based movement prediction.

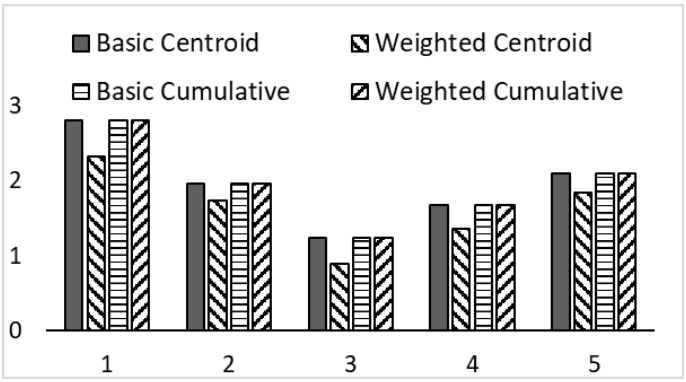
When varying the number of users or the entity change interval, the performance of all the solutions is similar to that achieved by varying the number of groups. Furthermore, our results on all the group movement parameters show that the error levels achieved by all the solutions remain steady under different parameter settings. For example, the centroid error distance achieved by Basic Centroid solution is always close to 10 metres when the number of groups changes from 3 to 20 (Figure 5.8 (a)). This can be due to the fact that all the users move along the streets in a CBD area in this synthetic scenario. The spread of a group is limited by the streets as the group tends to be distributed along a narrow strip in the area. Consequently, the effects of the different settings are limited.

5.5.1.2 Effects of GKF Parameters

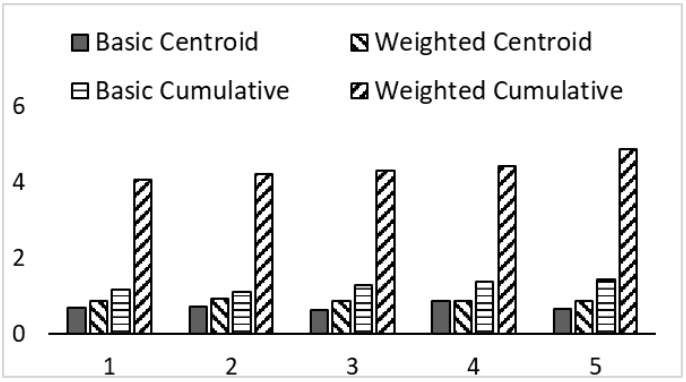
5.5.1.2.1 Time Step Size If the step size Δt is very small, the chance that all the group members post at the same time step is low. This can make it difficult to detect the members of the same group. On the other hand, if the step size is very large, we may have multiple posts from the same user in one time step. In other words, we cannot know the precise location of a group at a time step. Results on Δt are presented in Figure 5.9. We observe that the best setting of Δt is 3 minutes based on the results. When Δt increases from 1 minute, the errors in centroid prediction drop. The errors achieved by all the GKF variations are at the lowest level when Δt is at 3 minutes (Figure 5.9 (a)). The errors arise as Δt increases from 3 minutes. We observe a similar trend in the prediction errors of the radius (Figure 5.9 (b)). Since the default entity change interval is also 3 minutes, the



(a) Average centroid error distance in metres



(b) Average radius error distance in metres

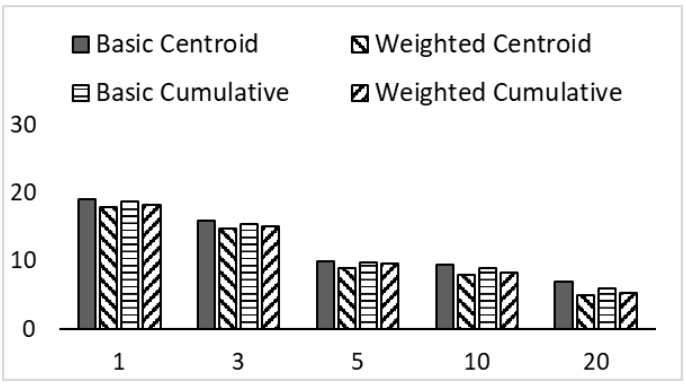


(c) Average computation time for centroid prediction in milliseconds.

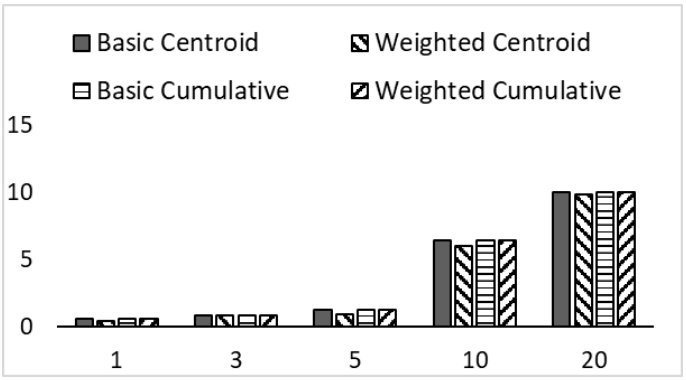
Figure 5.9: Results on time step size.

results show that GKF works best when time step size matches the entity change interval. We also observe that the Weighted Centroid variation outperforms other variations in predicting the centroid location and the radius.

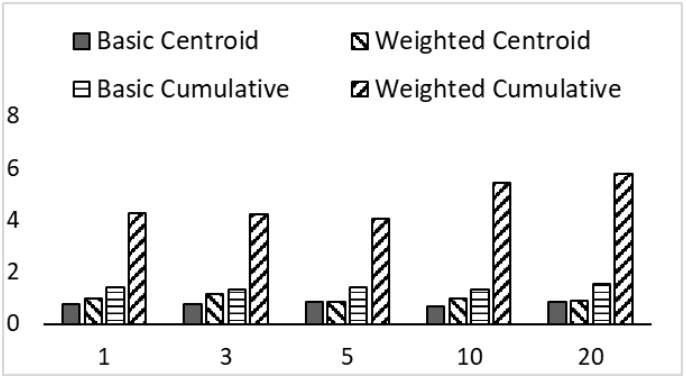
In terms of computation costs, the computation time of the Weighted Cumulative variation is at least 3 times the computation time of other variations (Figure 5.9 (c)). This is because the Weighted Cumulative model needs to perform calculations based on large matrices.



(a) Average centroid error distance in metres



(b) Average radius error distance in metres



(c) Average computation time for centroid prediction in milliseconds

Figure 5.10: Results on distance threshold in clustering.

5.5.1.2.2 Distance Threshold in Clustering If the distance threshold in clustering, d, is set to a low value, individual users can be detected as single-user groups. On the other hand, a large distance threshold can include all the users in a large area into one group. The effects of the parameter are presented in Figure 5.10. We can see that the centroid error drops when d increases from 1 metre to 20 metres (Figure 5.10 (a)). However, the radius error increases at the same time, especially when d is larger than 5 metres (Figure 5.10 (b)). With a higher value of d, a detected group tends to include more users, who are actually moving in multiple sub-groups. The centroid of the detected group tends to be static over time because the sub-groups can move in all directions. This is the reason that the error of centroid prediction drops with a higher value of d. However, due to the highly dynamic movement of the sub-groups, which are at the border of the space covered by the detected group, the radius prediction can contain significant errors. This results in an increase of radius prediction error when d increases.

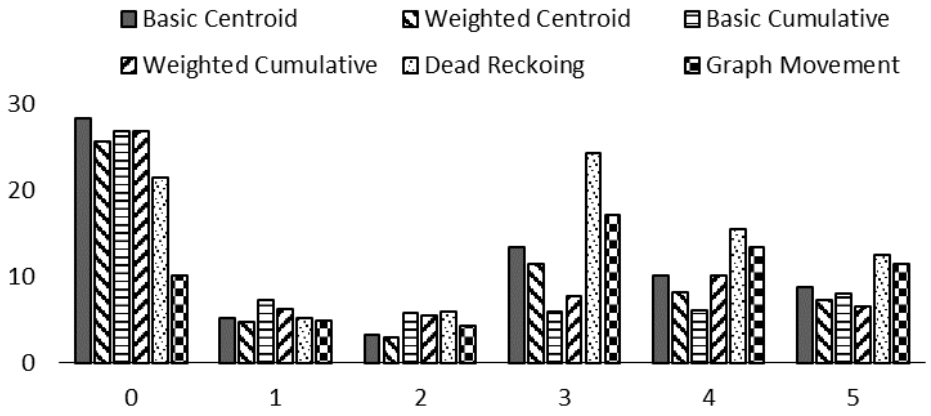
5.5.1.2.3 GES Threshold Our results on the GES Threshold, C, do not show a clear pattern with different settings of the parameter. When the threshold is low ($C \leq 0.1$), GKF may consider the location of users who are not actually exhibiting any group movement behaviour. On the contrary, when C is set to a relatively large value, GKF may lose the track of some groups with small entity changes. Therefore, the value of C should be determined case by case. We set the default value of C to 0.2 based on our results.

5.5.2 Results on Real Scenarios

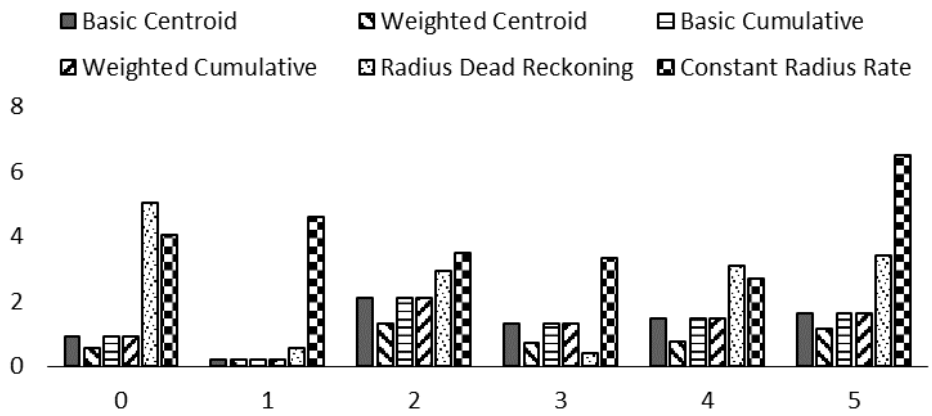
In the group movement tracking experiment done on the real data sets, we experienced that the tracking solutions based on GKF perform better than baseline solutions. For both real scenarios, GKF based solutions performed with over 18% more accuracy than baseline solutions while operating in the same execution time. This shows the superior effectiveness of GKF based solutions.

5.5.2.1 Disneyland Scenario

Based on the Yahoo! Flickr Creative Commons 100M (YFCC100M) dataset [117], we found a group of 5 users who travelled together through the entrance of Disneyland in California on January 3, 2012. The group travelled for 6 consecutive time steps, each of which covers a period of 2 minutes. The first half of the group trajectory exhibits a unidirectional movement. The second half of the trajectory exhibits the movement around a corner. In this experiment, the GKF parameters are set to the default values used in the experiments on synthetic scenarios (Table 5.2) except Δt, which is set to 2 minutes based on our preliminary tests.



(a) Centroid error distance (in metres) at different time steps



(b) Radius error distance (in metres) at different time steps

Figure 5.11: Prediction errors in the Disneyland scenario.

We collect the level of errors in centroid prediction and radius prediction at individual

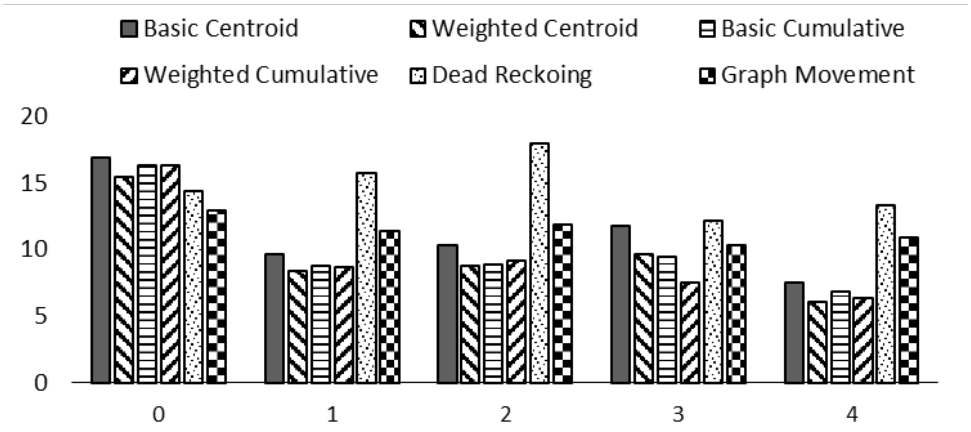
time steps. In the first time step, the four variations of GKF show larger errors in centroid prediction compared to the baseline solutions (Figure 5.11(a)). This is understandable as the Kalman gain is not stabilized at the first step. From the second step, GKFs achieve similar or smaller errors than the two baseline solutions. Overall, the four variations of GKF increase the accuracy of prediction by more than 24% from Dead Reckoning and by more than 18% from Graph Movement. We can also observe that the error distances in the second half of the period are generally higher than that in the first half of the period, except for time step 0. This is due to the fact that the group changes its movement pattern in the second half of the period.

The difference between the measured group radius and the predicted radius is shown in Figure 5.11(b). At most of the time steps, the variations of GKF achieve a better prediction of the radius than the two baseline solutions. Overall, GKFs improve the accuracy of radius prediction by more than 38% from Radius Dead Reckoning and by more than 65% from Constant Radius Rate.

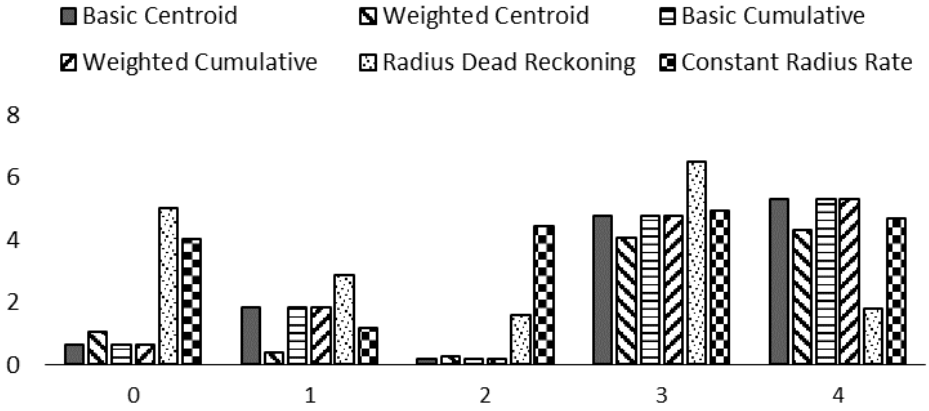
5.5.2.2 MCG Scenario

After analysing a Twitter data set, which contains the posts from a rectangular region that covers Australia and New Zealand between March 6, 2012, and April 23, 2012, we found a set of user trajectories that indicate group movement near the Melbourne Cricket Ground (MCG) stadium on March 29, 2012. The group consists of 4 users. The group travelled for 5 consecutive time steps, each of which covers a period of 3 minutes. The group was moving in one direction in a curved path around the stadium during the period. The parameters of GKF are set to their default settings as shown in Table 5.2.

Similar to the results on the Disneyland scenario, GKFs achieve relatively large errors in centroid prediction at the first time step as the Kalman gain is not stabilized at this step (Figure 5.12(a)). However, by considering the results from all the time steps, the variations of GKFs improve the accuracy of centroid prediction by more than 36% from Dead Reckoning and by more than 22% from Graph Movement.



(a) Centroid error distance (in metres) at different time steps



(b) Radius error distance (in metres) at different time steps

Figure 5.12: Prediction errors in the MCG scenario.

For radius prediction, the two baseline methods show larger errors than GKFs except in the last step (Figure 5.12(b)). Our data shows that only one group member posted in the last time step. This can affect the accuracy of GKF significantly. Overall, GKFs improve the accuracy of radius prediction by more than 26% from Radius Dead Reckoning and by more than 35% from Constant Radius Rate.

5.5.3 Analysis

We make the following key findings regarding the effectiveness of the variations of GKF compared to the baseline methods. First, when the system starts to track a new group,

GKF-based methods do not perform well due to the unstable Kalman gain in the first time step of the tracking. Once the Kalman gain is stabilized, GKFs perform considerably better against the baselines. Second, the basic centroid model performs similarly to the basic cumulative model in most cases. However, the later model can achieve smaller errors in certain circumstances. Third, from the results on the synthetic datasets, we observe that the weighted models can reduce prediction error by up to 36% from the basic models due to the usage of the history module. The two weighted models show similar performances in general. Fourth, all variations of GKF run faster than the baseline solution that uses Graph Movement for centroid prediction. Finally, the use of the cumulative models in GKF can cause an increase in running time due to the usage of large matrices.

Continuous accurate prediction of the GKF depends on the availability of location data in the future. Even if data related to a group become not available for some time GKF can still predict the groups location using an stabilized GKF related to that group. However the predictions may become less accurate as the time passes. If that group begins to post again the existing GKF instance can be updated with a update step and can be used to predict accurately again.

Note that we have selected the Kalman filter as the solution of GKF. However, the architecture of GKF is extensible with other filter solutions. Since other components of the GKF are not tightly coupled with the Kalman filter, it can be replaced with other filtering solutions related to the Kalman filter such as the Wiener filter [111] and particle filter [112]. We have selected the Kalman filter due to its simplicity, good accuracy and ease of explanation. With other replacement filters the inputs, outputs and process modelling with vary but the operation of the GKF will remain the same.

5.6 Summary

We introduced the concept of the Group Kalman Filter to detect, track and predict the movement of groups using LBSN data. Several variations of GKF have been developed.

We have demonstrated that GKF achieves lower errors in tracking groups than the baseline solutions while keeping computation time in the same range as the baseline solutions. We have also shown that our solutions can handle situations involving complex group movement patterns.

Next chapter concludes the thesis by providing an analysis on the implications of each contribution discussed thus far. The contribution will be linked to appropriate use cases from existing literature, and future research directions related to the contributions will be discussed.

Chapter 6

Conclusions and Future Work

In this thesis, we investigate how LBSN data can be used to analyse public movement. To do this we propose data structure and algorithmic solutions. In this chapter, we summarize what has been discussed so far and present future research directions based on our findings.

6.1 Summary

LBSN data is a rich inexpensive data source for analysing public movement. However, using LSBN data to support real-world use cases is challenging due to sparsity and irregularity associated with their posting patterns. This uncertainty resulting from the properties of LBSN data can be mitigated by combining uncertain data to generate more certain information.

In chapter 3 we presented the Stepping Stone Graph, to create a movement network between the locations posted to LBSN. The Stepping Stone Graph is created by evaluating the emptiness of a variable area named the Diversion neighbourhood between two locations. This Diversion neighbourhood varies based only on one parameter. Therefore the Stepping Stone Graph also varies based only on one parameter. We showed mathematically how the Stepping Stone Graph related to other well-known graph structures such as the Gabriel Graph, the relative neighbourhood graph and Delaunay triangulation. We

presented an efficient algorithm to extract the planar Stepping Stone graph using the Delaunay triangulation, and also discussed how the Shortest Path Graph can be efficiently calculated using created Stepping Stone Graph. Next, we discussed some of the application of the Stepping Stone Graph and using two real data sets and a synthetic data set we showed the effectiveness of the Stepping Stone Graph for public movement analysis using LBSN data.

Next we introduced the Diversion Graph in chapter 4, as an efficient method to process increased volumes of LBSN data. The Diversion Graph is also defined using the Diversion neighbourhood, but instead of checking complete emptiness of the neighbourhood, we check only whether vertices from neighbouring Delaunay triangle sit within. Due to this relaxed evaluation criteria, the Diversion Graph creation time is 90% less compared to the Stepping Stone Graph creation time. However, maximally 2% of additional edges can present in the Diversion Graph, which is disadvantageous. Considering the performance gain, the presence of additional edges is an advantageous trade-off. Using real data and synthetic data we showed that the Diversion graph can be constructed in less than 10% of the time taken to construct the Stepping Stone Graph. Furthermore, the Diversion Graph can be used to calculate the Stepping Stone graph and the Shortest Path Graph efficiently.

In chapter 5 we presented the Group Kalman filter for group movement prediction using LBSN data. Group movement prediction is the use case we selected to address relating to the challenge of combining LBSN data from different users. Since this problem cannot be solved by proposing only data structures we proposed an algorithmic solution named the Group Kalman filter. The solution first divides available LBSN data into equal time steps, and clusters posted locations to identify spatial groups. Then based on a newly proposed metric, we link groups from different time steps to identify the persisting user group. In order to do this, we introduced a history module to store and aggregate this information. Next, we process identified user groups using the Kalman filter to predict their next location. The Kalman filter is a well-known technique used in signal processing, which predicts the next state of a process based on the current state's

measurements and previous state. To apply the Kalman filter, we need an accurate model of the process we are tracking. We propose four group movement prediction models with increasing complexity. We show the effectiveness of the Group Kalman Filter using real world data and synthetic data. Also, we experimentally analyse the Group Kalman Filter's operation as values of the configuration parameters vary.

6.2 Analysis

Considering the big picture of graph theory, both the Stepping Stone Graph and the Diversion Graph covers both planar connected and non planar connected regions. Let us illustrate these areas of operations compared to the areas described in Figure 11 of Empty Region Graphs paper [6]. As an empty region graph, the Stepping Stone graph operates in both non-planar and planar connected regions. The Diversion Graph only operates in the planar connected region, because it is an approximation of the Stepping Stone Graph. Since in this thesis we focused on movement analysis in Euclidean plane, we analysed only the planar connected regions of the graphs.

We investigated the group movement prediction as a use case of combining location data from different users and proposed the Group Kalman filter. Note that generic operational structure of Kalman filter is applicable when proposing solutions for other use cases such as trajectory clustering. This happens because the Group Kalman filter is proposed adhering to the basic principle of tracking multiple moving objects [107]. The two main problems we have to address when tracking multiple objects is known as the curse of dimensionality and mapping to distributed platforms. Curse of dimensionality is mitigated by developing a distributed state-space representation. In our solution, this is done by proposing a model to track the user population as separate groups rather than as a whole. Second challenge of mapping to distributed platforms is solved by developing a collaborative group abstraction. In the Group Kalman filter models, each group is allocated a single Kalman filter for processing. Following these principles, for any use case of combining location information from different users, we can develop a solution

similar to the architecture proposed in chapter 5.

6.3 Future Directions

There are several future directions that emerge from our research on public movement analysis using LBSN data.

6.3.1 Stepping Stone Graph

By analysing experimental results on the Stepping Stone Graph ($SSG(d)$), we can observe that $SSG(d)$ is both effective and efficient for representing and processing the possible movement network. Therefore, the proposed analysis method can be used in situations such as analysing movement patterns of animals where similar locality distributions are observed. In our experiment only the spatial and temporal aspects of the LBSN data is considered for analysis without geographic information of the area. Travel networks generated using $SSG(d)$ can be further improved by incorporating geographical information into the process.

The parameter selection for variable skeleton generation is considered as a manual process. It would be desirable to develop a method to autonomously assign a suitable value for parameter d when skeleton generation. This can be done by analysing the characteristics of point data available for analysis. In this regard research conducted on reconstructing curves given a point sets, using β-Skeletons can be useful [15].

Skeletons generated using $SSG(d)$ are useful for generating visualization. Overlapping point set visualization is a use case of the Shortest Path Graph that can be improved using $SSG(d)$ [18]. Further research can be conducted on information that can be visualized using other aspects such as the sentimental and semantic aspects of LBSN data. Along with these developments, researchers may look into the development of colouring overlays based on the created skeleton.

Analysis of trajectories resulting from moving objects based on $SSG(d)$ can be listed as a promising future direction. Even though, counter examples can be created against the movement networks inferred using neighbourhood graphs, analysing large sets of location data using neighbourhood graphs is an upcoming research field [128]. In these movement network analysis work, usually the Relative Neighbourhood Graph is used as a base to analysis locations. Since $SSG(d)$ and $DG(d)$ are closely related to RNG and provides variability based on the location distribution, they can be more suitable candidates as a base to analyse location data. With an ability to filter the LBSN posts generated while travelling in a vehicle, road network inferring using $SSG(d)$ can be compared against road inferring methods that uses methods other than graph based approaches. Furthermore, $SSG(d)$ can be used for movement planning by filtering out edges longer than a given threshold [31]. Also, $SSG(d)$ skeletons can be useful in detecting herding behaviour in emergency evacuation situations [106]. With the introduction of $SSG(d)$ as a super graph of $SPG(t)$ when $d \leq t$, it is worth investigating how use cases of $SPG(t)$ can be further improved. Since $SSG(d)$ emphasizes on tight locality clusters as d increases, we may able to break the connectivity property of the graph and identify clusters.

Since we are analysing geographic data we limited the $SSG(d)$ definition to the 2-dimensional case with Euclidean distance. As the Diversion Neighbourhood definition is distance based, the concept of $SSG(d)$ can be extended to higher dimensions. As planarity of the extracted skeleton is observed to be a desired property, we only presented results with a configurable value range of $d \geq 2$. But, there are use cases that can benefit from value range $d < 2$ [129]. Therefore, research on how the non-planar region of $SSG(d)$ behave is another interesting direction. We also studied the shape of $DN(d)$ for $d < 1$ but found those shapes to be less useful in location data analysis using neighbourhood graphs due to their asymmetric nature, but it may be useful to investigate these shapes for their other benefits.

6.3.2 Diversion Graph

Many of the future directions discussed relating to $SSG(d)$ apply to the Diversion Graph ($DG(d)$) as well. Autonomous detection of configuration parameter d, incorporating geographic information into the generation of the skeleton, generating visualization using $DG(d)$, movement planning and road network inference are such future directions that are applicable to both $SSG(d)$ and $DG(d)$.

Further research is necessary to determine the bounds of the performance gain achieved by relaxed criteria of $DG(d)$. Along with that, we need to determine bounds on the additional edges added to the skeleton violating empty Diversion Neighbourhood criteria of $SSG(d)$, in order to clearly understand the relationship between $SSG(d)$ and $DG(d)$. Regarding this future direction, determining bounds of spanning ratio of $DG(d)$ may be helpful [115].

Similar to $SSG(d)$, $DG(d)$ can also be extended to higher dimensions using different distance metrics. However, in any dimension, $DG(d)$ will generate only planar skeletons. Due to this property and fast generation of the skeleton, $DG(d)$ can be used to propose fast clustering methods by breaking the connectivity property.

6.3.3 Group Kalman Filter

Our current implementation of the Group Kalman Filter (GKF) is focused on addressing the impact of the spatial extent factor, one of the dynamic factors that are important to the group tracking with LBSN data. As future work, incorporating other dynamic factors, such as group entity and posting time, into the Group Kalman Filter to further improve the effectiveness of tracking can be considered. Further research on incorporating time aspect into GKF can draw inspiration from the development of matrix profile [130].

GKF can also benefit from further research on the autonomous assignment of configuration parameters. Operation of GKF requires setting several parameters that depend

on the distribution of the data under analysis. We have provided experimental results relating to the behaviour of GKF as these parameters vary. Further research on automating these parameter sensitivity experiments and deciding on suitable values for the parameters will be useful.

Using geographic information of the users' area may also help enhance the accuracy of tracking, which can be considered in future work. This geographic information in the form of a publicly available map or generated travel network such as $SSG(d)$, can be used to limit the possible movement space of the GKF predictions.

Bibliography

[1] R. Jurdak, K. Zhao, J. Liu, M. AbouJaoude, M. A. Cameron, and D. Newth. "Understanding Human Mobility from Twitter". In: *CoRR* abs/1412.2154 (2014). arXiv: 1412.2154.

[2] A. M. MacEachren et al. "SensePlace2: GeoTwitter analytics support for situational awareness". In: *2011 IEEE Conference on Visual Analytics Science and Technology, VAST 2011, Providence, Rhode Island, USA, October 23-28, 2011*. IEEE Computer Society, 2011, pp. 181–190.

[3] J. Chae, D. Thom, Y. Jang, S. Kim, T. Ertl, and D. S. Ebert. "Public behavior response analysis in disaster events utilizing visual analytics of microblog data". In: *Computers & Graphics* 38 (2014), pp. 51–60.

[4] P. Katsikouli, A. C. Viana, M. Fiore, and A. Tarable. "On the Sampling Frequency of Human Mobility". In: *2017 IEEE Global Communications Conference, GLOBECOM 2017, Singapore, December 4-8, 2017*. IEEE, 2017, pp. 1–6.

[5] J. W. Jaromczyk and G. T. Toussaint. "Relative neighborhood graphs and their relatives". In: *Proceedings of the IEEE* 80.9 (1992), pp. 1502–1517.

[6] J. Cardinal, S. Collette, and S. Langerman. "Empty region graphs". In: *Computational geometry* 42.3 (2009), pp. 183–195.

[7] K. R. Gabriel and R. R. Sokal. "A new statistical approach to geographic variation analysis". In: *Systematic Biology* 18.3 (1969), pp. 259–278.

[8] D. W. Matula and R. R. Sokal. "Properties of Gabriel graphs relevant to geographic variation research and the clustering of points in the plane". In: *Geographical analysis* 12.3 (1980), pp. 205–222.

[9] G. T. Toussaint. "The relative neighbourhood graph of a finite planar set". In: *Pattern Recognition* 12.4 (1980), pp. 261–268.

[10] A. Lingas. "A Linear-time Construction of the Relative Neighborhood Graph From the Delaunay Triangulation". In: *Comput. Geom.* 4 (1994), pp. 199–208.

[11] R. B. Urquhart. "Algorithms for computation of relative neighbourhood graph". In: *Electronics Letters* 16.14 (1980), pp. 556–557.

[12] G. T. Toussaint. "Comment: Algorithms for computing relative neighbourhood graph". In: *Electronics Letters* 16.22 (1980), pp. 860–860.

[13] D. V. Andrade and L. H. de Figueiredo. "Good approximations for the relative neighbourhood graph". In: *Proceedings of the 13th Canadian Conference on Computational Geometry, University of Waterloo, Ontario, Canada, August 13-15, 2001*. 2001, pp. 25–28.

[14] D. G. Kirkpatrick and J. D. Radke. "A framework for computational morphology". In: *Computational Geometry*. Elesevier, 1984, pp. 217–248.

[15] N. Amenta, M. W. Bern, and D. Eppstein. "The Crust and the beta-Skeleton: Combinatorial Curve Reconstruction". In: *Graphical Models and Image Processing* 60.2 (1998), pp. 125–135.

[16] P. Chew. "There are Planar Graphs Almost as Good as the Complete Graph". In: *J. Comput. Syst. Sci.* 39.2 (1989), pp. 205–219.

[17] M. de Berg, W. Meulemans, and B. Speckmann. "Delineating imprecise regions via shortest-path graphs". In: *19th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, ACM-GIS 2011, November 1-4, 2011, Chicago, IL, USA, Proceedings*. Ed. by I. F. Cruz, D. Agrawal, C. S. Jensen, E. Ofek, and E. Tanin. ACM, 2011, pp. 271–280.

[18] W. Meulemans, N. H. Riche, B. Speckmann, B. Alper, and T. Dwyer. "KelpFusion: A Hybrid Set Visualization Technique". In: *IEEE transactions on visualization and computer graphics* 19.11 (2013), pp. 1846–1858.

[19] L. Wei, Y. Zheng, and W. Peng. "Constructing popular routes from uncertain trajectories". In: *The 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '12, Beijing, China, August 12-16, 2012*. Ed. by Q. Yang, D. Agarwal, and J. Pei. ACM, 2012, pp. 195–203.

[20] S. M. Mousavi, A. Harwood, S. Karunasekera, and M. Maghrebi. "Geometry of interest (GOI): spatio-temporal destination extraction and partitioning in GPS trajectory data". In: *J. Ambient Intell. Humaniz. Comput.* 8.3 (2017), pp. 419–434.

[21] N. V. Andrienko and G. L. Andrienko. "Spatial Generalization and Aggregation of Massive Movement Data". In: *IEEE Trans. Vis. Comput. Graph.* 17.2 (2011), pp. 205–219.

[22] R. Lee, S. Wakamiya, and K. Sumiya. "Discovery of unusual regional social activities using geo-tagged microblogs". In: *World Wide Web* 14.4 (2011), pp. 321–349.

[23] S. An, H. Yang, J. Wang, N. Cui, and J. Cui. "Mining urban recurrent congestion evolution patterns from GPS-equipped vehicle mobility data". In: *Inf. Sci.* 373 (2016), pp. 515–526.

[24] S. Jiang, G. A. Fiore, Y. Yang, J. F. Jr., E. Frazzoli, and M. C. González. "A review of urban computing for mobile phone traces: current methods, challenges and opportunities". In: *Proceedings of the 2nd ACM SIGKDD International Workshop on Urban Computing, UrbComp@KDD 2013, Chicago, Illinois, USA, August 11, 2013*. Ed. by Y. Zheng, S. E. Koonin, and O. E. Wolfson. ACM, 2013, 2:1–2:9.

[25] C. Smith-Clarke, D. Quercia, and L. Capra. "Finger on the pulse: identifying deprivation using transit flow analysis". In: *Computer Supported Cooperative Work, CSCW 2013, San Antonio, TX, USA, February 23-27, 2013*. Ed. by A. Bruckman, S. Counts, C. Lampe, and L. G. Terveen. ACM, 2013, pp. 683–692.

[26] S. M. Mousavi, A. Harwood, S. Karunasekera, and M. Maghrebi. "Enhancing the quality of geometries of interest (GOIs) extracted from GPS trajectory data using spatio-temporal data aggregation and outlier detection". In: *J. Ambient Intell. Humaniz. Comput.* 9.1 (2018), pp. 173–186.

[27] S. He et al. "RoadRunner: improving the precision of road network inference from GPS trajectories". In: *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2018, Seattle, WA, USA, November 06-09, 2018*. Ed. by F. B. Kashani, E. G. Hoel, R. H. Güting, R. Tamassia, and L. Xiong. ACM, 2018, pp. 3–12.

[28] Z. Yan, D. Chakraborty, C. Parent, S. Spaccapietra, and K. Aberer. "Semantic trajectories: Mobility data computation and annotation". In: *ACM TIST* 4.3 (2013), 49:1–49:38.

[29] Y. Zheng, L. Zhang, X. Xie, and W. Ma. "Mining interesting locations and travel sequences from GPS trajectories". In: *Proceedings of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain, April 20-24, 2009*. Ed. by J. Quemada, G. León, Y. S. Maarek, and W. Nejdl. ACM, 2009, pp. 791–800.

[30] D. Guo, S. Liu, and H. Jin. "A graph-based approach to vehicle trajectory analysis". In: *J. Location Based Services* 4.3&4 (2010), pp. 183–199.

[31] J. Cortés, S. Martínez, and F. Bullo. "Robust rendezvous for mobile autonomous agents via proximity graphs in arbitrary dimensions". In: *IEEE Transactions on Automatic Control* 51.8 (2006), pp. 1289–1298.

[32] P. Reunanen, A. Fall, and A. Nikula. "Spatial graphs as templates for habitat networks in boreal landscapes". In: *Biodiversity and conservation* 21.14 (2012), pp. 3569–3584.

[33] D. Naboulsi, R. Stanica, and M. Fiore. "Classifying call profiles in large-scale mobile traffic datasets". In: *2014 IEEE Conference on Computer Communications, INFOCOM 2014, Toronto, Canada, April 27 - May 2, 2014*. IEEE, 2014, pp. 1806–1814.

[34] A Lima, M De Domenico, V Pejovic, and M. Musolesi. "Disease containment strategies based on mobility and information dissemination". In: *Scientific reports* 5 (2015), p. 10650.

[35] G. Chen, S. Hoteit, A. C. Viana, M. Fiore, and C. Sarraute. "Enriching sparse mobility information in Call Detail Records". In: *Comput. Commun.* 122 (2018), pp. 44–58.

[36] C. Song, Z. Qu, N. Blumm, and A.-L. Barabási. "Limits of predictability in human mobility". In: *Science* 327.5968 (2010), pp. 1018–1021.

[37] M. C. González, C. A. Hidalgo, and A. Barabási. "Understanding individual human mobility patterns". In: *CoRR* abs/0806.1256 (2008). arXiv: 0806.1256.

[38] X. Wei and X. A. Yao. "A Conceptual Framework for Representation of Location-based Social Media Activities (Short Paper)". In: *10th International Conference on Geographic Information Science, GIScience 2018, August 28-31, 2018, Melbourne, Australia*. Ed. by S. Winter, A. Griffin, and M. Sester. Vol. 114. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018, 62:1–62:7.

[39] N. A. Rehman, K. Relia, and R. Chunara. "Creating full individual-level location timelines from sparse social media data". In: *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2018, Seattle, WA, USA, November 06-09, 2018*. Ed. by F. Banaei-Kashani, E. G. Hoel, R. H. Güting, R. Tamassia, and L. Xiong. ACM, 2018, pp. 379–388.

[40] P. Siriaraya et al. "Witnessing Crime through Tweets: A Crime Investigation Tool based on Social Media". In: *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2019, Chicago, IL, USA, November 5-8, 2019*. Ed. by F. B. Kashani, G. Trajcevski, R. H. Güting, L. Kulik, and S. D. Newsam. ACM, 2019, pp. 568–571.

[41] A. Ristea, O. Kounadi, and M. Leitner. "Geosocial Media Data as Predictors in a GWR Application to Forecast Crime Hotspots (Short Paper)". In: *10th International Conference on Geographic Information Science, GIScience 2018, August 28-31, 2018, Melbourne, Australia*. Ed. by S. Winter, A. Griffin, and M. Sester. Vol. 114. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018, 56:1–56:7.

[42] R. C. S. N. P. Souza, R. M. Assunção, D. B. Neill, and W. M. Jr. "Detecting Spatial Clusters of Disease Infection Risk Using Sparsely Sampled Social Media Mobility Patterns". In: *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2019, Chicago, IL, USA, November 5-8, 2019*. Ed. by F. B. Kashani, G. Trajcevski, R. H. Güting, L. Kulik, and S. D. Newsam. ACM, 2019, pp. 359–368.

[43] J. Xie, T. Yang, and G. Li. "Extracting Geospatial Information from Social Media Data for Hazard Mitigation, Typhoon Hato as Case Study (Short Paper)". In: *10th International Conference on Geographic Information Science, GIScience 2018, August 28-31, 2018, Melbourne, Australia*. Ed. by S. Winter, A. Griffin, and M. Sester. Vol. 114. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018, 65:1–65:6.

[44] T. H. Silva, P. O. S. V. de Melo, J. M. Almeida, J. F. S. Salles, and A. A. F. Loureiro. "A Picture of Instagram is Worth More Than a Thousand Words: Workload Characterization and Application". In: *IEEE International Conference on Distributed Computing in Sensor Systems, DCOSS 2013, Cambridge, MA, USA, May 20-23, 2013*. IEEE Computer Society, 2013, pp. 123–132.

[45] Z. Cheng, J. Caverlee, K. Lee, and D. Z. Sui. "Exploring Millions of Footprints in Location Sharing Services". In: *Proceedings of the Fifth International Conference on Weblogs and Social Media, Barcelona, Catalonia, Spain, July 17-21, 2011*. Ed. by L. A. Adamic, R. Baeza-Yates, and S. Counts. The AAAI Press, 2011.

[46] B. Jiang, D. Ma, J. Yin, and M. Sandberg. "Spatial distribution of city tweets and their densities". In: *Geographical Analysis* 48.3 (2016), pp. 337–351.

[47] P. A. Longley and M. Adnan. "Geo-temporal Twitter demographics". In: *International Journal of Geographical Information Science* 30.2 (2016), pp. 369–389.

[48] K. Leetaru, S. Wang, G. Cao, A. Padmanabhan, and E. Shook. "Mapping the global Twitter heartbeat: The geography of Twitter". In: *First Monday* 18.5 (2013).

[49] D. Falcone, C. Mascolo, C. Comito, D. Talia, and J. Crowcroft. "What is this place? Inferring place categories through user patterns identification in geo-tagged tweets". In: *6th International Conference on Mobile Computing, Applications and Services, MobiCASE 2014, Austin, TX, USA, November 6-7, 2014*. Ed. by C. Julien, N. D. Lane, and S. Mishra. IEEE, 2014, pp. 10–19.

[50] B. J. Hecht and M. Stephens. "A Tale of Cities: Urban Biases in Volunteered Geographic Information". In: *Proceedings of the Eighth International Conference on Weblogs and Social Media, ICWSM 2014, Ann Arbor, Michigan, USA, June 1-4, 2014*. Ed. by E. Adar, P. Resnick, M. D. Choudhury, B. Hogan, and A. H. Oh. The AAAI Press, 2014.

[51] K. Machado, T. H. Silva, P. O. S. V. de Melo, E. Cerqueira, and A. A. F. Loureiro. "Urban Mobility Sensing Analysis through a Layered Sensing Approach". In: *2015 IEEE International Conference on Mobile Services, MS 2015, New York City, NY, USA, June 27 - July 2, 2015*. Ed. by O. Altintas and J. Zhang. IEEE Computer Society, 2015, pp. 306–312.

[52] T. H. Silva et al. "Users in the urban sensing process: Challenges and research opportunities". In: *Pervasive Computing: Next Generation Platforms for Intelligent Data Collection* (2016), pp. 45–95.

[53] R. Lee and K. Sumiya. "Measuring geographical regularities of crowd behaviors for Twitter-based geo-social event detection". In: *Proceedings of the 2010 International Workshop on Location Based Social Networks, LBSN 2010, November 2, 2010, San Jose, CA, USA, Proceedings*. Ed. by X. Zhou, W. Lee, W. Peng, and X. Xie. ACM, 2010, pp. 1–10.

[54] D. Thom, H. Bosch, S. Koch, M. Wörner, and T. Ertl. "Spatiotemporal anomaly detection through visual analysis of geolocated Twitter messages". In: *2012 IEEE Pacific Visualization Symposium, PacificVis 2012, Songdo, Korea (South), February 28 - March 2, 2012*. Ed. by H. Hauser, S. G. Kobourov, and H. Qu. IEEE Computer Society, 2012, pp. 41–48.

[55] J. Chae et al. "Spatiotemporal social media analytics for abnormal event detection and examination using seasonal-trend decomposition". In: *2012 IEEE Conference on Visual Analytics Science and Technology, VAST 2012, Seattle, WA, USA, October 14-19, 2012*. IEEE Computer Society, 2012, pp. 143–152.

[56] A. Guille and C. Favre. "Event detection, tracking, and visualization in Twitter: a mention-anomaly-based approach". In: *Social Netw. Analys. Mining* 5.1 (2015), 18:1–18:18.

[57] B. Pan, Y. Zheng, D. Wilkie, and C. Shahabi. "Crowd sensing of traffic anomalies based on human mobility and social media". In: *21st SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2013, Orlando, FL, USA, November 5-8, 2013*. Ed. by C. A. Knoblock, M. Schneider, P. Kröger, J. Krumm, and P. Widmayer. ACM, 2013, pp. 334–343.

[58] H. Becker, M. Naaman, and L. Gravano. "Beyond Trending Topics: Real-World Event Identification on Twitter". In: *Proceedings of the Fifth International Conference on Weblogs and Social Media, Barcelona, Catalonia, Spain, July 17-21, 2011*. Ed. by L. A. Adamic, R. Baeza-Yates, and S. Counts. The AAAI Press, 2011.

[59] P. Arcaini, G. Bordogna, D. Ienco, and S. Sterlacchini. "User-driven geo-temporal density-based exploration of periodic and not periodic events reported in social networks". In: *Inf. Sci.* 340-341 (2016), pp. 122–143.

[60] H. Abdelhaq, C. Sengstock, and M. Gertz. "EvenTweet: Online Localized Event Detection from Twitter". In: *PVLDB* 6.12 (2013), pp. 1326–1329.

[61] J. Sankaranarayanan, H. Samet, B. E. Teitler, M. D. Lieberman, and J. Sperling. "TwitterStand: news in tweets". In: *17th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, ACM-GIS 2009, November 4-6, 2009, Seattle, Washington, USA, Proceedings*. Ed. by D. Agrawal et al. ACM, 2009, pp. 42–51.

[62] W. Herzallah, H. Faris, and O. Adwan. "Feature engineering for detecting spammers on Twitter: Modelling and analysis". In: *J. Inf. Sci.* 44.2 (2018), pp. 230–247.

[63] A. Albert and M. R. Hallowell. "Modeling the role of social networks in situational awareness and hazard communication". In: *Construction Research Congress 2014: Construction in a Global Network*. 2014, pp. 1752–1761.

[64] S. Vieweg, A. L. Hughes, K. Starbird, and L. Palen. "Microblogging during two natural hazards events: what twitter may contribute to situational awareness". In: *Proceedings of the 28th International Conference on Human Factors in Computing Systems, CHI 2010, Atlanta, Georgia, USA, April 10-15, 2010*. Ed. by E. D. Mynatt, D. Schoner, G. Fitzpatrick, S. E. Hudson, W. K. Edwards, and T. Rodden. ACM, 2010, pp. 1079–1088.

[65] A. Karami, V. Shah, R. Vaezi, and A. Bansal. "Twitter speaks: A case of national disaster situational awareness". In: *J. Inf. Sci.* 46.3 (2020).

[66] P. T. Metaxas and E. Mustafaraj. "Social media and the elections". In: *Science* 338.6106 (2012), pp. 472–473.

[67] A. Tumasjan, T. O. Sprenger, P. G. Sandner, and I. M. Welpe. "Predicting Elections with Twitter: What 140 Characters Reveal about Political Sentiment". In: *Proceedings of the Fourth International Conference on Weblogs and Social Media, ICWSM 2010, Washington, DC, USA, May 23-26, 2010*. Ed. by W. W. Cohen and S. Gosling. The AAAI Press, 2010.

[68] L. Shi, N. Agarwal, A. Agrawal, R. Garg, and J. Spoelstra. "Predicting US primary elections with Twitter". In: *workshop social network and social media analysis: methods, models and applications (NIPS)* (2012), pp. 1–8.

[69] M. Choy, M. L. F. Cheong, M. N. Laik, and K. P. Shung. "A sentiment analysis of Singapore Presidential Election 2011 using Twitter data with census correction". In: *CoRR* abs/1108.5520 (2011). arXiv: 1108.5520.

[70] K. Joseph, C. H. Tan, and K. M. Carley. "Beyond "local", "categories" and "friends": clustering foursquare users with latent "topics"". In: *The 2012 ACM Conference on Ubiquitous Computing, Ubicomp '12, Pittsburgh, PA, USA, September 5-8, 2012*. Ed. by A. K. Dey, H. Chu, and G. R. Hayes. ACM, 2012, pp. 919–926.

[71] A. Noulas, S. Scellato, C. Mascolo, and M. Pontil. "Exploiting Semantic Annotations for Clustering Geographic Areas and Users in Location-based Social Networks". In: *The Social Mobile Web, Papers from the 2011 ICWSM Workshop, Barcelona, Catalonia, Spain, July 21, 2011*. Vol. WS-11-02. AAAI Workshops. AAAI, 2011.

[72] M. Ye, D. Shou, W. Lee, P. Yin, and K. Janowicz. "On the semantic annotation of places in location-based social networks". In: *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21-24, 2011*. Ed. by C. Apté, J. Ghosh, and P. Smyth. ACM, 2011, pp. 520–528.

[73] R. Fileto, C. May, C. Renso, N. Pelekis, D. Klein, and Y. Theodoridis. "The Baquara^2 knowledge-based framework for semantic enrichment and analysis of movement data". In: *Data Knowl. Eng.* 98 (2015), pp. 104–122.

[74] Y. Zheng. "Tutorial on location-based social networks". In: *Proceedings of the 21st international conference on World wide web, WWW*. Vol. 12. 5. 2012.

[75] Y. Zheng. "Location-Based Social Networks: Users". In: *Computing with Spatial Trajectories*. Ed. by Y. Zheng and X. Zhou. Springer, 2011, pp. 243–276.

[76] Y. Zheng, Z. Zha, and T. Chua. "Mining Travel Patterns from Geotagged Photos". In: *ACM TIST* 3.3 (2012), 56:1–56:18.

[77] D. J. Crandall, L. Backstrom, D. P. Huttenlocher, and J. M. Kleinberg. "Mapping the world's photos". In: *Proceedings of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain, April 20-24, 2009*. Ed. by J. Quemada, G. León, Y. S. Maarek, and W. Nejdl. ACM, 2009, pp. 761–770.

[78] M. D. Choudhury, M. Feldman, S. Amer-Yahia, N. Golbandi, R. Lempel, and C. Yu. "Automatic construction of travel itineraries using social breadcrumbs". In: *HT'10, Proceedings of the 21st ACM Conference on Hypertext and Hypermedia, Toronto, Ontario, Canada, June 13-16, 2010*. Ed. by M. H. Chignell and E. G. Toms. ACM, 2010, pp. 35–44.

[79] I. R. Brilhante, J. A. F. de Macêdo, F. M. Nardini, R. Perego, and C. Renso. "Where shall we go today?: planning touristic tours with tripbuilder". In: *22nd ACM International Conference on Information and Knowledge Management, CIKM'13, San Francisco, CA, USA, October 27 - November 1, 2013*. Ed. by Q. He, A. Iyengar, W. Nejdl, J. Pei, and R. Rastogi. ACM, 2013, pp. 757–762.

[80] H. Hsieh, C. Li, and S. Lin. "Exploiting large-scale check-in data to recommend time-sensitive routes". In: *Proceedings of the ACM SIGKDD International Workshop on Urban Computing, UrbComp@KDD 2012, Beijing, China, August 12, 2012*. Ed. by O. E. Wolfson and Y. Zheng. ACM, 2012, pp. 55–62.

[81] K. Meehan, T. Lunney, K. Curran, and A. McCaughey. "Context-aware intelligent recommendation system for tourism". In: *2013 IEEE International Conference on Pervasive Computing and Communications Workshops, PERCOM 2013 Workshops, San Diego, CA, USA, March 18-22, 2013*. IEEE Computer Society, 2013, pp. 328–331.

[82] D. Quercia, R. Schifanella, and L. M. Aiello. "The shortest path to happiness: recommending beautiful, quiet, and happy routes in the city". In: *25th ACM Conference on Hypertext and Social Media, HT '14, Santiago, Chile, September 1-4, 2014*. Ed. by L. Ferres, G. Rossi, V. A. F. Almeida, and E. Herder. ACM, 2014, pp. 116–125.

[83] M. Sklar, B. Shaw, and A. Hogue. "Recommending interesting events in real-time with foursquare check-ins". In: *Sixth ACM Conference on Recommender Systems, RecSys '12, Dublin, Ireland, September 9-13, 2012*. Ed. by P. Cunningham, N. J. Hurley, I. Guy, and S. S. Anand. ACM, 2012, pp. 311–312.

[84] Z. Chen, H. T. Shen, and X. Zhou. "Discovering popular routes from trajectories". In: *Proceedings of the 27th International Conference on Data Engineering, ICDE 2011, April 11-16, 2011, Hannover, Germany*. Ed. by S. Abiteboul, K. Böhm, C. Koch, and K. Tan. IEEE Computer Society, 2011, pp. 900–911.

[85] W. Luo, H. Tan, L. Chen, and L. M. Ni. "Finding time period-based most frequent path in big trajectory data". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2013, New York, NY, USA, June 22-27, 2013*. Ed. by K. A. Ross, D. Srivastava, and D. Papadias. ACM, 2013, pp. 713–724.

[86] S. Jiang, X. Qian, J. Shen, Y. Fu, and T. Mei. "Author Topic Model-Based Collaborative Filtering for Personalized POI Recommendations". In: *IEEE Trans. Multim.* 17.6 (2015), pp. 907–918.

[87] T. H. Silva, P. O. S. V. de Melo, J. M. Almeida, A. C. Viana, J. F. S. Salles, and A. A. F. Loureiro. "Participatory Sensor Networks as Sensing Layers". In: *2014 IEEE Fourth International Conference on Big Data and Cloud Computing, BDCloud 2014, Sydney, Australia, December 3-5, 2014*. IEEE Computer Society, 2014, pp. 386–393.

[88] Y. Zheng, T. Liu, Y. Wang, Y. Zhu, Y. Liu, and E. Chang. "Diagnosing New York city's noises with ubiquitous data". In: *The 2014 ACM Conference on Ubiquitous Computing, UbiComp '14, Seattle, WA, USA, September 13-17, 2014*. Ed. by A. J. Brush, A. Friday, J. A. Kientz, J. Scott, and J. Song. ACM, 2014, pp. 715–725.

[89] N. Maisonneuve, M. Stevens, M. E. Niessen, and L. Steels. "NoiseTube: Measuring and mapping noise pollution with mobile phones". In: *Information Technologies in Environmental Engineering, Proceedings of the 4th International ICSC Symposium, ITEE 2009, Thessaloniki, Greece, May 28-29, 2009*. Ed. by I. N. Athanasiadis, P. A. Mitkas, A. E. Rizzoli, and J. M. Gómez. Springer, 2009, pp. 215–228.

[90] E. D'Hondt, M. Stevens, and A. Jacobs. "Participatory noise mapping works! An evaluation of participatory sensing as an alternative to standard techniques for environmental monitoring". In: *Pervasive Mob. Comput.* 9.5 (2013), pp. 681–694.

[91] D. Quercia, R. Schifanella, L. M. Aiello, and K. McLean. "Smelly Maps: The Digital Life of Urban Smellscapes". In: *Proceedings of the Ninth International Conference on Web and Social Media, ICWSM 2015, University of Oxford, Oxford, UK, May 26-29, 2015*. Ed. by M. Cha, C. Mascolo, and C. Sandvig. AAAI Press, 2015, pp. 327–336.

[92] D. Kershaw, M. Rowe, and P. Stacey. "Towards tracking and analysing regional alcohol consumption patterns in the UK through the use of social media". In: *ACM Web Science Conference, WebSci '14, Bloomington, IN, USA, June 23-26, 2014*. Ed. by F. Menczer, J. Hendler, W. H. Dutton, M. Strohmaier, C. Cattuto, and E. T. Meyer. ACM, 2014, pp. 220–228.

[93] A. Llorente, M. García-Herranz, M. Cebrián, and E. Moro. "Social media fingerprints of unemployment". In: *CoRR* abs/1411.3140 (2014). arXiv: 1411.3140.

[94] F. A. Santos, T. H. Silva, T. Braun, A. A. F. Loureiro, and L. A. Villas. "Towards a sustainable people-centric sensing". In: *IEEE International Conference on Communications, ICC 2017, Paris, France, May 21-25, 2017*. IEEE, 2017, pp. 1–6.

[95] H. Aly, J. Krumm, G. Ranade, and E. Horvitz. "On the value of spatiotemporal information: principles and scenarios". In: *Proceedings of the 26th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2018, Seattle, WA, USA, November 06-09, 2018*. Ed. by F. B. Kashani, E. G. Hoel, R. H. Güting, R. Tamassia, and L. Xiong. ACM, 2018, pp. 179–188.

[96] K. Joseph, P. M. Landwehr, and K. M. Carley. "Two 1%s Don't Make a Whole: Comparing Simultaneous Samples from Twitter's Streaming API". In: *Social Computing, Behavioral-Cultural Modeling and Prediction - 7th International Conference, SBP 2014, Washington, DC, USA, April 1-4, 2014. Proceedings*. Ed. by W. G. Kennedy, N. Agarwal, and S. J. Yang. Vol. 8393. Lecture Notes in Computer Science. Springer, 2014, pp. 75–83.

[97] J. Lee, J. Han, and K. Whang. "Trajectory clustering: a partition-and-group framework". In: *Proceedings of the ACM SIGMOD International Conference on Management of Data, Beijing, China, June 12-14, 2007*. Ed. by C. Y. Chan, B. C. Ooi, and A. Zhou. ACM, 2007, pp. 593–604.

[98] C. Hung, W. Peng, and W. Lee. "Clustering and aggregating clues of trajectories for mining trajectory patterns and routes". In: *VLDB J.* 24.2 (2015), pp. 169–192.

[99] M. Benkert, J. Gudmundsson, F. Hübner, and T. Wolle. "Reporting flock patterns". In: *Comput. Geom.* 41.3 (2008), pp. 111–125.

[100] M. Benkert, J. Gudmundsson, F. Hübner, and T. Wolle. "Reporting Flock Patterns". In: *Algorithms - ESA 2006, 14th Annual European Symposium, Zurich, Switzerland, September 11-13, 2006, Proceedings*. Ed. by Y. Azar and T. Erlebach. Vol. 4168. Lecture Notes in Computer Science. Springer, 2006, pp. 660–671.

[101] M. R. Vieira, P. Bakalov, and V. J. Tsotras. "On-line discovery of flock patterns in spatio-temporal data". In: *17th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, ACM-GIS 2009, November 4-6, 2009, Seattle, Washington, USA, Proceedings*. Ed. by D. Agrawal et al. ACM, 2009, pp. 286–295.

[102] M. Wirz, P. Schläpfer, M. B. Kjærgaard, D. Roggen, S. Feese, and G. Tröster. "Towards an online detection of pedestrian flocks in urban canyons by smoothed spatio-temporal clustering of GPS trajectories". In: *Proceedings of the 2011 International Workshop on Location Based Social Networks, LBSN 2011, November 1, 2011, Chicago, IL, USA, Proceedings*. ACM, 2011, pp. 17–24.

[103] C. S. Jensen, D. Lin, Beng Chin Ooi, and Rui Zhang. "Effective Density Queries on Continuously Moving Objects". In: *22nd International Conference on Data Engineering (ICDE'06)*. 2006, pp. 71–71.

[104] L. Tu and Y. Chen. "Stream data clustering based on grid density and attraction". In: *ACM Trans. Knowl. Discov. Data* 3.3 (2009), 12:1–12:27.

[105] H. Jeung, M. L. Yiu, X. Zhou, C. S. Jensen, and H. T. Shen. "Discovery of convoys in trajectory databases". In: *PVLDB* 1.1 (2008), pp. 1068–1080.

[106] D. Amores, M. Vasardani, and E. Tanin. "Early Detection of Herding Behaviour during Emergency Evacuations". In: *10th International Conference on Geographic Information Science, GIScience 2018, August 28-31, 2018, Melbourne, Australia*. Ed. by S. Winter, A. Griffin, and M. Sester. Vol. 114. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018, 1:1–1:15.

[107] F. Zhao and L. Guibas. "Canonical Problem:Localization and Tracking". In: *Wireless Sensor Networks: An Information Processing Approach*. Morgan Kaufmann publishers, 2004. Chap. 2, pp. 23–62.

[108] R. G. Brown and P. Y. C. Hwang. "Signal Processing". In: *Introduction to Random Signals and Applied Kalman Filtering with Matlab Exercises*. Wiley Education Services, 2012. Chap. 6, pp. 207–248.

[109] X. Li, K. Wang, W. Wang, and Y. Li. "A multiple object tracking method using Kalman filter". In: *The 2010 IEEE International Conference on Information and Automation*. 2010, pp. 1862–1866.

[110] X. Chen, X. Wang, and J. Xuan. "Tracking Multiple Moving Objects Using Unscented Kalman Filtering Techniques". In: *CoRR* abs/1802.01235 (2018).

[111] N. Wiener. *Extrapolation, interpolation, and smoothing of stationary time series: with engineering applications*. MIT press Cambridge, 1950.

[112] P. Del Moral. "Nonlinear filtering: Interacting particle resolution". In: *Comptes Rendus de l'Académie des Sciences-Series I-Mathematics* 325.6 (1997), pp. 653–658.

[113] D. G. Kirkpatrick and J. D. Radke. "A framework for computational morphology". In: *Machine Intelligence and Pattern Recognition*. Vol. 2. Elsevier, 1985, pp. 217–248.

[114] G. T. Toussaint. "The relative neighbourhood graph of a finite planar set". In: *Pattern recognition* 12.4 (1980), pp. 261–268.

[115] P. Bose, L. Devroye, W. S. Evans, and D. G. Kirkpatrick. "On the Spanning Ratio of Gabriel Graphs and beta-Skeletons". In: *SIAM J. Discrete Math.* 20.2 (2006), pp. 412–427.

[116] H. Jeung, M. L. Yiu, and C. S. Jensen. "Trajectory Pattern Mining". In: *Computing with Spatial Trajectories*. Ed. by Y. Zheng and X. Zhou. Springer, 2011, pp. 143–177.

[117] B. Thomee et al. "YFCC100M: the new data in multimedia research". In: *Commun. ACM* 59.2 (2016), pp. 64–73.

[118] K. Ramamohanarao et al. "SMARTS: Scalable Microscopic Adaptive Road Traffic Simulator". In: *ACM TIST* 8.2 (2017), 26:1–26:22.

[119] D. Sinclair. "S-hull: a fast radial sweep-hull routine for Delaunay triangulation". In: *CoRR* abs/1604.01428 (2016).

[120] D. Sacharidis, P. Bouros, and T. Chondrogiannis. "Finding The Most Preferred Path". 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*. Ed. by E. G. Hoel, S. D. Newsam, S. Ravada, R. Tamassia, and G. Trajcevski. ACM, 2017, 5:1–5:10.

[121] K. H. Lim, J. Chan, C. Leckie, and S. Karunasekera. "Personalized Tour Recommendation Based on User Interests and Points of Interest Visit Durations". In: *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*. Ed. by Q. Yang and M. Wooldridge. AAAI Press, 2015, pp. 1778–1784.

[122] D. Sarkar, R. Sieber, and R. Sengupta. "GIScience Considerations in Spatial Social Networks". In: *Geographic Information Science - 9th International Conference, GIScience 2016, Montreal, QC, Canada, September 27-30, 2016, Proceedings*. Ed. by J. A. Miller, D. O'Sullivan, and N. Wiegand. Vol. 9927. Lecture Notes in Computer Science. Springer, 2016, pp. 85–98.

[123] R. E. Kalman. "A new approach to linear filtering and prediction problems". In: *Journal of basic Engineering* 82.1 (1960), pp. 35–45.

[124] R. G. Brown and P. Y. C. Hwang. "Canonical Problem: Localization and Tracking". In: *Introduction to Random Signals and Applied Kalman Filtering with Matlab Exercises, 4th Edition*. John Wiley and Sons Inc., 2012. Chap. 4, pp. 141–172.

[125] S. Yang and M. Baum. "Second-order extended Kalman filter for extended object and group tracking". In: *19th International Conference on Information Fusion, FUSION 2016, Heidelberg, Germany, July 5-8, 2016*. IEEE, 2016, pp. 1178–1184.

[126] I. Kang, T. Kim, and K. Li. "A Spatial Data Mining Method by Delaunay Triangulation". In: *GIS '97. Proceedings of the 5th International Workshop on Advances in Geographic Information Systems, November 13-14, 1997, Las Vegas, Nevada, USA*. ACM, 1997, pp. 35–39.

[127] R. Zhou. "Pedestrian dead reckoning on smartphones with varying walking speed". In: *2016 IEEE International Conference on Communications, ICC 2016, Kuala Lumpur, Malaysia, May 22-27, 2016*. IEEE, 2016, pp. 1–6.

[128] F. Zhao, H. Sun, J. Wu, Z. Gao, and R. Liu. "Analysis of road network pattern considering population distribution and central business district". In: *PloS one* 11.3 (2016), e0151676.

[129] J. C. Park, H. Shin, and B. K. Choi. "Elliptic Gabriel graph for finding neighbors in a point set and its application to normal vector estimation". In: *Computer-Aided Design* 38.6 (2006), pp. 619–626.

[130] C. M. Yeh et al. "Matrix Profile I: All Pairs Similarity Joins for Time Series: A Unifying View That Includes Motifs, Discords and Shapelets". In: *IEEE 16th International Conference on Data Mining, ICDM 2016, December 12-15, 2016, Barcelona, Spain*. Ed. by F. Bonchi, J. Domingo-Ferrer, R. Baeza-Yates, Z. Zhou, and X. Wu. IEEE Computer Society, 2016, pp. 1317–1322.

[131] S. Winter, A. Griffin, and M. Sester, eds. *10th International Conference on Geographic Information Science, GIScience 2018, August 28-31, 2018, Melbourne, Australia*. Vol. 114. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2018.

[132] F. B. Kashani, G. Trajcevski, R. H. Güting, L. Kulik, and S. D. Newsam, eds. *Proceedings of the 27th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems, SIGSPATIAL 2019, Chicago, IL, USA, November 5-8, 2019*. ACM, 2019.