Techniques for Spatio-temporal Analysis of Trajectory Data

by

PRAVEEN KUMAR TRIPATHI

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

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT ARLINGTON
September 2016

To my mentors, family and my friends.

ACKNOWLEDGEMENTS

I would like to thank my supervising professor Dr. Ramez Elmasri for constantly motivating and encouraging me, and also for his invaluable advice during the course of my doctoral studies. I wish to thank my committee members Dr. Gautam Das, Dr. Chengkai Li, Dr. Manfred Huber and Dr. Leonidas Fegaras for their interest in my research and for taking time to serve in my dissertation committee.

I am grateful to all the professors who taught me during my graduate studies in University of Texas at Arlington.

Finally, I would like to express my deep gratitude to my mother, father, wife and in-laws for their sacrifice, encouragement, patience and inspiration during my PhD work. I am also extremely grateful to several of my friends who have helped me throughout my career. At last but not the least, I thank my daughter for her sacrifice and understanding, she has been a great source of encouragement and motivation for me.

September 7, 2016

ABSTRACT

Techniques for Spatio-temporal Analysis of Trajectory Data

Praveen Kumar Tripathi, Ph.D.

The University of Texas at Arlington, 2016

Supervising Professor: Ramez Elmasri

The goal of this thesis is to develop novel techniques for the analysis of spatio-temporal trajectory data. Recent advances in tracking devices such as Global Positioning System (GPS) and mobile phones have resulted in an abundance of spatio-temporal trajectory data. Because of the importance of both the space and time dimensions in trajectory data, different analysis techniques are needed. The analysis techniques explored in this thesis involve two variations: point based analysis and trajectory based analysis.

Point based analysis has been done to identify the hot spots in the trajectory dataset (hurricane data). Here we consider the trajectory data as a point data set, comprised of specific points along the trajectory where line segments start and end. This analysis involves different combinations of spatial, temporal and non-spatial attributes. We use density based clustering algorithms DBSCAN to identify these hot spots. We extend the DBSCAN algorithm to incorporate non-spatial attributes (for example, wind speed and time) with spatial attributes. This approach has also been used to identify the starting region and the landing regions of hurricanes.

In the trajectory based analysis, we focus on trajectory simplification (smoothing), outlier filtration and directional analysis. Some trajectory data sets are noisy in nature, therefore we need some pre-processing to remove the noise. In the pre-processing stage, we smooth the trajectories to remove some trajectory points to obtain smooth and directionally consistent trajectories. We propose methods for smoothing the trajectories considering the directional attribute, and then propose a framework for the directional analysis of the trajectory data to obtain the directional patterns.

The framework involves segmentation of the smooth trajectories into directionally consistent categories of sub-trajectories. We identify 16 directional categories for this task. After this stage of the framework, we performs outlier filtration using a novel convex hull based approach. The outlier filtration stage is followed by a clustering algorithm on these sub-trajectories to obtain the directional patterns. For the evaluation of the proposed framework we used two real datasets: a hurricane data set and an animal movement data set.

TABLE OF CONTENTS (Draft)

ACKNOWLEDGEMENTS iv

ABSTRACT v

LIST OF ILLUSTRATIONS x

LIST OF TABLES xv

Chapter Page

1. Introduction 1

2. Related Work and Background 5

2.1 Related Work 5

2.2 Background 7

2.2.1 Trajectory data 7

2.2.1.1 Trajectory Definition 8

2.2.2 DBSCAN 9

2.3 Summary 12

3. Extracting Dense Regions From Hurricane Trajectory Data (Drafted from [1]) 13

3.1 Introduction 13

3.2 Related work 14

3.3 Trajectory Clustering on Point Data 16

3.3.1 DBSCAN 16

3.3.2 Dense region extraction 19

3.3.2.1 Spatial Clustering 19

3.3.2.2 Spatial and non spatial clustering 21

3.3.2.3 Spatio-temporal clustering 22
3.4 Experiments and Analysis 23
3.4.1 Analyzing spatial attributes 23
3.4.2 Qualitative analysis of clusters 26
3.4.2.1 Analyzing combination of spatial and non-spatial attributes 27
3.4.3 Analyzing storm starting and landing information 36
3.4.4 Analyzing storms for 10 year's duration 37
3.4.5 Grid based clustering algorithm 38
3.5 Summary 48
4. A Direction Based Framework for Trajectory Data Analysis (Drafted from [2]) 50
4.1 Introduction 50
4.2 Related Work 54
4.3 Proposed framework 55
4.3.1 Trajectory Smoothing 58
4.3.2 Trajectory Segmentation 59
4.3.3 Convex Hull based trajectory filtering to remove outliers . . . 62
4.3.4 Clustering 63
4.4 Experiments and Results 65
4.4.1 Qualitative analysis of clusters: 72
4.5 Summary 72
5. Trajectory Smoothing and Simplification 80
5.1 Introduction 80
5.2 Preliminaries 81
5.3 Trajectory Smoothing 82

5.3.1 Trajectory smoothness measure($sm\,(Tr)$): 82
5.3.2 3 - points trajectory smoothing : $3PtSpTraj$ 82
5.3.3 5 - points trajectory smoothing : $5PtSpTraj$ 85
5.4 Experiments and Results 88
5.5 Summary 89
6. Conclusion 93
Appendix
REFERENCES 95
BIOGRAPHICAL STATEMENT 98

LIST OF ILLUSTRATIONS

Figure Page

2.1 Trajectory example 8

2.2 DBSCAN Types of data points: Core points, Boundary points and Outliers 10

3.1 DBSCAN Types of data points: Core points, Boundary points and Outliers 17

3.2 DBSCAN spatial clusters: $Min_{Pts} = 10$, $Min_{Dst} = 35$ 23

3.3 DBSCAN spatial clusters: $Min_{Pts} = 8$, $Min_{Dst} = 30$ 25

3.4 DBSCAN spatial and wind clusters: $Min_{Pts} = 12$, $Min_{Dst} = 35$ 25

3.5 DBSCAN spatial and wind clusters: $Min_{Pts} = 10$, $Min_{Dst} = 35$, $Min_{DstWspeed} = 20$ 28

3.6 DBSCAN spatial and wind clusters: $Min_{Pts} = 10$, $Min_{Dst} = 35$, $Min_{DstWspeed} = 30$ 28

3.7 DBSCAN spatial and wind clusters: $Min_{Pts} = 10$, $Min_{Dst} = 35$, $Min_{DstWspeed} = 40$ 29

3.8 DBSCAN spatial and wind clusters: $Min_{Pts} = 10$, $Min_{Dst} = 35$, $Min_{DstWspeed} = 50$ 29

3.9 DBSCAN spatial and wind clusters: $Min_{Pts} = 10$, $Min_{Dst} = 35$, $Min_{DstWspeed} = 60$ 30

3.10 DBSCAN spatial and wind clusters: $Min_{Pts} = 10$, $Min_{Dst} = 35$, $Min_{DstWspeed} = 70$ 30

3.11 DBSCAN spatial and wind clusters: $Min_{Pts} = 10$, $Min_{Dst} = 35$, $Min_{DstWspeed} = 80$ 31
3.12 DBSCAN spatial and wind clusters: $Min_{Pts} = 10$, $Min_{Dst} = 35$, $Min_{DstWspeed} = 90$ 31
3.13 DBSCAN spatial and wind clusters: $Min_{Pts} = 10$, $Min_{Dst} = 35$, $Min_{DstWspeed} = 100$ 32
3.14 DBSCAN spatio-temporal clusters: $k = 10$, $Min_{Dst} = 35$, $\epsilon_t = 0.2$ . . . 32
3.15 DBSCAN spatio-temporal clusters: $k = 10$, $Min_{Dst} = 35$, $\epsilon_t = 0.3$ . . . 32
3.16 DBSCAN spatio-temporal clusters: $k = 10$, $Min_{Dst} = 35$, $\epsilon_t = 0.4$ . . . 33
3.17 DBSCAN spatio-temporal clusters: $k = 10$, $Min_{Dst} = 35$, $\epsilon_t = 0.5$ . . . 34
3.18 DBSCAN spatio-temporal clusters: $k = 10$, $Min_{Dst} = 35$, $\epsilon_t = 0.6$ . . . 34
3.19 DBSCAN spatio-temporal clusters: $k = 10$, $Min_{Dst} = 35$, $\epsilon_t = 0.7$ . . . 35
3.20 DBSCAN spatio-temporal clusters: $k = 10$, $Min_{Dst} = 35$, $\epsilon_t = 1.0$ . . . 36
3.21 Storm starting clusters 37
3.22 Storm Landing clusters 37
3.23 Storm activity ground truth 38
3.24 DBSCAN on Hurricane data for year 1950 to 1960 $Min_{Pts} = 10$, $Min_{Dst} = 35$ 39
3.25 DBSCAN on Hurricane data for year 1961 to 1970 $Min_{Pts} = 10$, $Min_{Dst} = 35$ 39
3.26 DBSCAN on Hurricane data for year 1971 to 1980 $Min_{Pts} = 10$, $Min_{Dst} = 35$ 40
3.27 DBSCAN on Hurricane data for year 1981 to 1990 $Min_{Pts} = 10$, $Min_{Dst} = 35$ 40
3.28 DBSCAN on Hurricane data for year 1990 to 2000 $Min_{Pts} = 10$, $Min_{Dst} = 35$ 41

3.29 CLIQUE $\tau = 0.1$, $grid\_size = 30 * 30$ 41
3.30 CLIQUE $\tau = 0.1$, $grid\_size = 40 * 40$ 41
3.31 CLIQUE $\tau = 0.1$, $grid\_size = 50 * 50$ 42
3.32 CLIQUE $\tau = 0.1$, $grid\_size = 60 * 60$ 42
3.33 CLIQUE $\tau = 0.1$, $grid\_size = 70 * 70$ 43
3.34 CLIQUE $\tau = 0.2$, $grid\_size = 40 * 40$ 43
3.35 CLIQUE $\tau = 0.2$, $grid\_size = 70 * 70$ 44
3.36 CLIQUE $\tau = 0.5$, $grid\_size = 30 * 30$ 44
3.37 CLIQUE $\tau = 0.1$, $grid\_size = 30 * 30$, $Min_{DstWspeed} = 30$ 45
3.38 CLIQUE $\tau = 0.1$, $grid\_size = 40 * 40$, $Min_{DstWspeed} = 40$ 46
3.39 CLIQUE $\tau = 0.1$, $grid\_size = 30 * 30$, $\epsilon_t = 0.1$ 46
3.40 CLIQUE Storm starting locations 47
3.41 CLIQUE Storm landing locations 47
4.1 Small angles (45) 52
4.2 Large angles (90) 53
4.3 MBR Example 53
4.4 CH Example 54
4.5 Directional pattern 56
4.6 Self intersecting Trajectories 57
4.7 Basic directional encoding of Hurricane Love 1950 61
4.8 Basic to General direction encoding of Hurricane Love 1950 61
4.9 Convex Hull 63
4.10 Distance between two sub-trajectories 64
4.11 Trajectory clustering without filtering 66
4.12 Hurricane Data combined sub-trajectory clustering 66
4.13 Trajectory original data 66

4.14 Trajectory clustering using the framework 67
4.15 Convex Hull DBSCAN, Direction 1 68
4.16 MBR DBSCAN, Direction 1 68
4.17 Convex Hull DBSCAN, Direction 9 68
4.18 MBR DBSCAN, Direction 9 69
4.19 Convex Hull DBSCAN, Direction 10 69
4.20 MBR DBSCAN, Direction 10 69
4.21 Convex Hull DBSCAN, Direction 11 70
4.22 MBR DBSCAN, Direction 11 70
4.23 Convex Hull DBSCAN, Direction 12 70
4.24 MBR DBSCAN, Direction 12 71
4.25 Convex Hull DBSCAN, Direction 13 71
4.26 MBR DBSCAN, Direction 13 71
4.27 Animal data : Relative performance Direction 1 73
4.28 Animal data : Relative performance Direction 2 73
4.29 Animal data : Relative performance Direction 3 73
4.30 Animal data : Relative performance Direction 4 73
4.31 Animal data : Relative performance Direction 5 74
4.32 Animal data : Relative performance Direction 6 74
4.33 Animal data : Relative performance Direction 7 74
4.34 Animal data : Relative performance Direction 8 75
4.35 Animal data : Relative performance Direction 9 75
4.36 Animal data : Relative performance Direction 10 75
4.37 Animal data : Relative performance Direction 11 76
4.38 Animal data : Relative performance Direction 12 76
4.39 Animal data : Relative performance Direction 13 76

4.40 Animal data : Relative performance Direction 14 77
4.41 Animal data : Relative performance Direction 15 77
4.42 Animal data : Relative performance Direction 16 77
4.43 Hurricane Qmeasure 78
4.44 Animal Qmeasure 79
5.1 Trajectory simplification 83
5.2 Case 1 - 4 for step simplification 85
5.3 Case 5 - 8 for step simplification 87

LIST OF TABLES

Table Page

3.1 Storm clustering analysis, on Spatial clustering , $Min_{Pts} = 10$, $Min_{Dst} = 35$ 26

3.2 Storm Clustering analysis, Impact of Non spatial attribute $Min_{Pts} = 10$, $Min_{Dst} = 35$ 27

3.3 Qualitative measure of clustering results 33

3.4 Qualitative measure of clustering results 33

3.5 Qualitative measure of clustering results DBSCAN 46

3.6 Qualitative measure of clustering results CLIQUE 47

4.1 Effect of Smoothing on Elk data set 65

5.1 Elk trajectory data simplification $\epsilon_\theta = 5$ 89

5.2 Elk trajectory data simplification $\epsilon_\theta = 10$ 89

5.3 Elk trajectory data simplification $\epsilon_\theta = 15$ 90

5.4 Elk trajectory data simplification $\epsilon_\theta = 30$ 90

5.5 Deer trajectory data simplification $\epsilon_\theta = 5$ 90

5.6 Deer trajectory data simplification $\epsilon_\theta = 10$ 90

5.7 Deer trajectory data simplification $\epsilon_\theta = 30$ 91

5.8 Cattle trajectory data simplification $\epsilon_\theta = 5$ 91

5.9 Cattle trajectory data simplification $\epsilon_\theta = 10$ 91

5.10 Cattle trajectory data simplification $\epsilon_\theta = 15$ 91

5.11 Cattle trajectory data simplification $\epsilon_\theta = 30$ 92

CHAPTER 1

Introduction

Trajectory data of an object is obtained when its consecutive locations are monitored over a sequence of time in multi-dimension (two or three dimensional) space [3]. Some examples of applications involving trajectory data are: global positioning system (GPS), hurricane and storm tracking and animal movement. Due to the recent advances in mobile technology many trajectory data sets have become readily available. The analysis for this data is vital in knowing and managing the traffic patterns of vehicles, monitoring and predicting weather conditions, examining wild animal behavior and movement. A number of attempts have been made in this domain to analyze these kinds of data set for example in [4–12]

In this thesis we propose novel techniques for the spatio-temporal analysis of trajectory data. These techniques basically involve point based and trajectory based concepts.

We perform point based analysis to extract dense regions in the trajectory data set to identify the prominent regions of activity. This analysis is done on hurricane data to identify the dense regions and hence the hot spots of the hurricane activity. This analysis involved clustering of the hurricane data while considering different combinations of spatial, non-spatial and temporal attributes. We used two density based clustering algorithms which are DBSCAN [13] and CLIQUE [14]. We further identify the regions from where the hurricanes start and the regions where the hurricanes land. This kind of point based analysis of the trajectory data (hurricane) is

novel. It is an important analysis as it provides some key information which would be very valuable to the domain experts.

Second, we propose a directional framework for trajectory data analysis. This framework considers the directional attribute of a trajectory as its most important attribute. We segment trajectories into directional sub-trajectories and group them into similar directional categories. We consider 16 directional categories in this work. This segmentation is done to keep the sub-trajectory directionally consistent. In each directional category we do outlier filtration using a novel convex hull based method. Finally we obtain the final directional patters as the clusters in the respective directional category. We applied this framework to hurricane and animal movement data. This kind of directional analysis has not been done before. It is a valuable analysis for map matching problems and traffic management problems.

Another important problem addressed in this work relates to the simplification of trajectories. Some trajectories like animal movement tend to be very chaotic. In order to obtain a general global patterns (directional) from these trajectories, we need to simplify them. We identify that these non-smooth behavior of the trajectories are mostly marked by the frequent sharp angular movements. We propose a trajectory smoothing technique where we identify these sharp angular turns and remove those from the original trajectories.

The present work is different from previous works on trajectory data analysis as discussed in the next several paragraphs.

The article [5] proposes a partition-and-group framework for clustering trajectory data. Using the concept of minimum descriptive length (MDL) the original trajectories are segmented. These segments are called trajectory partitions, which are then clustered using a modified version of DBSCAN algorithm, which clusters the line segments. Finally the clusters are represented by representative trajectories.

This work is different from our work because in this paper the trajectories are segmented into line segments and these line segments are clustered. In our work we consider the sub-trajectories which need not be line segments but can be polylines with a consistent direction. The criteria for segmentation in our work is directional consistency, whereas [5] uses MDL as the segmentation criteria.

In [6], a clustering algorithm is given for the trajectory data that uses a combination of techniques from data mining, computational geometry and string processing. In [9], the trajectory clustering technique of [5] has been extended for trajectory classification. A spatio-temporal pattern called convoy has been proposed in [11]. In this article the authors propose various efficient algorithms for convoy detection. In [7], a similar technique to [11] for mining spatio temporal pattern called flocking behavior is proposed. A flock refers to the set of the trajectories that remain close to each other for some reasonable time interval. In a flock pattern mining both the time as well as the spatial attributes are required. All these works are related to trajectories and trajectory clustering, but none of these methods consider the directional aspect of the trajectories as we have done in this work.

Trajectory smoothing has been addressed in [15–17]. In [15] the authors propose a trajectory smoothing method that preserves the direction information. However, our approach to trajectory smoothing is very different compared to these works. We do smoothing as a preprocessing step for the directional analysis. Therefore, we do smoothing to capture the global directional orientation while removing the local arbitrary sharp angular movements.

This thesis is organized as follows.

- **Chapter 2: Related Work and Background:** Describes the related work and the background.

- **Chapter 3: Extracting dense regions from hurricane data:** This work has been done on a hurricane dataset. Here we consider the problem of identifying the dense regions (hot spots) of hurricane activity. This work considers the trajectory data as point data and used DBSCAN algorithm. The analysis involves clustering using only spatial attributes (longitude and latitude), clustering with spatial as well as non spatial attribute (wind speed) and clustering involving spatial as well as temporal attribute (relative time). The clustering results under different combinations of parameters (spatial only, spatial and non spatial, and spatial and temporal) have been compared. The regions of the storms origin and landing have been also identified.
- **Chapter 4: Directional framework for trajectory data analysis:** In this work we analyzed trajectory datasets from the directional point of view. We believe that direction is one of the main attribute of a trajectory. We propose a framework for the directional analysis of trajectory dataset. In this framework 16 directions have been identified. On the basis of these directions, trajectories are segmented to retain their directional consistency. After the segmentation process, the outlier trajectories in each directional class are removed using a novel convex hull based filtration process. At the end we identify the directional patterns as the directional clusters. These directional clusters are identified using DBSCAN algorithm which works on sub-trajectories as the basis data.
- **Chapter 5: Trajectory Smoothing and Simplification:** In this chapter we propose two novel methods for smoothing trajectory data sets. We also highlight the significance of our approach and the weakness with the existing window based trajectory smoothing concepts.
- **Chapter 6: Conclusion:** This chapter will conclude the thesis.

CHAPTER 2

Related Work and Background

The abundance of spatio-temporal tracking data in applications like global positioning system (GPS), hurricane and storm tracking data and animal movement data have made their analysis very important. This analysis is vital in knowing and managing the traffic pattern of vehicles, monitoring and predicting weather conditions, examining wild animal behavior and movement as well as analyzing the spread of a disease. A number of attempts have been made in this domain to analyze these kinds of data sets. Some of these analysis could be found in [4–12]

2.1 Related Work

In [8] a non-parametric approach to spatial trajectory clustering, called DENTRAC (DENsity based TRAjectory Clustering) is proposed. DENTRAC is a trajectory based approach which uses the non-parametric density estimation technique. The post processing of the obtained spatial clusters is performed to get more domain specific knowledge.

The article [5] proposes a partition-and-group framework for clustering the trajectory data. Using the concept of minimum descriptive length (MDL) principle, the most important points on the trajectory, called characteristic points, are identified. The original trajectories are now represented by connecting the consecutive characteristic points. Each segments thus obtained are called trajectory partitions. These trajectory partitions are then clustered using a modified version of DBSCAN algo-

rithm, which clusters the line segments. Finally the clusters are represented by the representative trajectories.

In [6], a clustering algorithm is given for the trajectory data that uses the combination of techniques from data mining, computational geometry and string processing. The trajectories are preprocessed to remove noise after which they are segmented into sub-trajectories. These segments are then classified and accordingly labeled on their geometric properties e.g., "wide left right" or "short straight segments". The next phase of the algorithm finds the frequent occurring substrings; these are called the $motifs$. The algorithm then maps the sub trajectories corresponding to the motifs to some feature space. The next stage performs the density based clustering and the final stage does the post processing of the clusters.

In [10] a novel algorithm called Slicing-STS-Miner has been proposed for mining the sequential patterns from the spatio-temporal data. This analysis is very valuable for analyzing the evolution of phenomena in spatial and temporal domain.

A spatio-temporal pattern called convoy has been proposed in [11]. Convoy is defined as the collection of trajectories that move together for a significant time. Closeness of trajectories at different time is computed on the basis of the density connected property [13]. In this article the authors propose various efficient algorithms for convoy detection.

In [7], a similar technique to [11] for mining spatio-temporal pattern called flocking behavior is proposed. The flock refers to the set of the trajectories that remain close to each other for some reasonable time interval. Closeness between trajectories is based on the size of a disc through which trajectories should pass. Therefore, unlike convoys, flock patterns are always dependent on the disc size assumption. In the flock pattern mining both the time as well as the spatial attributes are required.

In the article [9], the trajectory clustering technique of [5] has been extended for trajectory classification. In this article two levels of clustering; namely, the region-based and trajectory based clustering is done. Clustering is used to find the discriminative features for classification. The first level of the clustering is region level which identifies the higher level, region based features of the trajectories. The second level of the clustering identifies the lower level movement based features. These two clustering collaboratively identify the high-quality features for the classification.

In [12] the authors propose a classification technique for the trajectory data which incorporates the duration of the trajectory as an important feature.

Trajectory simplification (smoothing) has been addressed in [15–17]. These methods can be classified as *1) position preserving and 2) direction preserving.* Simplification methods presented in [16] and [17] belong to position preserving category whereas, the method in [15] belongs to direction preserving category. In the position preserving simplification method trajectories are simplified considering a positional threshold error, whereas in direction preserving simplification methods the simplification threshold is an angle.

2.2 Background

2.2.1 Trajectory data

In this section, we give formal definitions for a trajectory and its major defining characteristics.

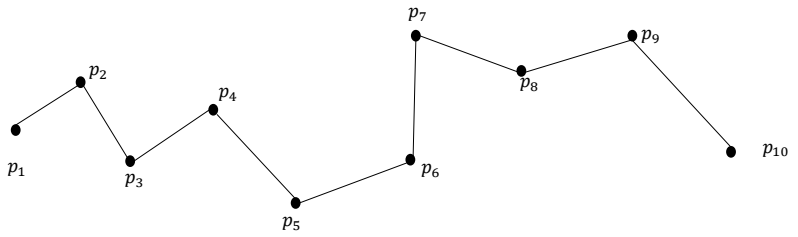


Figure 2.1: Trajectory example

2.2.1.1 Trajectory Definition

A trajectory Tr_i of size s is a sequence of points $[p_1, p_2, \ldots, p_s]$, where p_1 is the initial point, p_s the final point and p_i is the i^{th} sampled point in Tr_i. The trajectory in Figure 2.1, $[p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}]$ is of size $s = 10$.

A given trajectory can be characterized by the following attributes:

- *Size (s)*: of trajectory in Figure 2.1 is 10.
- *Trajectory segments (L_i)*: A trajectory Tr_i consists of line segments $L_i = \overline{p_i p_{i+1}}$ which are formed by joining the i^{th} and $i + 1^{th}$ consecutive points in it.
  - The line segments in a trajectory can be characterized by their length $len(L_i)$. This length will be the distance between its end points which are p_i and p_{i+1} .We take the *Euclidean distance* as the distance between the end points of the line segments, which becomes the length of the line segments forming a trajectory.

– If the size of the trajectory is s, then it will have $s-1$ line segments in it.

- The angular attribute of the trajectory is computed considering the angle between its L_i and L_{i+1} consecutive line segments. Which signifies that we need to consider three successive points which are p_i, p_{i+1} and p_{i+2}. More formally let us denote the angle between the two line segments $\overline{p_i p_{i+1}}$ and $\overline{p_{i+1} p_{i+2}}$ as $angle(\overline{p_i p_{i+1}}, \overline{p_{i+1} p_{i+2}})$ which is measured in anti-clockwise rotation from $\overline{p_i p_{i+1}}$ to $\overline{p_{i+1} p_{i+2}}$. Finally we consider the angle:

$$\theta = min(angle(\overline{p_i p_{i+1}}, \overline{p_{i+1} p_{i+2}}), 360 - angle(\overline{p_i p_{i+1}}, \overline{p_{i+1} p_{i+2}})) \tag{2.1}$$

  This formulation means that we consider the smaller of the angles between the two line segments considered here.

2.2.2 DBSCAN

DBSCAN is a density based clustering algorithm [13]. It has two important parameters called the Min_{Pts} and Min_{Dst}. These two parameters determine the density of the data to be clustered. For a point to be evaluated as dense, we need to look at a neighborhood of size Min_{Dst} centered around it. In this neighborhood there should be at least Min_{Pts} number of data points to make this particular data item dense. On the basis of the density of the data points in the data set, DBSCAN identifies three types of points viz., 1) *core points*, 2) *boundary points* and 3) *noise points*. Figure 3.1 gives a scenario of the data points and distinguishes between the three kinds of data points. Formally, these points are defined on the basis of $Min_{Pts}-neighborhood$, viz., $(N_{Min_{Pts}}(p))$ for a point p in the dataset D.

$$N_{Min_{Pts}}(p) = (a | a \in D \; and \; dist\,(p, a) \leq Min_{Dst}) \tag{2.2}$$

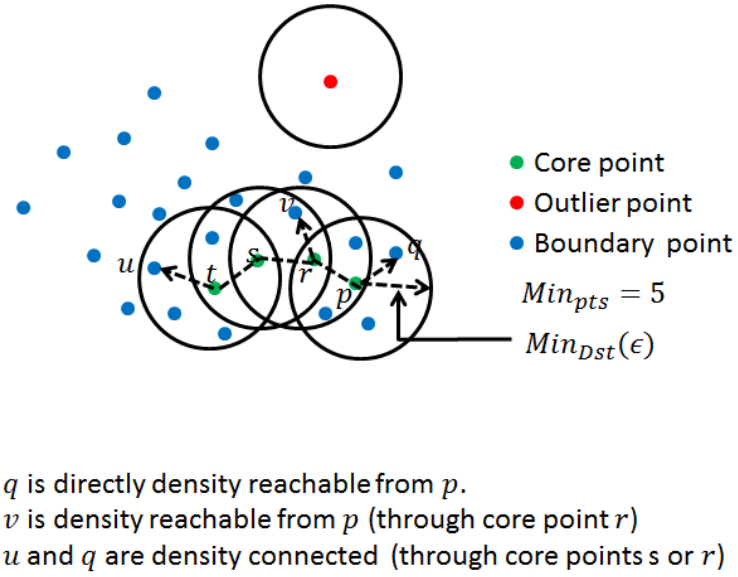


Figure 2.2: DBSCAN Types of data points: Core points, Boundary points and Outliers

For every point p in the dataset D, its $Min_{Pts}-neighborhood$, viz., $(N_{Min_{Pts}}(p))$ is determined on the basis of the parameter Min_{Dst} and similarity measure viz., $dist\,(p, a)$ (for example, Euclidean distance), between the point and its neighbors. If the size of $N_{Min_{Pts}}(p)$ for a particular point p, is not less than Min_{Pts} then the point is considered a core point. If the point p is not core but it lies in the $N_{Min_{Pts}}(q)$ of a core point q, then it is called a boundary point. If it is not a core point and also does not lie in the neighborhood of any core point, then it is called an outlier (see Figure 2.2).

To define the clusters in terms of DBSCAN, three more concepts have been defined, these are:

1) *directly density reachable*, 2) *density reachable* and 3) *density connected*. A point q will be directly density reachable only from a core point (p), only when it lies

Algorithm 1 The DBSCAN Algorithm

$DBSCAN(D, Min_{Dst}, Min_{Pts})$

$C = 0$

for all unvisited point P in dataset D **do**

mark P as visited

$Neighbor_{Pts} = regionQuery(P, Min_{Dst})$

if $sizeof(Neighbor_{Pts}) < Min_{Pts}$ **then**

mark P as $NOISE$

else

$C = nextcluster$

$expandCluster(P, Neighbor_{Pts}, C, Min_{Dst}, Min_{Pts})$

end if

end for

in the $N_{Min_{Pts}}(p)$. For example, point q is directly density reachable from the core point p in Figure 2.2. Similarly a point t would be density reachable from a core point p, if there is a sequence of data points $\{x_1, x_2, \ldots, x_n | x_i \in D\}$, where x_i is directly density reachable from x_{i-1}, and also $x_1 = p$, whereas $x_n = t$. For example point v is density reachable from core point p in Figure 2.2. Similarly the density connectivity between two points a and b in the data set is defined as the existence of a core point c such that the points a and b are density reachable from c. For example in Figure 2.2, points q and u are density connected with respect to the core point s (also r).

A cluster C is defined as the subset of objects satisfying two criteria: 1) Connected: means that $\forall p, q \in C$, p and q are density connected, 2) Maximal: It means that $\forall p, q$, if $p \in C$ and q is density reachable from p, then $q \in C$.

The structure of the DBSCAN is given in Algorithm 4, Algorithm 5 and Algorithm 6.

Algorithm 2 expandCluster

$expandCluster(P, Neighbor_{Pts}, C, Min_{Dst}, Min_{Pts})$

add P to cluster C

for all each point P' in $Neighbor_{Pts}$ **do**

if P' is not visited **then**

mark P' as *visited*

$Neighbor_{Pts'} = regionQuery(P', Min_{Dst})$

if $sizeof(Neighbor_{Pts'}) >= Min_{Pts}$ **then**

$Neighbor_{Pts} = Neighbor_{Pts}$ joined with $Neighbor_{Pts'}$

end if

end if

if P' is not yet member of any cluster **then**

add P' to cluster C

end if

end for

2.3 Summary

In this chapter, we discuss previous work, and formally defined the trajectory concepts. We also gave an overview of DBSCAN [13], which is used for spatial clustering in our work.

Algorithm 3 regionQuery

$regionQuery(P, Min_{Dst})$

return all points within $P's\ Min_{Dst} - neighborhood\ (including P)$

CHAPTER 3

Extracting Dense Regions From Hurricane Trajectory Data (Drafted from [1])

3.1 Introduction

The abundance of spatio-temporal tracking data in applications like global positioning system (GPS), hurricane and storm tracking data and animal movement data have made their analysis very important. This analysis is vital in knowing and managing the traffic pattern of vehicles, monitoring and predicting weather conditions, examining wild animal behavior and movement as well as analyzing the spread of a disease. A number of attempts have been made in this domain to analyze these kinds of data sets. Some of these analysis could be found in [4–12]

For our task of identifying the dense regions of hurricane activity, we use a clustering algorithm called DBSCAN [13]. Clustering is a very useful task in data mining, which groups similar objects (physical or abstract) together [18]. The weather trajectory data has been analyzed using clustering algorithms in [5] and [6] (see Section 2).

Since the hurricane data is in the form of a trajectory, that represents the spatial locations of hurricane at different time instances, DBSCAN has been used which is a spatial clustering algorithm. For the current analysis we consider the hurricane data as point data, unlike the approach in [5] and [6]. This approach has been motivated by the need of the analysis and also by the fact that the hurricane trajectory lengths are different.

We cluster the data to obtain dense regions that effectively identify the hot spots for the storm activity. These dense regions have been identified considering

different combinations of parameters. Initially we do only the spatial dense regions identification considering latitude and longitude, then we incorporate wind speed also as an additional attribute. This has been done to evaluate the impact of the non spatial attribute on the dense regions identification. Finally we do clustering considering the spatial as well as the temporal attribute to identify the spatio-temporal dense regions. For this analysis we consider the relative time framework as we are interested in the storm progression. We normalize the temporal value in the range of $[0-1]$, to handle the different length hurricanes. Our framework for combining the spatial and non-spatial attributes is inspired by the approach in [4].

We identified some dense regions that would be useful for the domain experts. First the locations from where the storms are most likely to originate, second the locations where the storms are most likely to land and finally, the regions that have been mostly affected by the storm activity. For the storm staring location identification, we cluster the initial portion of the hurricane, whereas, for the potential storm landing locations we cluster the last portion of the hurricane trajectory. The clustering considering the whole hurricane length data will find out the dense regions of high storm activity.

We have used the hurricane (Atlantic region) data set for 50 years from 1950 to 1999. The data set was obtained from [19]. The data has six attributes, which are latitude, longitude, time, wind speed, pressure, and status. The data has been sampled in the interval of 6 hours. The dataset has 15319 data points and 496 trajectories.

3.2 Related work

In this section, we give a brief overview of eight related articles [5–12].

In [8] a non-parametric approach to spatial trajectory clustering, called DENTRAC (DENsity based TRAjectory Clustering) is proposed. DENTRAC uses the non-parametric density estimation technique. The post processing of the obtained spatial clusters is performed to get more domain specific knowledge.

The article [5] proposes a partition-and-group framework for clustering the trajectory data. Using the concept of minimum descriptive length (MDL) principle, most important points on the trajectory called characteristic points are identified. The original trajectories are now represented by connecting the consecutive characteristic points. Each segments thus obtained are called trajectory partitions. These trajectory partitions are then clustered using a modified version of DBSCAN algorithm, which clusters the line segments. Finally the clusters are represented by the representative trajectories.

In [6], a clustering algorithm is given for the trajectory data that uses the combination of techniques from data mining, computational geometry and string processing. The trajectories are preprocessed to remove noise after which they are segmented into sub-trajectories. These segments are then classified and accordingly labeled on their geometric properties e.g., "wide left right" or "short straight segments". The next phase of the algorithm finds the frequent occurring substrings; these are called the $motifs$. Algorithm then maps the sub trajectories corresponding to the motifs to some feature space. The next stage performs the density based clustering and the final stage does the post processing of the clusters.

In [10] a novel algorithm called Slicing-STS-Miner has been proposed for mining the sequential patterns from the spatio temporal data. This analysis is very valuable for analyzing the evolution of phenomena in spatial and temporal domain.

A spatio-temporal pattern called convoy has been proposed in [11]. In this article authors propose various efficient algorithms for the convoy detection.

In the article [9], the trajectory clustering technique of [5] has been extended for trajectory classification. In this article two levels of clustering; namely, the region-based and trajectory based clustering is done. Clustering is used to find the discriminative features for classification. The first level of the clustering is region level which identifies the higher level, region based features of the trajectories. The second level of the clustering identifies the lower level movement based features. These two clustering collaboratively identify the high-quality features for the classification.

In [12] authors propose a classification technique for the trajectory data which incorporates the duration of the trajectory as an important feature.

In [7], a similar technique to [11] for mining spatio temporal pattern called the flocking behavior is proposed. The flock refers to the set of the trajectories that remain close to each other for some reasonable time interval. In the flock pattern mining both the time as well as the spatial attributes are required.

3.3 Trajectory Clustering on Point Data

In our approach we have considered the trajectory data set as consisting of just the individual data points in the trajectories. In this analysis we are not constraining these points to belong to their respective parent trajectory or sub trajectory by enforcing them to belong to a line segment or a sequence of line segments as has been done in [5] and [6]. Since the core algorithm behind our analysis is DBSCAN [13] we review that algorithm first.

3.3.1 DBSCAN

DBSCAN is a density based clustering algorithm. It has two important parameters called the Min_{Pts} and Min_{Dst}. These two parameters determine the density of the data to be clustered. For a point to be evaluated as dense, we need to look at a

neighborhood of size Min_{Dst} centered around it. In this neighborhood there should be at least Min_{Pts} number of data points to make this particular data item dense. On the basis of the density of the data points in the data set, DBSCAN identifies three types of points viz., 1) *core points*, 2) *boundary points* and 3) *noise points*. Figure 3.1 gives a scenario of the data points and distinguishes between the three kinds of data points. Formally, these points are defined on the basis of $Min_{Pts}-neighborhood$, viz., $(N_{Min_{Pts}}(p))$ for a point p in the dataset D.

$$N_{Min_{Pts}}(p) = (a|a \in D \; and \; dist\,(p,a) \leq Min_{Dst}) \tag{3.1}$$

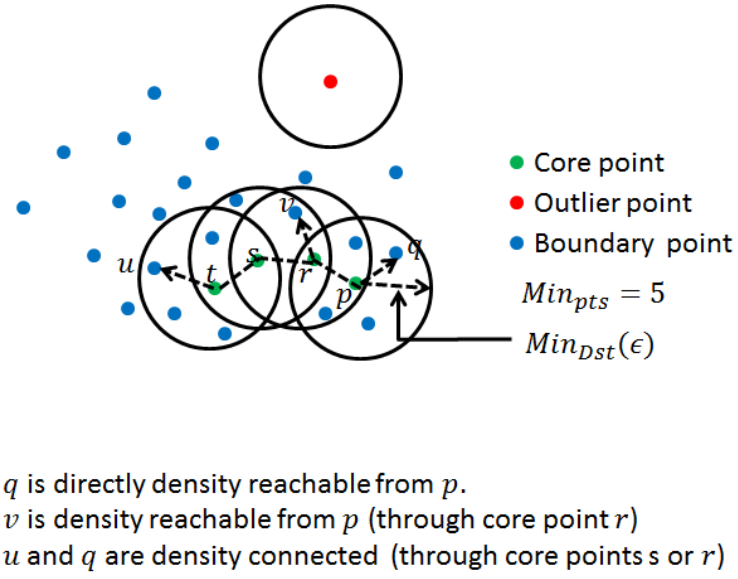


Figure 3.1: DBSCAN Types of data points: Core points, Boundary points and Outliers

For every point p in the dataset D, its $Min_{Pts}-neighborhood$, viz., $(N_{Min_{Pts}}(p))$ is determined on the basis of the parameter Min_{Dst} and similarity measure viz.,

Algorithm 4 The DBSCAN Algorithm

$DBSCAN(D, Min_{Dst}, Min_{Pts})$

$C = 0$

for all unvisited point P in dataset D **do**

mark P as visited

$Neighbor_{Pts} = regionQuery(P, Min_{Dst})$

if $sizeof(Neighbor_{Pts}) < Min_{Pts}$ **then**

mark P as $NOISE$

else

$C = nextcluster$

$expandCluster(P, Neighbor_{Pts}, C, Min_{Dst}, Min_{Pts})$

end if

end for

$dist\,(p, a)$ (for example, Euclidean distance), between the point and its neighbors. If the size of $N_{Min_{Pts}}(p)$ for a particular point p, is not less than Min_{Pts} then the point is considered a core point. If the point p is not core but it lies in the $N_{Min_{Pts}}(q)$ of a core point q, then it is called a boundary point. If it is not a core point and also does not lie in the neighborhood of any core point, then it is called an outlier (see Figure 3.1).

To define the clusters in terms of DBSCAN, three more concepts have been defined, these are:

1) *directly density reachable*, 2) *density reachable* and 3) *density connected*. A point q will be directly density reachable only from a core point (p), only when it lies in the $N_{Min_{Pts}}(p)$. For example, point q is directly density reachable from the core point p in Figure 3.1. Similarly a point t would be density reachable from a core point

p, if there is a sequence of data points $\{x_1, x_2, \ldots, x_n | x_i \in D\}$, where x_i is directly density reachable from x_{i-1}, and also $x_1 = p$, whereas $x_n = t$. For example point v is density reachable from core point p in Figure 3.1. Similarly the density connectivity between two points a and b in the data set is defined as the existence of a core point c such that the points a and b are density reachable from c. For example in Figure 3.1, points q and u are density connected with respect to the core point s (also r).

A cluster C is defined as the subset of objects satisfying two criteria: 1) Connected: means that $\forall p, q \in C$, p and q are density connected, 2) Maximal: It means that $\forall p, q$, if $p \in C$ and q is density reachable from p, then $q \in C$.
The structure of the DBSCAN is given in Algorithm 4, Algorithm 5 and Algorithm 6.

3.3.2 Dense region extraction

Our contribution in this article is in analysis of the hurricane data to find the regions of high storm activity. We used DBSCAN algorithm which uses the parameter Min_{Dst} for finding the neighborhood and hence the density of the data points. Since the hurricane data set which has been used in this paper has spatial as well as non-spatial attributes, first we find the neighborhood considering only the spatial neighborhood, then we considered a non-spatial attribute (wind speed) also.

3.3.2.1 Spatial Clustering

For the spatial clustering we considered the latitude and the longitude values of the data points. We considered the $Haversine\ formula$ for computing the distance between the two data points represented by their latitude and longitude values, i.e., $P_i = (\phi_i, \lambda_i)$, where ϕ_i is the latitude and λ_i is the longitude of the data point P_i. If

Algorithm 5 expandCluster

$expandCluster(P, Neighbor_{Pts}, C, Min_{Dst}, Min_{Pts})$

add P to cluster C

for all each point P' in $Neighbor_{Pts}$ **do**

if P' is not visited **then**

mark P' as *visited*

$Neighbor_{Pts'} = regionQuery(P', Min_{Dst})$

if $sizeof(Neighbor_{Pts'}) >= Min_{Pts}$ **then**

$Neighbor_{Pts} = Neighbor_{Pts}$ joined with $Neighbor_{Pts'}$

end if

end if

if P' is not yet member of any cluster **then**

add P' to cluster C

end if

end for

---

we have two points $P_1 = (\phi_1, \lambda_1)$ and $P_2 = (\phi_2, \lambda_2)$, the $Haversine\ formula$ is given as:

$$a = \sin(\Delta/2) + \cos(\phi_1) \times \cos(\phi_2) \times \sin^2(\Delta\lambda/2)$$

$$c = 2 \times \arctan 2(\sqrt{a}, \sqrt{(1-a)})$$

$$d = R \times c$$

Algorithm 6 regionQuery

$regionQuery(P, Min_{Dst})$

return all points within $P's\ Min_{Dst} - neighborhood\ (includingP)$

where $\Delta(\phi) = \phi_2 - \phi_1$ and $\Delta(\lambda) = \lambda_2 - \lambda_1$, $R = 6371Km$ is the radius of the Earth, and d is the Haversine distance. Since the spatial co-ordinates in geographical data sets are reported in terms of the latitude and longitude, it will be natural to adopt some distance measure that computes the circular distance between two points lying on a spherical object (earth).

3.3.2.2 Spatial and non spatial clustering

We extend the DBSCAN algorithm to handle the non spatial attributes also for clustering. Our approach to this analysis is inspired by [4], where authors extended the DBSCAN algorithm to incorporate non-spatial attributes. Let there be a data set $D = \{d_1, d_2, \ldots, d_n\}$, where $d_i = (x_i, y_i, a_i, b_i)$. Let (x_i, y_i) be the spatial attributes and (a_i, b_i) the non spatial attributes. According to [4], we can consider the spatial as well as the non spatial distances to find out the respective neighborhoods. More formally, let us consider the distance between two data points $d_1 = (x_1, y_1, a_1, b_1)$ and $d_2 = (x_2, y_2, a_2, b_2)$. Now if we denote the non-spatial distance as $dist_s$ between d_1 and d_2, then $dist_s(d_1, d_2) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$. Similarly, we can find the non-spatial distance between the same data points d_1 and d_2 as $dist_{ns}(d_1, d_2) = \sqrt{(a_1 - a_2)^2 + (b_1 - b_2)^2}$. Here we assume that the Euclidean distance is meaningful between the data points in spatial as well as non spatial domain. Let us define two user specified spatial and non-spatial threshold as ϵ_s and ϵ_{ns}, respectively. Now we can define two neighborhoods of a point d_k, which are the spatial neighborhood and the non-spatial neighborhood. The spatial neighborhood is $N_{\epsilon_s}(k) = \{d_j \in D | dist_s(d_k, d_j) \leq \epsilon_s\}$, and the non-spatial neighborhood is $N_{\epsilon_{ns}}(k) = \{d_j \in D | dist_s(d_k, d_j) \leq \epsilon_{ns}\}$. Finally the composite neighborhood will include the data points common in $N_{\epsilon_s}(k)$ and $N_{\epsilon_{ns}}(k)$ which is $N(k) = N_{\epsilon_s}(k) \cap N_{\epsilon_{ns}}(k)$. Effectively, the neighborhood $N(k)$ consists of the data points around d_k, which are

closer to it with respect to spatial as well as non-spatial distance, given the respective thresholds ϵ_s and ϵ_{ns}.

For the non-spatial attribute we have used wind speed in our analysis, because wind speed is an important attribute of hurricanes. Further, only the wind speed data is available in the data set for all the records.

3.3.2.3 Spatio-temporal clustering

Since the hurricane data is naturally spatio-temporal data, we need to consider the time also in the analysis. In the current work we have considered time also as a non-spatial attribute. We use the time in the following formulation. Let the user specified temporal threshold be ϵ_t. Here the data point $d_i = (\phi_i, \lambda_i, t_i)$, where ϕ_i is the latitude, λ_i is the longitude, and t_i is the time. $dist_s$ is the $Haversine$ distance, where as the $dist_{ns}$ is simply the Manhattan distance, viz., $dist_{ns}(d_i, d_j) = |t_i - t_j|$.

In the hurricane data set, hurricanes have been sampled at 6 hours intervals, so a trajectory of a particular hurricane of length l is represented as $Tr_i = \{(\phi_{i1}, \lambda_{i1}, 1), (\phi_{i2}, \lambda_{i2}, 2)$
$\ldots (\phi_{il}, \lambda_{il}, l)\}$. In this work we have considered the relative time framework, where instead of using the absolute time in the hurricane tracks, we consider the relative time from the start of a particular track. This framework is more important in the current analysis because our aim is to analyze this data from the hurricane's movement patterns point of view. Since the hurricanes are of different length, we have normalized the time component by the length of the hurricane trajectory. Which is:

$$Trn_i = \left\{ \left(\phi_{i1}, \lambda_{i1}, \frac{0}{l-1} \right), \left(\phi_{i2}, \lambda_{i2}, \frac{1}{l-1} \right) \ldots (\phi_{il}, \lambda_{il}, 1) \right\}$$

Because of this normalization the time component of the trajectory will vary from $(0 - 1.0)$. Please note that the lower value of t_i close to 0.00 will signify that the trajectory data point is in its initial stage, where as the higher values close to 1.00

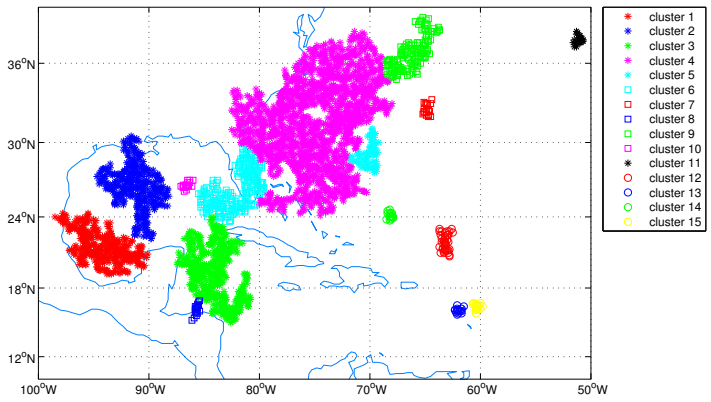


Figure 3.2: DBSCAN spatial clusters: $Min_{Pts} = 10$, $Min_{Dst} = 35$

would signify that the particular data point lies towards the end of the hurricane. Similarly the values of t_i close to 0.5 will signify the middle data points in the trajectory. If we use the spatial and this temporal information together in the DBSCAN for finding the clusters then we will obtain spatio-temporal clusters.

3.4 Experiments and Analysis

We have done the implementation and analysis using MATLAB on the hurricane data set described at the end of Section 1.

3.4.1 Analyzing spatial attributes

Figure 3.2 shows the result of the DBSCAN algorithm applied on the 50 years storm data. For this experiment the parameter values were $Min_{Dst} = 35$ and $Min_{Pts} = 10$. We obtained 15 clusters of different sizes across the region. This choice of the parameters Min_{Dst} and Min_{Pts} is based on the experimentation. We tried to get the values of these parameters on the basis of the suggestions in [13], but

we got just one single cluster as the output. Therefore, we ran the DBSCAN algorithm for different combinations of Min_{Dst} and Min_{Pts} and picked the above value for the case where we got well separated and compact clusters.

In order to analyze the results depicted in Figure 3.2 we will utilize Table 3.1, which summarizes certain properties of the 15 clusters. The big cluster in the center (Magenta stars) shows the most dominant region of the storms. Numerical values corresponding to this cluster, which has cluster id 4, can be obtained from Table 3.1. This cluster with cluster id 4 is ranked first with respect to the number of trajectories($Storm_{rank}$ (#traject.)) as well as the number of data points belonging to it viz., ($Storm_{rank}$ (#data points)). The number of different storm trajectories that pass through this cluster is 185 and it has total of 1673 storm data points (Table 3.1). We conclude that this region is the most prominent region in terms of hurricane storm activity. The next dominant region corresponds to the cluster with cluster id 3 (green stars), which has the second rank in terms of the number of storm trajectories (viz., 80) (Table 3.1) that passed through it and the number of storm data points (viz., 471) in it. On the other side, we have some smaller size clusters that are lowest in the ranks. Clusters with id 10 and 14 rank as the last two in terms of the number of storms that pass through them, where as the storms with id 12 and 13 rank as the lowest two in terms of the ranking on the basis of the storms data points that belong to them. We have some medium class storm clusters also. This kind of analysis will identify spatially the regions that are susceptible to storms, and at the same time we can identify relative susceptibility among them.

Figure 3.3, Figure 3.2 and Figure 3.4 show the results of the spatial clustering using DBSCAN for different combination of parameters Min_{Pts} and Min_{Dst}. Figure 3.3 has values for $Min_{Pts} = 8$ and $Min_{Dst} = 30$. Since the value for Min_{Pts} is 8 which is a smaller value we got 32 compact and small clusters. This result has

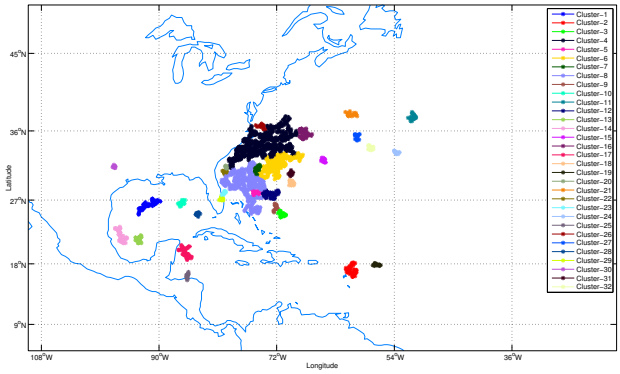


Figure 3.3: DBSCAN spatial clusters: $Min_{Pts} = 8$, $Min_{Dst} = 30$

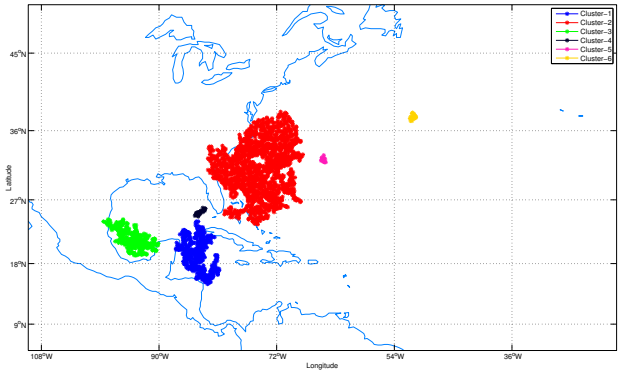


Figure 3.4: DBSCAN spatial and wind clusters: $Min_{Pts} = 12$, $Min_{Dst} = 35$

been obtained because of Min_{Dst} parameter also, which is 30 and is the smallest value in the results reported in this work. In Figure 3.2 we picked a larger value of $Min_{Pts} = 10$ and $Min_{Dst} = 35$, which resulted in comparatively smaller number of clusters which is, 14 and these clusters are comparatively bigger compared to those in Figure 3.3.

In Figure 3.4 we increase the value of Min_{Pts} to 12 while retaining the value of Min_{Dst} as 35. This resulted in 6 clusters. Three of these clusters are bigger while the other three are smaller. This is the result of the higher value of Min_{Dst} which imposes stronger density constraint (number of neighboring data points in Min_{Dst}

Table 3.1: Storm clustering analysis, on Spatial clustering , $Min_{Pts} = 10$, $Min_{Dst} = 35$

Storm ID	$Storm_{rank}$ (#traject.)	$Storm_{rank}$ (#data points)	$\#traject.$ $(Cluster_i)$	$\#DataPts.$ $(Cluster_i)$	$Color(Symb.)$
1	6	4	53	361	Red (star)
2	3	3	68	371	Blue (star)
3	2	2	80	471	Green (star)
4	1	1	185	1673	Magenta (star)
5	7	7	37	76	Cyan (star)
6	5	5	58	244	Cyan (square)
7	10	9	16	24	Red (square)
8	4	6	60	144	Blue (square)
9	9	11	18	20	Green (square)
10	14	10	9	22	Magenta (square)
11	8	8	27	40	Black (star)
12	13	14	12	15	Red (circle)
13	11	15	14	15	Blue (circle)
14	15	12	9	20	Green (circle)
15	12	13	14	17	Yellow (circle)

neighborhood) for a data point to be considered dense and hence included in the cluster.

3.4.2 Qualitative analysis of clusters

In this work we have extended DBSCAN algorithm to find out the dense regions in hurricane point data while considering the non-spatial attributes along with the natural spatial attribute. The two non-spatial attributes considered in this work are wind speed and time. We varied the wind speed threshold $Min_{DstWspeed}$ from $20-100$ (interval of 10), and the temporal threshold ϵ_t from $0.2 - 1.0$ in the interval of 0.1. Since we got a range of clustering results when we changed these parameters, we

Table 3.2: Storm Clustering analysis, Impact of Non spatial attribute $Min_{Pts} = 10$, $Min_{Dst} = 35$

$Min_{DstWspeed}$	$Mean(Std(Cluster\_i))$	(#clusters)
20	13.5669	5
30	16.222	6
40	22.4366	9
50	23.2284	9
60	25.458	12
70	25.9649	14
80	26.3079	14
90	27.6229	15
100	27.6691	15

needed a quality measure for the results of clustering. We used the quality measure proposed in [5]. This measure is given below:

$$\sum_{i=1}^{num_{clus}} \left(\frac{1}{2C_i} \sum_{x \in C_i} \sum_{y \in C_i} dist(x,y)^2 \right) + \frac{1}{2|N|} \sum_{w \in N} \sum_{z \in N} dist(w,z)^2 \tag{3.2}$$

where, num_{clus} is the number of clusters, N is the set of noise points and C_i is the i^{th} cluster. This quality measure computes the sum of the square error (SSE), which means the smaller this value, the better will be the clustering result.

3.4.2.1 Analyzing combination of spatial and non-spatial attributes

We did some additional analysis that would give us more concrete information in terms of the nature of the storms. The data set includes the attribute value wind speed, which is available for all the data points, and is one of the key characteristics of hurricanes. We used this as a non-spatial attribute and redefined the distance parameter in the DBSCAN algorithm to combine spatial and non spatial values. This approach is inspired by the work in article [4]. Now the clusters that we get are the spatial regions that are prone to similar kind of storms in terms of the wind speed. In the redefinition of the Min_{Pts}, we need to consider the neighbors as the

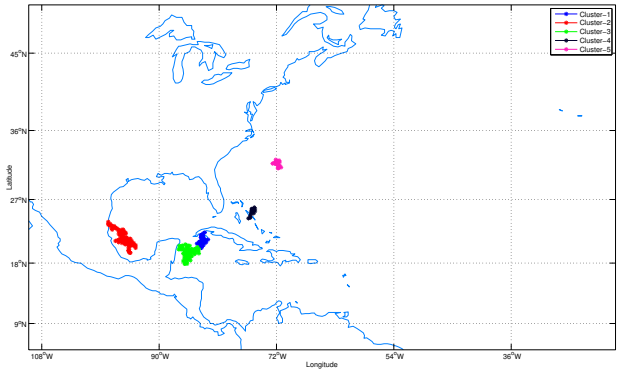


Figure 3.5: DBSCAN spatial and wind clusters: $Min_{Pts} = 10$, $Min_{Dst} = 35$, $Min_{DstWspeed} = 20$

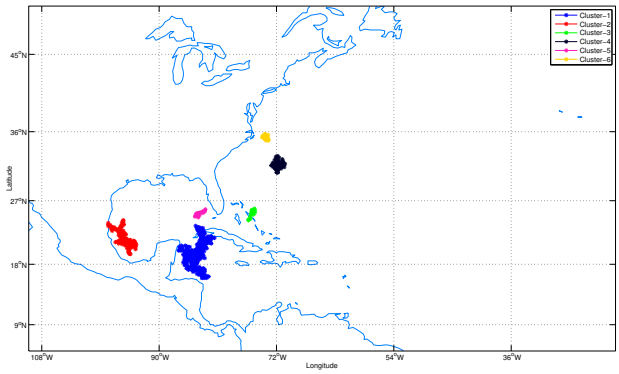


Figure 3.6: DBSCAN spatial and wind clusters: $Min_{Pts} = 10$, $Min_{Dst} = 35$, $Min_{DstWspeed} = 30$

data points which are closer to the particular data point with respect to the spatial distance as well as similar in wind speed values.

As expected when we constrained the neighborhood criteria by incorporating non-spatial attribute, viz., wind speed the number of clusters as well as the size of the clusters was reduced. This trend can be seen from Figure 3.5 to Figure 3.13. This is because now the clusters represent the regions that were influenced by the same nature of storms. When we relaxed the similarity in the wind speed to be 100 the result degenerated to the case of totally spatial clustering, as the wind speed similarity

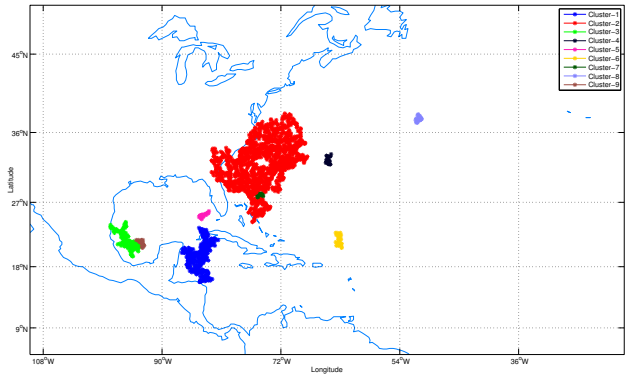


Figure 3.7: DBSCAN spatial and wind clusters: $Min_{Pts} = 10$, $Min_{Dst} = 35$, $Min_{DstWspeed} = 40$

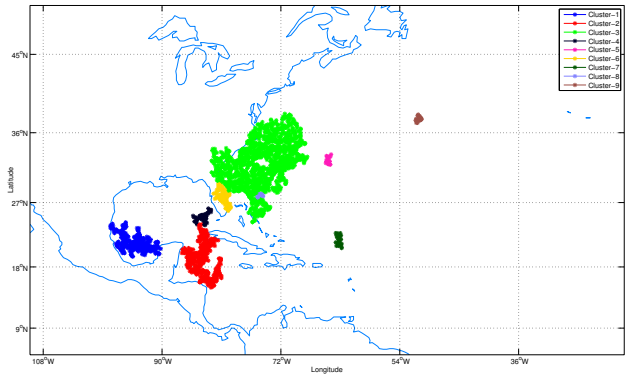


Figure 3.8: DBSCAN spatial and wind clusters: $Min_{Pts} = 10$, $Min_{Dst} = 35$, $Min_{DstWspeed} = 50$

had no impact. But as we reduced the value of wind speed similarity to lower values which are 70 (Figure 3.10), 50 (Figure 3.8) and 30 (Figure 3.6), respectively, we got smaller number of clusters and they got more compact. In order to reflect that extent of compactness we have a column in Table 3.2 $Mean(Std(Cluster\_i))$, viz., the mean of the standard deviation of the wind speed in the individual clusters for the particular choice of the wind speed similarity threshold. The standard deviation goes down as

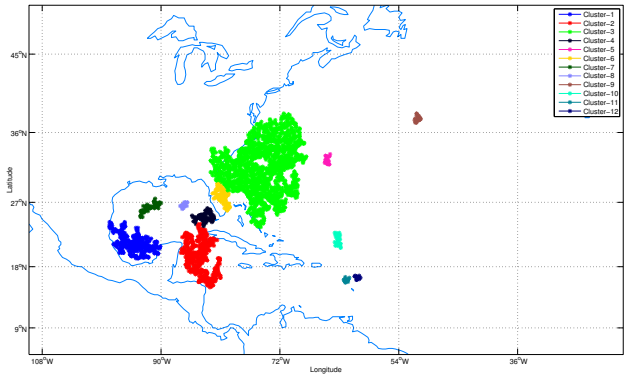


Figure 3.9: DBSCAN spatial and wind clusters: $Min_{Pts} = 10$, $Min_{Dst} = 35$, $Min_{DstWspeed} = 60$

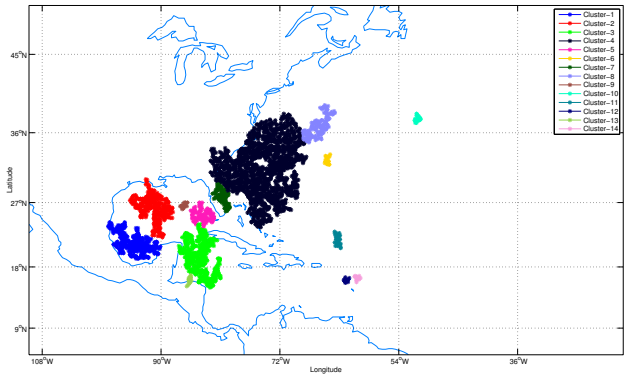


Figure 3.10: DBSCAN spatial and wind clusters: $Min_{Pts} = 10$, $Min_{Dst} = 35$, $Min_{DstWspeed} = 70$

we reduce the value of wind similarity. This measure gives an intuitive measure of the compactness and the homogeneous nature of the clusters.

Now using the measure in Equation 4.2, since it has the notion of distance, we used two distances separately. First we used the spatial distance using the latitude and longitude of the data points, and second the difference in wind speed of two data points. We got the following performance in Table 3.3. The results shows that as we reduce the $Min_{DstWspeed}$ value from 100 to 20 the clustering result improved in terms

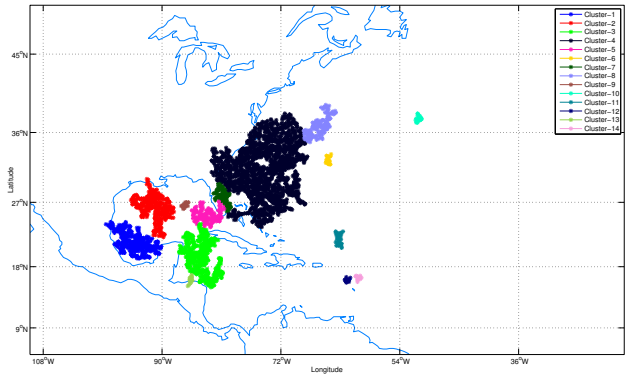


Figure 3.11: DBSCAN spatial and wind clusters: $Min_{Pts} = 10$, $Min_{Dst} = 35$, $Min_{DstWspeed} = 80$

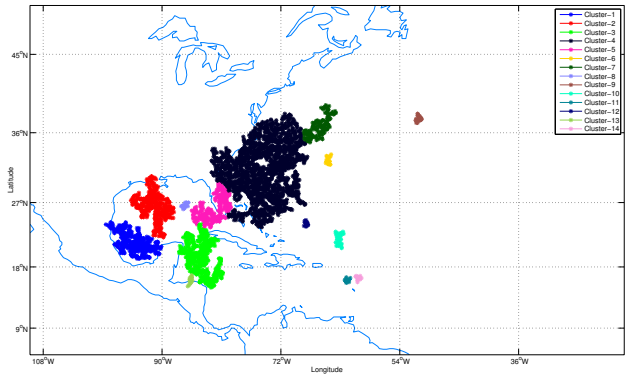


Figure 3.12: DBSCAN spatial and wind clusters: $Min_{Pts} = 10$, $Min_{Dst} = 35$, $Min_{DstWspeed} = 90$

of Q_{Wspeed} as well as $Q_{spatial}$. This result is obvious as the reduction in the threshold value $Min_{DstWspeed}$ forces more compact and homogeneous clusters.

Table 3.4 shows the spatio temporal clustering result. Here we have done the clustering using the normalized time parameter along with the spatial attributes. We reduced the temporal threshold ϵ_t from 1.0 (Figure 3.20) to 0.2 (Figure 3.14) and found the performance of the clustering results to improve in terms of the quality

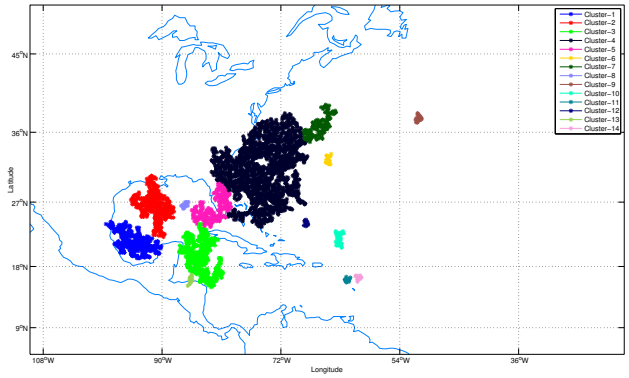


Figure 3.13: DBSCAN spatial and wind clusters: $Min_{Pts} = 10$, $Min_{Dst} = 35$, $Min_{DstWspeed} = 100$

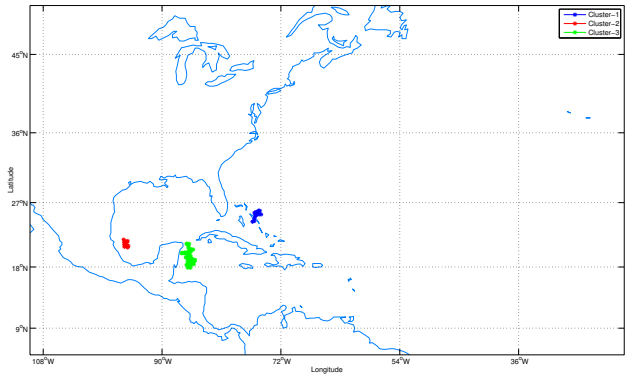


Figure 3.14: DBSCAN spatio-temporal clusters: $k = 10$, $Min_{Dst} = 35$, $\epsilon_t = 0.2$

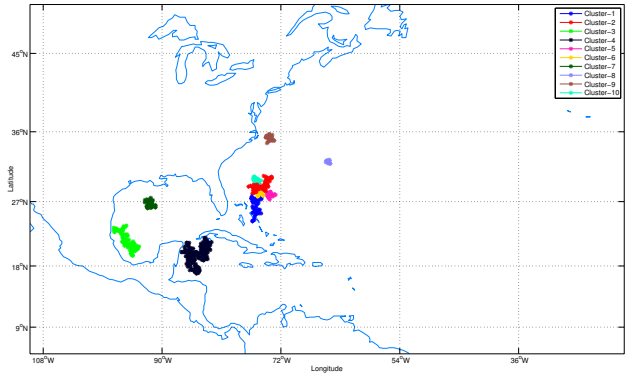


Figure 3.15: DBSCAN spatio-temporal clusters: $k = 10$, $Min_{Dst} = 35$, $\epsilon_t = 0.3$

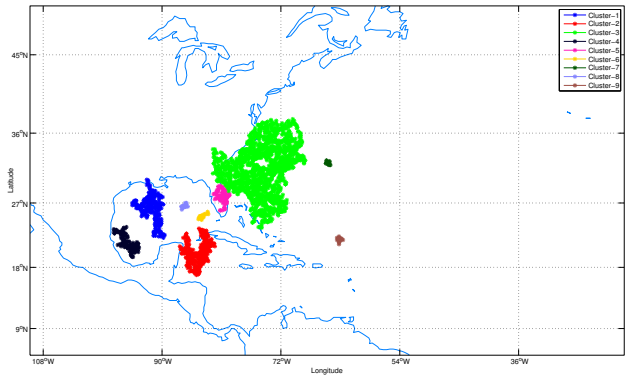


Figure 3.16: DBSCAN spatio-temporal clusters: $k = 10$, $Min_{Dst} = 35$, $\epsilon_t = 0.4$

Table 3.3: Qualitative measure of clustering results

$Min_{DstWspeed}(mph)$	$Q_{Wspeed}(mph)$	$Q_{spatial}(km)$
20	4.92E+04	3.35E+04
30	1.49E+05	8.62E+04
40	1.15E+06	2.56E+05
50	1.43E+06	2.96E+05
60	1.83E+06	3.46E+05
70	2.20E+06	4.11E+05
80	2.41E+06	4.73E+05
90	2.53E+06	5.18E+05
100	2.55E+06	5.18E+05

Table 3.4: Qualitative measure of clustering results

ϵ_t	Q_{ϵ_t}	$Q_{spatial}$
0.2	1.81E+00	1.13E+04
0.3	2.23E+01	9.30E+04
0.4	1.23E+02	3.04E+05
0.5	1.54E+02	3.54E+05
0.6	1.79E+02	4.11E+05
0.7	1.96E+02	5.05E+05
0.8	2.02E+02	5.18E+05
0.9	2.03E+02	5.19E+05
1.0	2.03E+06	5.19E+05

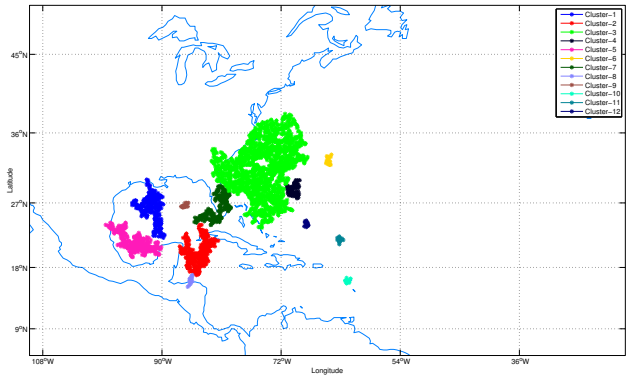


Figure 3.17: DBSCAN spatio-temporal clusters: $k = 10$, $Min_{Dst} = 35$, $\epsilon_t = 0.5$

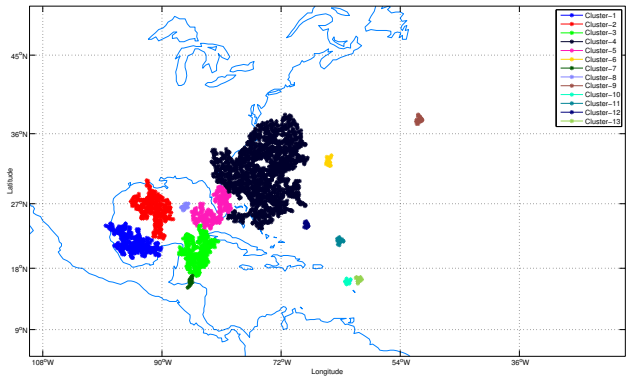


Figure 3.18: DBSCAN spatio-temporal clusters: $k = 10$, $Min_{Dst} = 35$, $\epsilon_t = 0.6$

Q_{ϵ_t} and $Q_{spatial}$. These two parameters denote the quality measure in terms of the temporal and spatial homogeneity of the clusters.

Note that although the lower values of the quality measure in Equation 4.2 signify better clustering result, these best performance results may not be what the end user wants. This is because the clustering concept is very subjective.

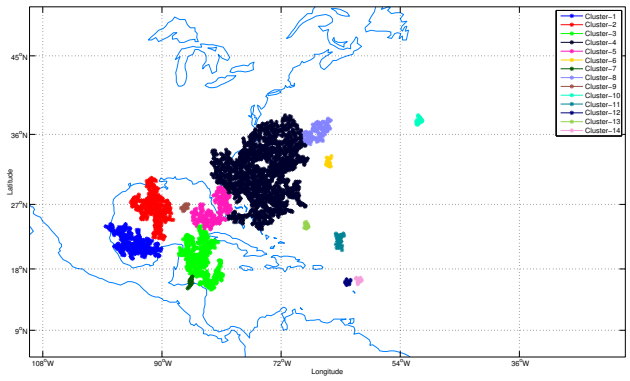


Figure 3.19: DBSCAN spatio-temporal clusters: $k = 10$, $Min_{Dst} = 35$, $\epsilon_t = 0.7$

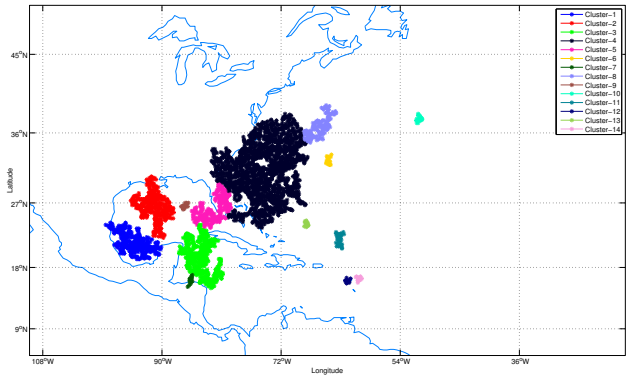


Figure 3.20: DBSCAN spatio-temporal clusters: $k = 10$, $Min_{Dst} = 35$, $\epsilon_t = 1.0$

3.4.3 Analyzing storm starting and landing information

One more analysis that we have done on the storm data is about the landing and the starting information for the storms. Here we first considered only the starting three data points corresponding to the first three time stamps, for all the storm trajectories. The DBSCAN algorithm was run on this data set. The resulting clusters give the regions from where the storms are most likely to start. Figure 3.21 shows the potential regions from where the storms may originate.

Similarly we did the analysis on the last three data point of the storm trajectories. The result in Figure 3.22 shows the potential regions, where the storms may end. Interestingly while comparing our findings to the ground truth in Figure 3.23, we see that our result captures most of the activity reported.

Our motivation for doing the separate analysis for the landing and starting trajectory portion is based on the fact that when we consider all of the data points for the clustering using DBSCAN, these (landing and starting) information related with the storm activity may be lost as noise.

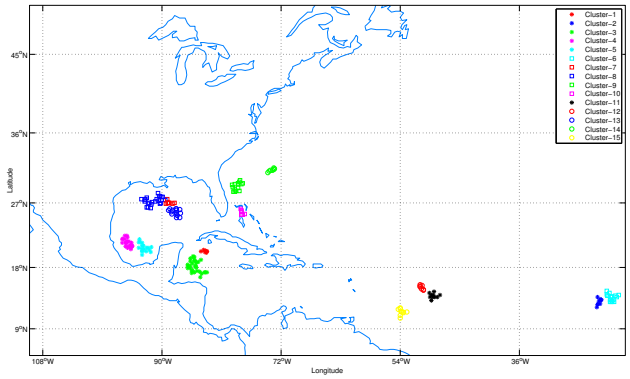


Figure 3.21: Storm starting clusters

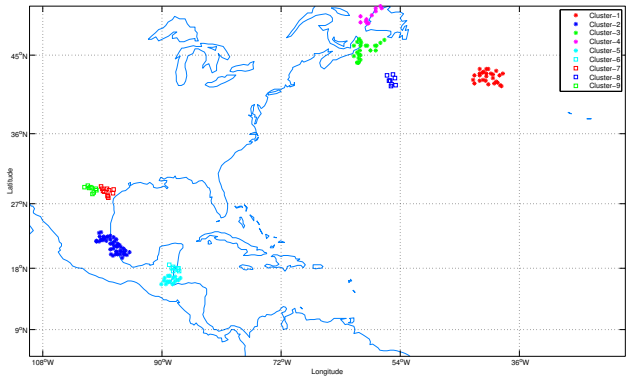


Figure 3.22: Storm Landing clusters

3.4.4 Analyzing storms for 10 year's duration

Here we discuss the results of clustering analysis on the hurricane data in the interval of 10 days. Since we have 50 years data so we split the data set in the interval of 10 years each and performed the clustering analysis. This is motivated by that we will get an idea about the trends in the storm over a fixed period of time. This analysis could also highlight an interesting fact about the hurricane tracking technology. It means that if we find an increasing number of hurricane regions or activity then it may be a result of more advanced hurricane tracking technology. The results of this analysis can be seen in Figure 3.24 to Figure 3.28.

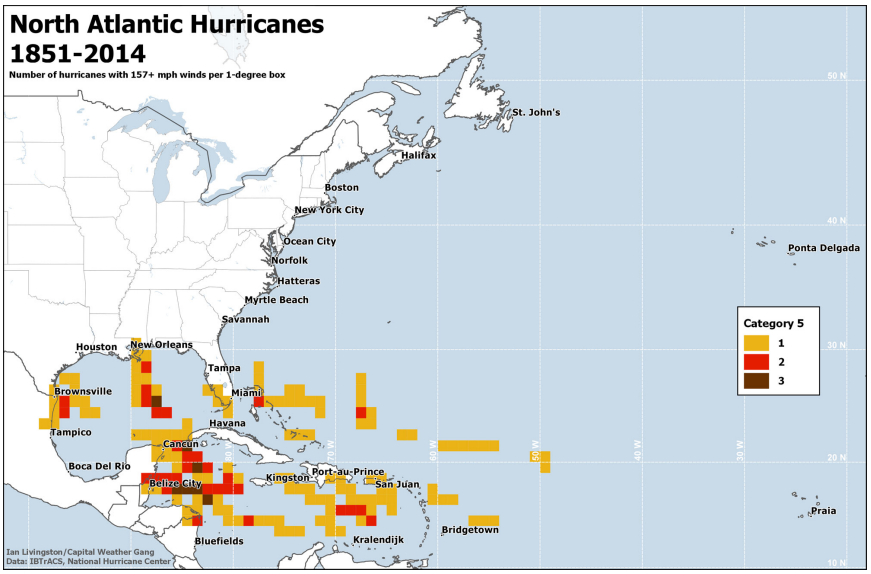


Figure 3.23: Storm activity ground truth

Since there is no general trend that can be seen from the above results so we can not say anything about the trends in hurricane activity over these 50 years, when we consider 10 years hurricane data. For the same reason we can not conclude anything about the technical advancement in hurricane tracking from this analysis.

3.4.5 Grid based clustering algorithm

We incorporate a grid based clustering algorithm for comparing the results obtained using DBSCAN algorithm. We use a grid based algorithm is called CLIQUE [14]. CLIQUE stands for CLustring In QUEst. CLIQUE is efficient in finding clusters in high dimensional space, where clusters may exist in some subspace of the original dimensional space. It is based on apriori principal. CLIQUE has two important parameters which are τ and $grid\_size$. τ is the density threshold where as $grid\_size$ is the parameter that determines the number of bins in each dimensions of the data. The clusters in CLIQUE are the maximal connected dense grid cells.

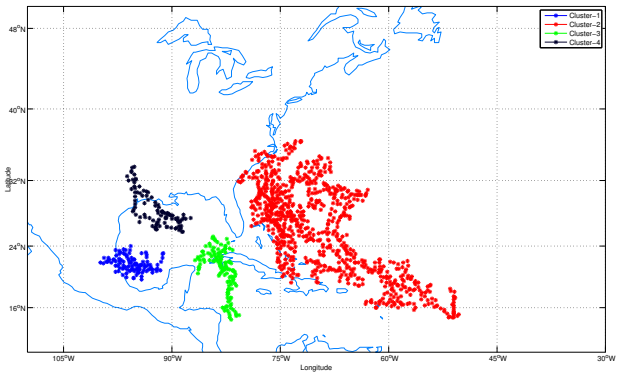


Figure 3.24: DBSCAN on Hurricane data for year 1950 to 1960 $Min_{Pts} = 10$, $Min_{Dst} = 35$

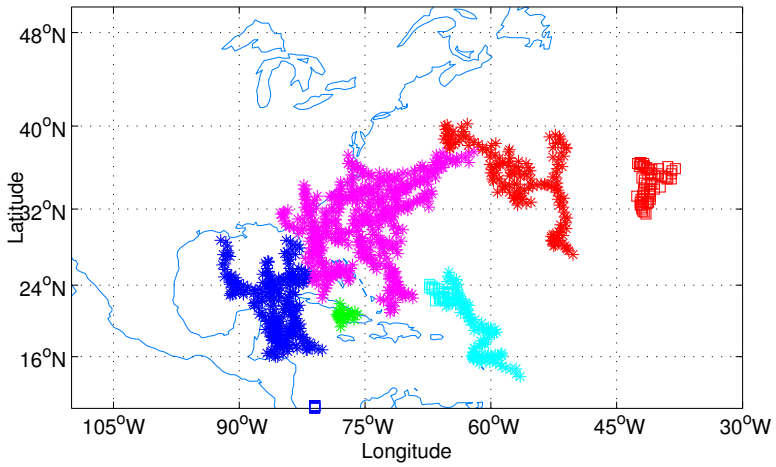


Figure 3.25: DBSCAN on Hurricane data for year 1961 to 1970 $Min_{Pts} = 10$, $Min_{Dst} = 35$

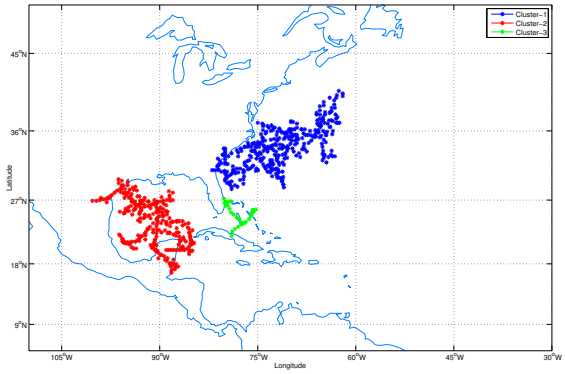


Figure 3.26: DBSCAN on Hurricane data for year 1971 to 1980 $Min_{Pts} = 10$, $Min_{Dst} = 35$

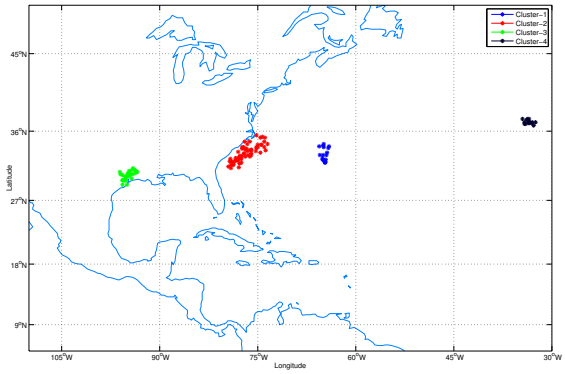


Figure 3.27: DBSCAN on Hurricane data for year 1981 to 1990 $Min_{Pts} = 10$, $Min_{Dst} = 35$

We present the results of spatial clustering using CLIQUE clustering algorithm. Figure 3.29 to Figure 3.36 represents the results. The problem with CLIQUE has been that in cases like Figure 3.29 to Figure 3.32 when we fix $\tau = 0.1$ and change the grid size from $30 * 30$ to $60 * 60$ there remains one big dominating cluster and the other clusters are very small and hence insignificant. In Figure 3.33 we increased the grid size to $70 * 70$, again we can see a very large cluster in the middle and a very large number of small clusters. In Figure 3.34 we increase the density threshold parameter

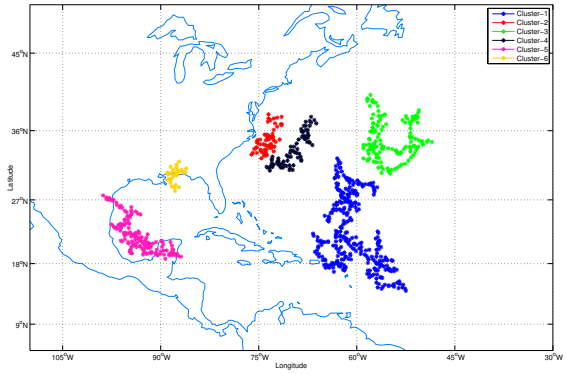


Figure 3.28: DBSCAN on Hurricane data for year 1990 to 2000 $Min_{Pts} = 10$, $Min_{Dst} = 35$

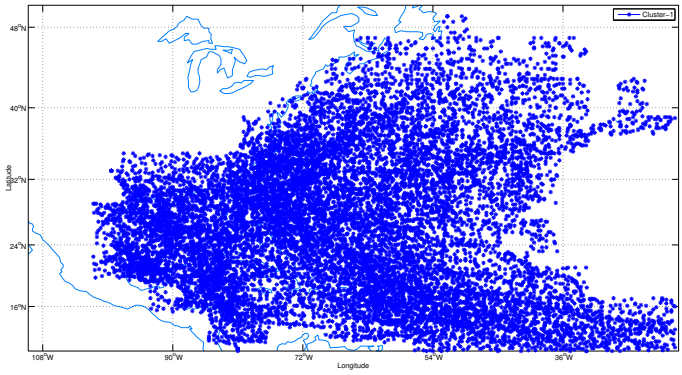


Figure 3.29: CLIQUE $\tau = 0.1$, $grid\_size = 30 * 30$

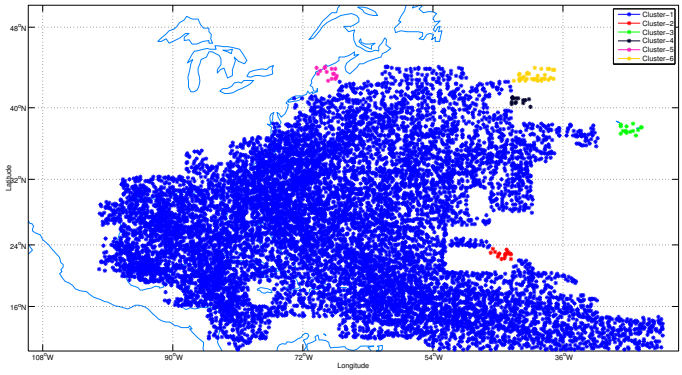


Figure 3.30: CLIQUE $\tau = 0.1$, $grid\_size = 40 * 40$

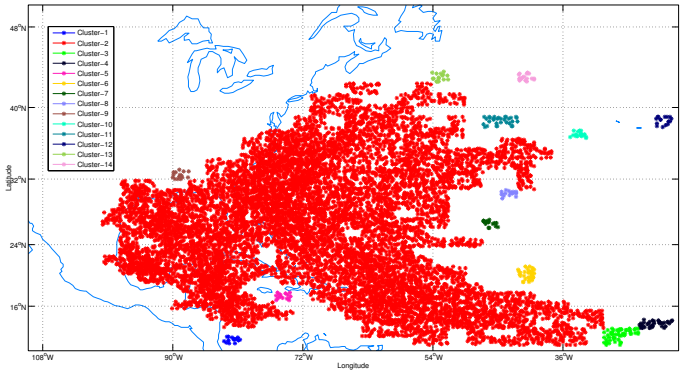


Figure 3.31: CLIQUE $\tau = 0.1$, $grid\_size = 50 * 50$

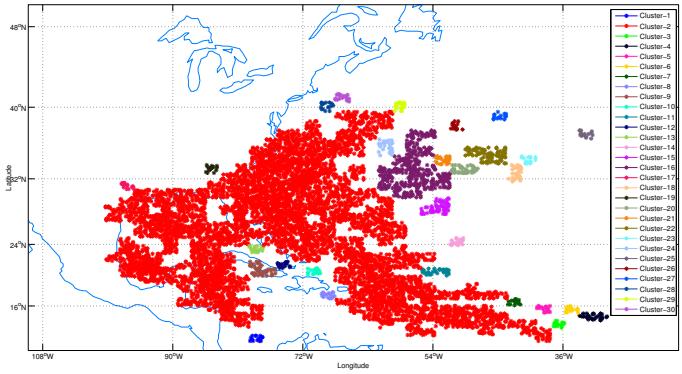


Figure 3.32: CLIQUE $\tau = 0.1$, $grid\_size = 60 * 60$

τ to 0.2 and grid size as $40 * 40$, but again we have the same scenario, there is one huge dominating cluster and the rest of the clusters are very small and insignificant.

When we increase the grid size to $70*70$ we get the result as given in Figure 3.35. Note that the results for the other grid size between $50 * 50$ to $70 * 70$ show the same trend. The result in Figure 3.35 are the result of very small size grid cells, with comparatively higher density threshold. There is an interesting result with CLIQUE as shown in Figure 3.36. This is the result corresponding to the value of $\tau = 0.5$ and grid size $= 30 * 30$. This result highlights another weak aspect of CLIQUE algorithm

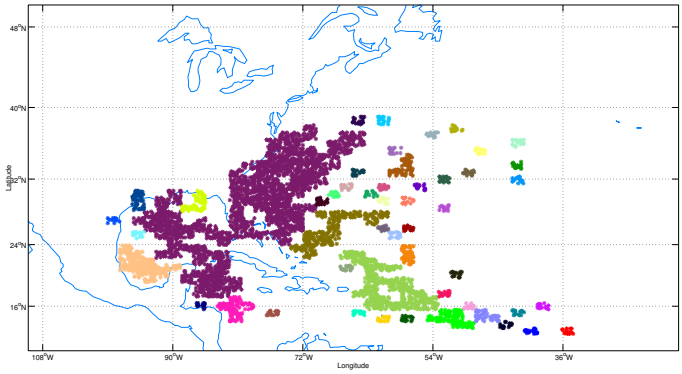


Figure 3.33: CLIQUE $\tau = 0.1$, $grid\_size = 70 * 70$

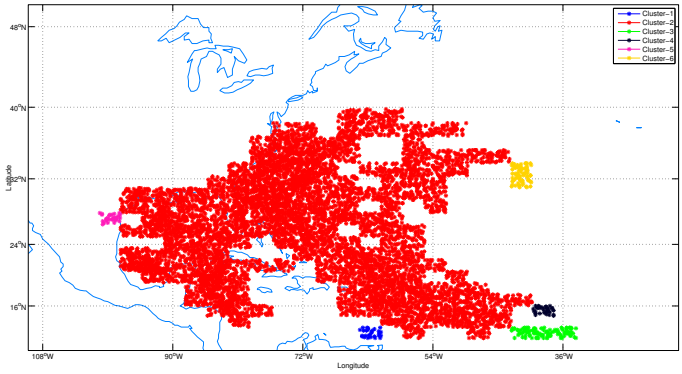


Figure 3.34: CLIQUE $\tau = 0.2$, $grid\_size = 40 * 40$

which has the tendency of producing the axis parallel clusters. CLIQUE produces axis parallel clusters because it is grid based clustering algorithm.

We present the clustering result of CLIQUE considering the spatial as well as non-spatial attribute which is wind speed in this case. Figure 3.37 and Figure 3.38. The real challenge in our experiments for CLIQUE was in finding the clusters with spatial and non spatial attributes (wind and time). This can be easily seen in results from Figure 3.37 and Figure 3.38. In Figure 3.37 when we choose $\tau = 0.1$, grid size as $30 * 30$ and $Min_{DstWspeed} = 30$ we get many small sized and overlapping clusters. In Figure 3.38 when we simply change the grid size from $30 * 30$ to $40 * 40$

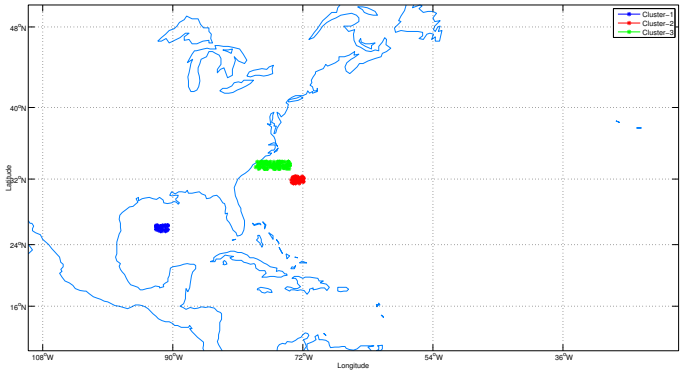


Figure 3.35: CLIQUE $\tau = 0.2$, $grid\_size = 70 * 70$

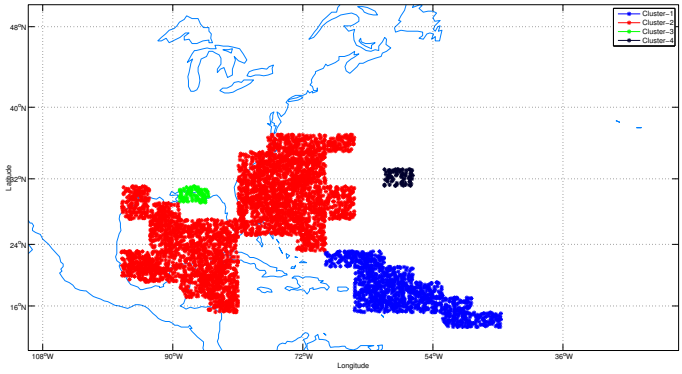


Figure 3.36: CLIQUE $\tau = 0.5$, $grid\_size = 30 * 30$

we get few very small clusters. This result is not as consistent as that with DBSCAN. This is because of the nature of CLIQUE, unlike DBSCAN we can not control the dimension specific neighborhood. In DBSCAN we pick the spatial neighborhood and the non-spatial neighborhood, the final neighborhood is the intersection of this spatial and non-spatial neighborhood. In case of CLIQUE we have a single parameter that determines the number of sub-divisions in each dimension. Because of this nature we can not control the size of the neighborhood in each data dimension independently of other dimensions. The performance of CLIQUE in doing spatio-temporal clustering was also not good for the aforementioned reason. One of the result of CLIQUE on

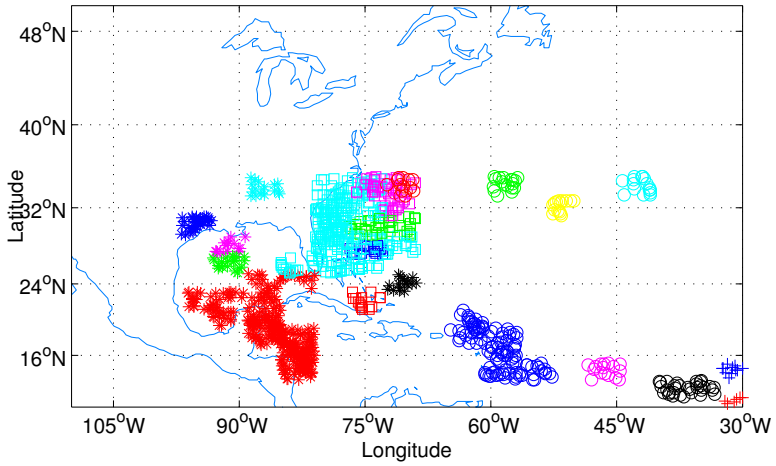


Figure 3.37: CLIQUE $\tau = 0.1$, $grid\_size = 30 * 30$, $Min_{DstWspeed} = 30$

spatio-temporal clustering is give in Figure 3.39, here we used $\tau = 0.1$, grid size $30 * 30$ and $\epsilon_t = 0.1$. Here we see just a single small cluster. Interestingly this is the only cluster that we could obtain after trying different combination of clustering parameters for CLIQUE.

In order to provide the comparative performance of point based analysis using DBSCAN and CLIQUE algorithm we provide the quantitative results in Table 3.5 and Table 3.6 respectively. Since the nature of two algorithms is different so we could not set the same value of parameters for them. We can see from these results that in terms of SSE, DBSCAN has resulted in better value for most of the cases. Similar trend can be seen for the value of measure $Mean(std)$, where we compute the mean of the standard deviation of the different cluster members. In terms of $Penalty$ which is dependent on the number of noise points CLIQUE is slightly better than DBSCAN.

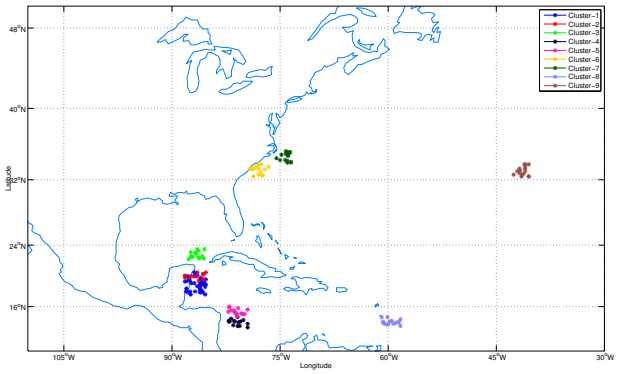


Figure 3.38: CLIQUE $\tau = 0.1$, $grid\_size = 40 * 40$, $Min_{DstWspeed} = 40$

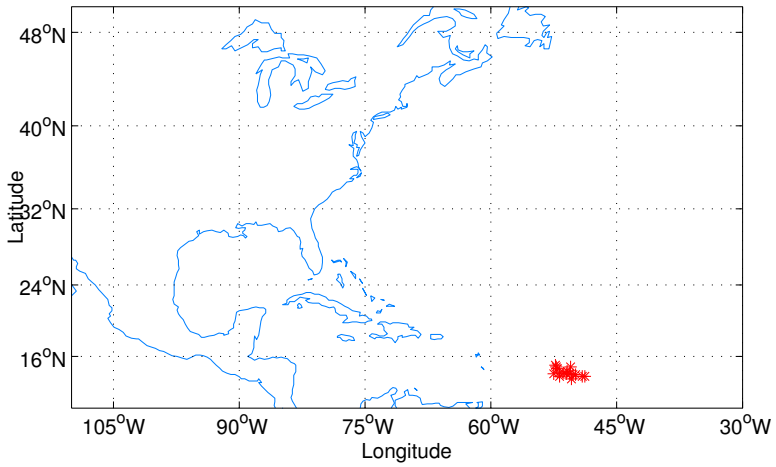


Figure 3.39: CLIQUE $\tau = 0.1$, $grid\_size = 30 * 30$, $\epsilon_t = 0.1$

Table 3.5: Qualitative measure of clustering results DBSCAN

Min_{Pts}	Min_{Dst}	SSE	Penalty	Mean(std)
8	35	1.05E+06	3.61E+10	1.75E+07
10	25	4.03E+03	4.29E+10	2.78E+04
10	30	1.54E+05	4.46E+10	7.82E+06
10	35	5.19E+05	3.96E+10	2.70E+07
10	40	1.85E+06	3.12E+10	2.08E+09
10	45	2.12E+06	2.68E+10	3.48E+09

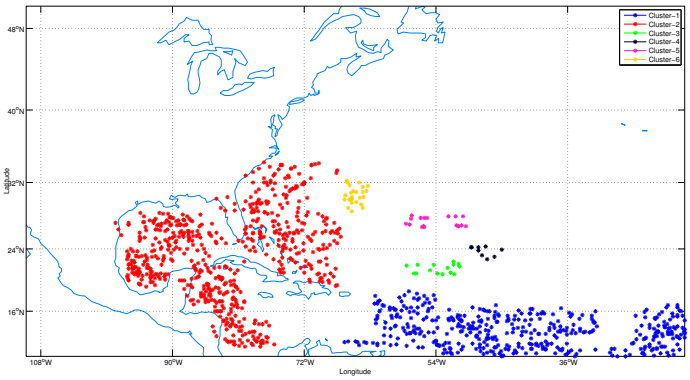


Figure 3.40: CLIQUE Storm starting locations

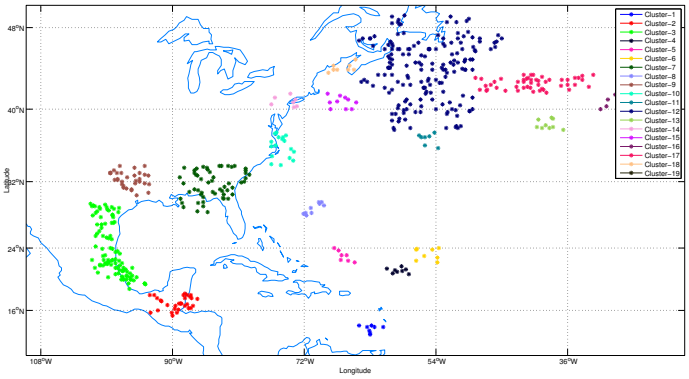


Figure 3.41: CLIQUE Storm landing locations

Table 3.6: Qualitative measure of clustering results CLIQUE

τ	$grid\_size$	SSE	Penalty	Mean(std)
0.1	30	2.49E+06	1.15E+10	5.39E+10
0.1	40	2.44E+06	1.95E+10	7.58E+09
0.1	50	2.37E+06	2.91E+10	2.21E+09
0.1	60	2.23E+06	4.18E+10	5.74E+08
0.1	70	1.34E+06	5.44E+10	3.87E+07
0.4	40	3.05E+05	6.84E+10	7.53E+06

Figure 3.40 and Figure 3.41 show the storm starting and storm landing locations. The storm starting locations as predicted in Figure 3.40 are not precise, clusters are rather scattered. Similarly for the clusters showing the landing locations of the storm, there are large number of very loose clusters.

3.5 Summary

In this chapter we analyzed hurricane storm trajectory data to find areas of extreme hurricane density, as well as areas where hurricanes originate and land. We took a different approach to trajectory analysis by focusing on points along the trajectory, rather than line segments as in previous work. We used the DBSCAN algorithm for the clustering analysis. Initially the clusters are obtained on the basis of only the spatial attributes. After that, we looked at the influence of the non-spatial attributes, in particular wind speed, on the clusters obtained. We also propose a spatio temporal DBSCAN algorithm where the normalized relative time information with the point data has been considered as another non-spatial attribute for DBSCAN. We post processed the clustering results to obtain the storm starting, storm landing and storm tracking information. Our work differs from other work because of the focus on trajectory points, which results in identifying high-activity regions, as well as regions at start and end of storms. In an attempt to analyze the trends in storms activity over the years, we did our analysis on hurricane data by considering 10 years of data at a time ($1950-1960$, $1961-1970$, $1971-1980$, $1981-1990$, $1991-2000$). This analysis did not show any significant trend in storm activity for fifty years of that we analyzed. In order to highlight our approach we compare our DBSCAN based point analysis with CLIQUE based point analysis. Experiments have shown that the DBSCAN based analysis results in better results. Further unlike CLIQUE, DBSCAN

based analysis is more flexible because of which we could find results for all the range of parameter values.

CHAPTER 4

A Direction Based Framework for Trajectory Data Analysis (Drafted from [2])

4.1 Introduction

Trajectory data is an example of spatio temporal data where the spatial location as well as the time order associated with each data point is very important. Some examples of spatio temporal data are tracking data in applications like global positioning system (GPS), hurricane and storm tracking data, and animal movement data. The analysis for GPS trajectories is valuable in knowing and managing the traffic pattern of vehicles. Monitoring and analyzing hurricane/storm trajectories can be useful for predicting weather conditions, whereas animal trajectory analysis is used in examining wild animal behavior and movement. A number of attempts have been made in this domain to analyze these kinds of data sets [5, 6, 9, 15, 20, 21].

The directional aspect of trajectory analysis is very important in various applications [15], for example in map matching [22] and in direction based query processing [21]. This kind of analysis will be very useful in analyzing weather data (hurricane tracks), public transport data (GPS) and animal movement data. To the best of our knowledge this directional analysis has not been done previously in spatio temporal data.

The first stage of the proposed framework deals with *trajectory smoothing*, which is important in analyzing trajectories that exhibit non-smooth characteristics. Trajectory smoothing (simplification) has been addressed in [15–17]. Non-smoothness is marked by frequent sharp directional changes. The trajectories in animal movement data exhibits this non-smooth property. In the smoothing step we approximate the

original trajectories by eliminating these sharp angular turns to focus on directional characteristics of the trajectory.

In the next stage we do *directional segmentation* of the trajectories. We impose directional consistency on the trajectories (sub trajectories) by allowing them to deviate initially by a maximum of 45, and eventually by a maximum of 90. If any trajectory shows a deviation more than 90, we split it into sub-trajectories such that the directional consistency is maintained. We consider 16 directional ranges shown in Fig. 4.1 and Fig. 4.2. Each sub-trajectory is assigned to one of these 16 classes. Analysis of trajectory data after segmentation has been a well studied domain in spatio temporal data, for example [5, 6, 9, 20].

The filtration stage removes outlier sub-trajectories from the directional categories to focus on important directional ranges. We considered two approaches for this task. The first approach uses the *minimum bounding rectangle (MBR)*, whereas the second one uses a novel *convex hull (CH)* based approach (Fig. 4.3 and Fig. 4.4). The MBR based approach results in big spaces around a trajectory; hence it is a very vague approximation of the trajectory. The CH based approach approximates the trajectories to a much closer extent. We finally use a modified DBSCAN [13] algorithm to identify the inherent clusters, which capture the significant directional patterns in the data sets. We used animal movement data [23] that consists of movement of Elk in 1993 and comprises 33 trajectories and 20065 data points.

The first stage of the proposed framework deals with the task of trajectory smoothing, which is important in analyzing trajectories that exhibit non-smooth characteristics. Trajectory smoothing (simplification) has been addressed in [15–17]. Non-smoothness is marked by frequent sharp directional changes. The trajectories in animal movement data exhibits this non-smooth property. In the smoothing step

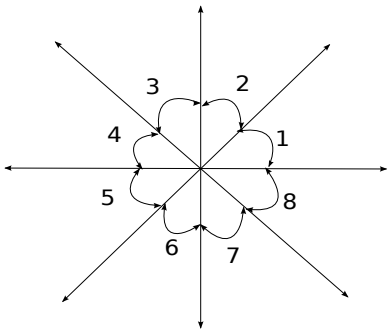


Figure 4.1: Small angles (45)

we approximate the original trajectories by eliminating these sharp angular turns to focus on directional characteristics of the trajectory.

In the next stage we do directional segmentation of the trajectories. For this task, we characterize the trajectories and sub-trajectories on the basis of their directions. We want the trajectories (sub-trajectories) to be directionally consistent. Here we impose the directional consistency on the trajectories (sub trajectories) by allowing them to deviate initially by a maximum of 45, and eventually by a maximum of 90. If any trajectory shows a deviation more than 90, we split it into sub-trajectories such that the directional consistency is maintained. To accomplish directional consistency for sub-trajectories, we consider 16 directional ranges shown in Fig. 4.1 and Fig. 4.2. Each sub-trajectory is assigned to one of these 16 classes. Analysis of trajectory data after segmentation has been a well studied domain in spatio temporal data, for example [9], [5], [6] and [20].

The filtration stage removes outlier sub-trajectories from the respective directional categories to focus on important directional ranges. We considered two approaches for this task. The first approach uses the *minimum bounding rectangle (MBR)*, whereas the second one uses a novel *convex hull (CH)* based approach. Examples of MBR and CH applied to direction *class* 4 for hurricane data set can be

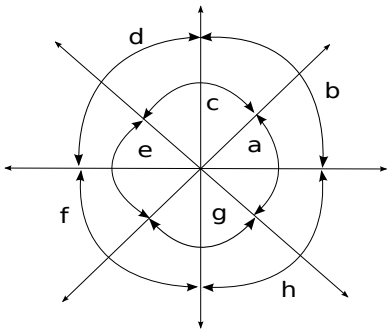


Figure 4.2: Large angles (90)

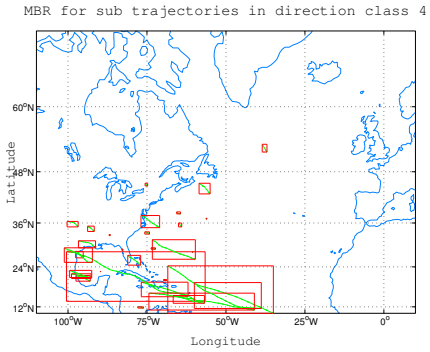


Figure 4.3: MBR Example

seen in Fig. 4.3 and Fig. 4.4 respectively. We can see that the MBR based approach results in big spaces around a trajectory when approximated by its MBR, and hence it may be a very vague approximation of the trajectory. This fact motivated us to use the CH based approach as it approximates the trajectories to a much greater extent.

We finally use a modified DBSCAN [13] algorithm to identify the inherent clusters, which are interesting patterns as they capture the significant directional patterns in respective data sets.

We used two real data sets: hurricane data [19] and animal movement data [23]. The hurricane (Atlantic region) data set is for 50 years from 1950 to 1999. It has six attributes: latitude, longitude, time, wind speed, pressure, and status. The data is sampled in 6 hours interval, it has 15319 data points and 496 trajectories. The

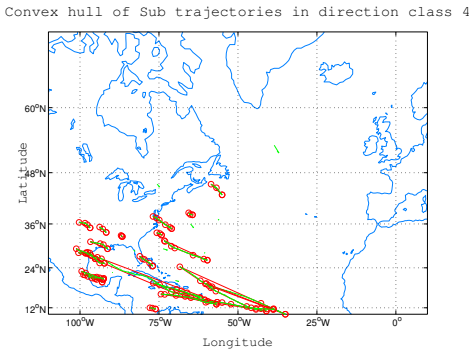


Figure 4.4: CH Example

animal movement data [23] consists of movement of Elk in 1993 and comprises 33 trajectories and 20065 data points.

The rest of the article is organized as follows: **Section** 4.2 provides related work, **Section** 5.2 describes the proposed framework, **Section** 4.4 presents the experiments and results, and **Section** 4.5 concludes the article.

4.2 Related Work

The article [5] proposes a partition-and-group framework for clustering trajectory data. Using the concept of minimum descriptive length (MDL) the original trajectories are segmented. These segments are called trajectory partitions, which are then clustered using a modified version of DBSCAN algorithm, which clusters the line segments. Finally the clusters are represented by representative trajectories.

In [6], a clustering algorithm is given for the trajectory data that uses a combination of techniques from data mining, computational geometry and string processing. Trajectories are preprocessed followed by segmentation and classification. The next phase finds the frequent occurring sub-strings; these are called $motifs$, and then maps the sub-trajectories corresponding to the motifs to some feature space. The next stage

performs density based clustering and the final stage does the post processing of the clusters.

In [9], the trajectory clustering technique of [5] has been extended for trajectory classification. Two levels of clustering, namely region-based and trajectory based clustering, are done. Clustering is used to find the discriminative features for classification. The first level of clustering is region level, which identifies the higher level, region based features of the trajectories. The second level identifies the lower level movement based features. These two clustering levels collaboratively identify the high-quality features for the classification.

Trajectory smoothing has been addressed in [15–17]. In [15] authors propose a trajectory smoothing method that preserves the direction information. The aforementioned works are similar to the present work on the basis of trajectory segmentation followed by the clustering of the trajectory segments. However, our contribution is on the directional consistency, so our trajectory segmentation method is purely direction based. The other contributions are trajectory simplification and convex hull based outlier removal concepts.

4.3 Proposed framework

Figure 4.5 shows a directional pattern scenario. These kinds of directional pattern mining are the motivation behind the work in this chapter. In this figure we can clearly see three dominant directional regions. These regions are characterized by Dir_1, Dir_2 and Dir_3.

Before we describe the proposed framework, we define a trajectory and its major defining characteristics. Let us consider, a trajectory data set D comprising of n trajectories viz., $D = (Tr_1, Tr_2, \ldots, Tr_n)$.

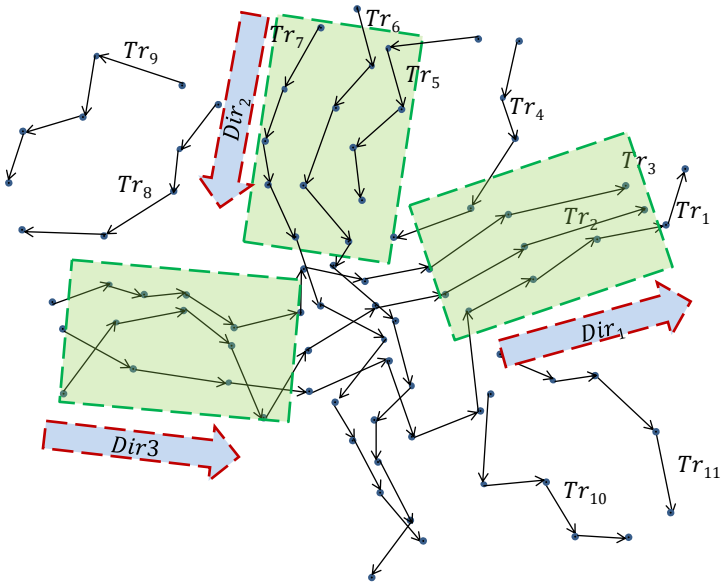


Figure 4.5: Directional pattern

Trajectory Definition: A trajectory Tr_j of size s is a sequence of points $[p_1, p_2, \ldots, p_s]$, where p_1 is it's initial point and p_s is the final point. An i^{th} point p_i in Tr_j is associated with spatial co-ordinates (x_i, y_i) and the associated time t_i. For example, in Figure 5.1 *a*, $Tr = [p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11}]$ is a trajectory of size $s = 11$.

Trajectory Segments (L_k): A trajectory Tr_j consists of line segments $L_k = \overline{p_k p_{k+1}}$ which are formed by joining the k^{th} and $(k+1)^{th}$ consecutive points in it, where $k \in [1, \ldots, s-1]$.

Angular Attribute (θ): This is a very important attribute of a trajectory for directional analysis, which considers the angles between its L_k and L_{k+1} consecutive line segments. This involves three successive points: p_k, p_{k+1} and p_{k+2}.

$$\theta = min(\angle(\overline{p_k p_{k+1}}, \overline{p_{k+1} p_{k+2}}), 360 - \angle(\overline{p_k p_{k+1}}, \overline{p_{k+1} p_{k+2}})) \tag{4.1}$$

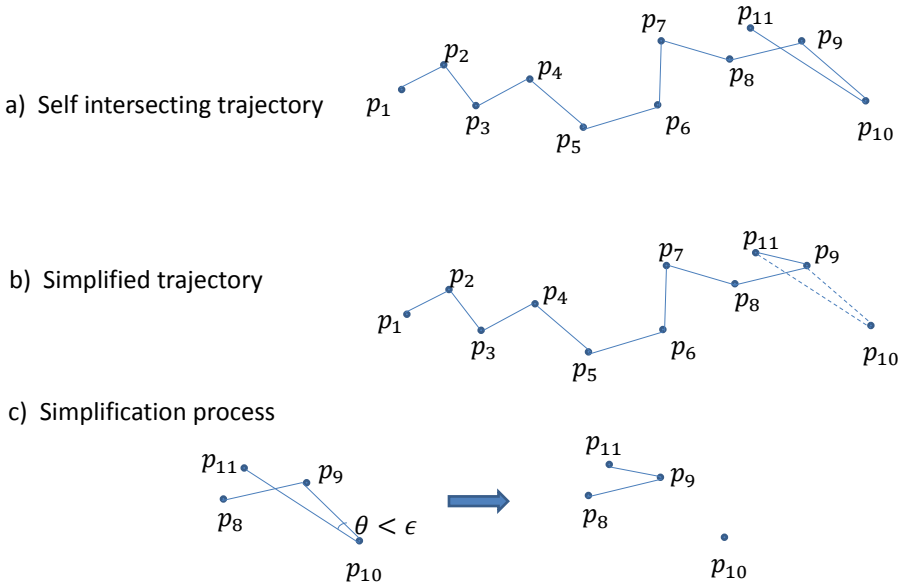


Figure 4.6: Self intersecting Trajectories

where, angle between the two line segments $\overline{p_k p_{k+1}}$ and $\overline{p_{k+1} p_{k+2}}$ is $\angle(\overline{p_k p_{k+1}}, \overline{p_{k+1} p_{k+2}})$. Angle is measured in anti-clockwise rotation from $\overline{p_k p_{k+1}}$ to $\overline{p_{k+1} p_{k+2}}$. We consider the smaller of the angles between the two line segments so that $0 \leq \theta \leq 180$. A large value of θ represents a small change in direction whereas a small value represents a sharp change. $\angle$ is the symbol for absolute value of an angle.

The proposed framework consists of the following stages:

1. trajectory smoothing
2. trajectory segmentation and categorization
3. convex hull based sub-trajectory filtration
4. clustering

4.3.1 Trajectory Smoothing

Unlike hurricane trajectories [19], animal movement trajectories [23] tend to be very haphazard and are non-uniformly sampled. For meaningful trajectory data analysis, trajectory smoothing becomes imperative. Although, the smoothing of trajectories has been studied in [15–17], self-intersecting trajectories (see Fig. 5.1) are not explicitly addressed.

Trajectory Smoothness $(sm\,(Tr))$**:** Smoothness of a trajectory determines its directional consistency over its whole length. In this work as a smoothness measure, we consider the mean $(sm_{\mu}\,(Tr))$ as well as the standard deviation $(sm_{sd}\,(Tr))$ of the angular attributes of a trajectory. If $\overline{\Theta}$ is the vector of the angular attributes of a trajectory Tr, then : $sm_{\mu}(Tr) = mean(\overline{\Theta})$, $sm_{sd}(Tr) = sd(\overline{\Theta})$. A large $sm_{\mu}(Tr)$ and a small $sm_{sd}(Tr)$ would indicate a smooth trajectory, whereas small values for $sm_{\mu}(Tr)$ indicate a very jagged trajectory. The simplified (approximated) trajectory should have better smoothness to highlight its main directional properties. If an initial trajectory is Tr, then the goal of the smoothing process is to simplify it to a trajectory Tr_s such that $sm_{\mu}(Tr) \leq sm_{\mu}(Tr_s)$ and $sm_{sd}(Tr) \geq sm_{sd}(Tr_s)$.

To address the unevenly sampled trajectories, we segment a bigger trajectory into sub-trajectories at a point where, the time difference between the successive sampled points is larger than a given threshold (30 minutes for animal data).

Fig. 5.1 shows how we deal with the self-intersecting trajectories as part of the smoothing process. The segment $\overline{p_{10}p_{11}}$ intersects the segment $\overline{p_8p_9}$ because the angle between the segment $\overline{p_9p_{10}}$ and $\overline{p_{10}p_{11}}$ is very small. As mentioned earlier, small angles indicate sharp turns leading to non-smooth trajectories. Self-intersecting trajectories are the results of extreme cases of small angles. Therefore, as is given in the Fig. 5.1, using small angle threshold ϵ, if an intermediate angle is less than this threshold, we filter it out by discarding the intermediate point (p_{10} in this case). This will result

Algorithm 7 The Directional Segmentation Algorithm

$DIRSEGMENT(T)$

$S(T) = NULL$

for all $traj$ in data-set T **do**

$trajDir_b = computeBasicDirection(traj)$

$trajDir_g = computeGeneralDirection(trajDir_b)$

$DirectSeg_{traj} = directionalSegment(trajDir_g)$

$S(T) = S(T) \cup DirectSeg_{traj}$

end for

in the formation of the new segment $\overline{p_9 p_{11}}$. If the resulting angle is still less than ϵ, further smoothing is applied.

4.3.2 Trajectory Segmentation

In this step of the proposed framework, trajectories are segmented into sub-trajectories belonging to one of the 16 directions depicted in Fig. 4.1 and Fig. 4.2. Eight of these directions (Fig. 4.1), are 45 apart covering the whole 360 angular space, whereas the remaining ones (Fig. 4.2) are 90 apart. This choice of the smaller angle directional segment viz., 45 and a larger overlapping directional segments viz., 90 was made so that a trajectory that moves along a boundary between 45 regions of the two regions, say 45, does not keep being segmented into very short segments, in this case some in region 1 and some in region 2. Rather, a longer sub-trajectory in region b would be created.

Algorithms 7-10, describe the concepts used for segmentation of the main trajectory into the corresponding directional sub-trajectories.

Algorithm 8 The basic 8 direction Segmentation Algorithm

$computeBasicDirection(t)$

$dir_{basic} = NULL$

for all ls in t **do**

$\theta_i = slope_{xAxis}(ls)$

$d_i = \frac{\theta_i}{45} + 1$

$dir_{basic}.append(d_i)$

end for

$return(dir_{basic})$

Algorithm 9 The general direction Segmentation Algorithm

$computeGeneralDirection(t)$

$gen_{dir} = NULL$

$temp_{dir} = NULL$

$breakPoint = NULL$

$label = NULL$

$Scan\ t\ from\ left\ to\ right$

$breakPoint = findAndStoreBreakPoints(t)$

$labelGeneral = findGeneralLabels(t)$

$gen_{dir} = segment(breakPoint, labelGeneral)$

$return(gen_{dir})$

Algorithm 10 The Directional Segmentation Algorithm

$directionalSegment(t)$

$directSeg = segment(t, differentSymbols)$

$return(directSeg)$

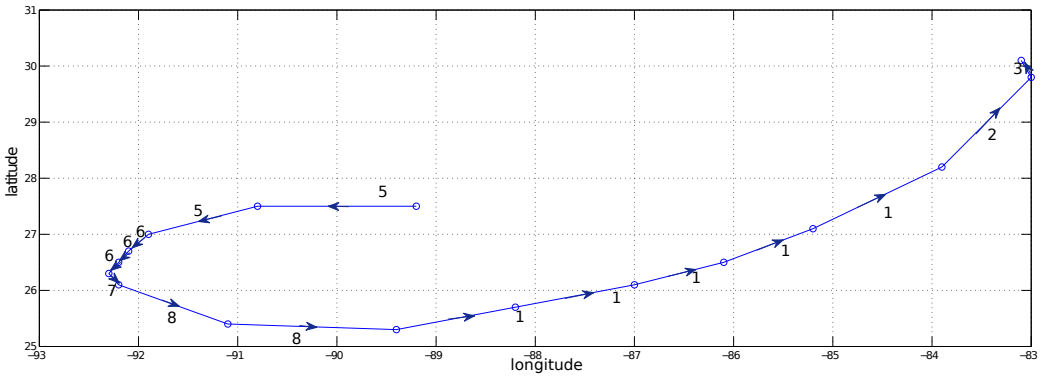


Figure 4.7: Basic directional encoding of Hurricane Love 1950

Transitions	*General label*
(1 -> 2) or (2 -> 1)	*b*
(5 -> 6) or (6 -> 5)	*f*
(7 -> 8) or (8 -> 7)	*h*

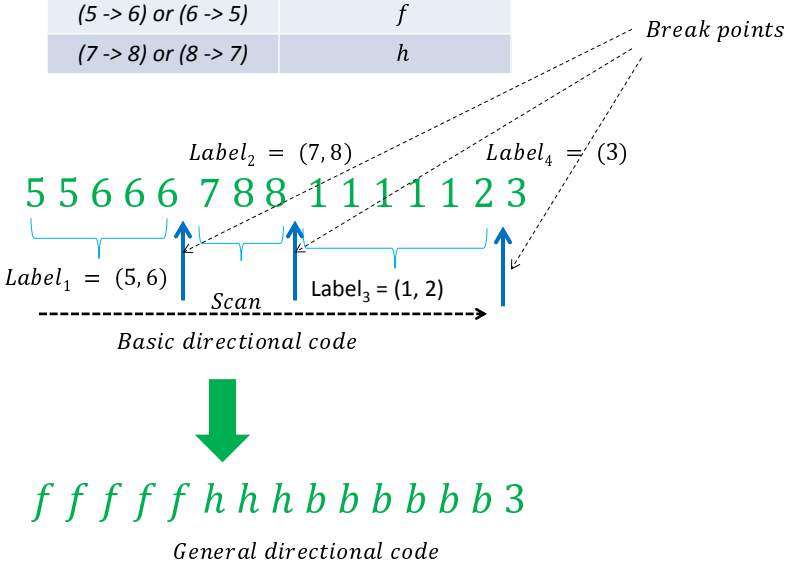


Figure 4.8: Basic to General direction encoding of Hurricane Love 1950

Initially (**Algorithm 8**) we examine the slope of the line segments forming the trajectory, and label the line segments according to their slope. Since the basic angle for segmentation is in this work is 45, the total number of labels is $360/45 = 8$. Therefore the label for a particular line segment will be the quotient when the angle of

the slope is divided by 45. The example in Fig. 4.7 corresponds to Hurricane Love in 1950. Using the basic directional encoding algorithm, the trajectory is now labeled as (556667881111123) (Fig. 4.7). This output becomes an input for the next algorithm **Algorithm 9** which does the general directional encoding.

Algorithm 9 scans the basic directional code of the trajectory from left to right and looks for a break point (please see Fig.4.8). The break point is the position along the basic directional encoding where, the direction changes by 2 or more labels. Along with the break point a *Label* list is also stored, this list will hold the basic angular transitions so that it may be accordingly encoded in the general directional code. For example in Fig. 4.8, break points create 4 sub-trajectories 55666, 788, 111112 and 3, the last is discarded as too short. Corresponding $Label_i$ lists have the entries $(5, 6)$, $(7, 8)$ and $(1, 2)$, respectively. These transitions correspond to the general angle code of f, h and b respectively (Fig. 4.2, Fig. 4.8). The break point will act as a marker for the **Algorithm 10**, while segmenting the original trajectory into its directional sub-trajectory. Here the output will be ($fffffhhhbbbbbb$). The final algorithm **Algorithm 10** does the segmentation returning the three sub-trajectories which are ($fffff$), (hhh) and ($bbbbbb$).

4.3.3 Convex Hull based trajectory filtering to remove outliers

We use convex hull (CH) to approximate each trajectory, then apply a filtering step to the convex hulls.

Definition: Convex Hull : The convex hull of a set of points Q, denoted as $CH(Q)$, is the smallest convex polygon P for which each point in Q is either on the boundary of P or in its interior [24]. For example in Fig. 4.9, $Q = \{P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8\}$. In this example, the convex hull is the polygon $P = \{P_1, P_2, P_3, P_4, P_5\}$, because all the points in this figure either lie on or are inside P.

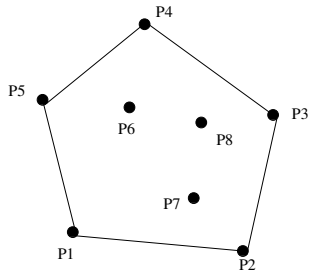


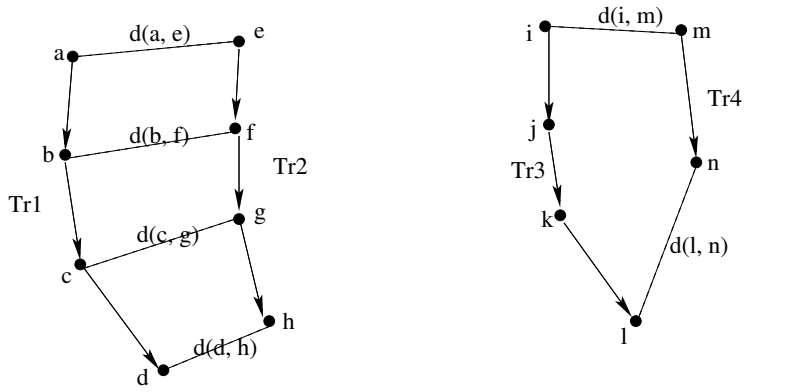
Figure 4.9: Convex Hull

After the approximation of sub-trajectories by their respective convex hulls, we identify the sub-trajectories whose convex hull do not intersect with any other sub-trajectory. We use geometrical algorithm to find out the intersecting convex hulls. These sub-trajectories with non intersecting convex hull are considered as outlier and are removed from the later stages of analysis.

4.3.4 Clustering

After removing the outlier sub-trajectories using the CH based technique, we do the clustering of the remaining sub-trajectories to obtain the final directional patterns **Algorithm** 11. We use a modified version of DBSCAN [13], which is a density based clustering algorithm. This is very similar to the one used in [5], except for the fact that here we have the sub-trajectories as the basic data to be clustered in stead of line segments. We omit the details of DBSCAN due to space limitations, see [5, 13] for the details.

For the DBSCAN algorithm we need a distance measure to find the similarity between the sub-trajectories. Fig. 4.10 shows how we computed the distance between the two sub-trajectories. There are two possibilities, Fig. 4.10(a) where the two sub-trajectories have the same length, and Fig. 4.10(b) where the length of the two sub-trajectories is not the same. For the first case we simply consider the average



a) Tr1 and Tr2 trajectories have same length =4

b) Tr3 length = 4, Tr4 length = 2

Figure 4.10: Distance between two sub-trajectories

of the distance between the sequential points in the sub-trajectories. In the second case we consider only the average of the distance between the first points and the last points in the sub-trajectories. In Fig. 4.10(a) for example the distance will be $D(Tr_1, Tr_2) = (d(a,e) + d(b,f) + d(c,g) + d(d,h))/4$. In Fig. 4.10(b) the distance will be $D(Tr_3, Tr_4) = (d(i,m) + d(l,n))/2$. In this atricle the distance $d(x,y)$ is the $Haversine\ formula$ as used in [1] for hurricane data, whereas the *Euclidean distance* for animal data.

Algorithm 11 The Directional Clustering Algorithm

$DIRECTCLUST(T)$

$DIRSEGMENT(T)$

$T_r = outlierFilter(T)$

for all $Traj_{class}$ in dataset T_r **do**

$Clust_{class} = DBSCAN(Traj_{class})$

end for

Table 4.1: Effect of Smoothing on Elk data set

$Trajectories$	$sm_{\mu}(Tr)$	$sm_{sd}(Tr)$
Original Animal Data	75.93	55.72
Smoothed Animal Data	87.96	51.08

4.4 Experiments and Results

The framework has been implemented in MATLAB. We use two real data sets as described in Section 4.1. In this work we did not simplify the hurricane trajectories as they already exhibit smooth behavior. The result of applying smoothing on animal movement trajectories can be seen in Table 4.1. The impact of smoothing is evident as sm_{μ} has been improved (is higher) and, the value of sm_{sd} is lower after the application of smoothing, compared to that of the original trajectories. These results are the aggregate values considering all the trajectories together.

For the DBSCAN algorithm, on hurricane data we use $Min_{line} = 5$, whereas for animal data it is 8 (minimum number of sub-trajectories in the neighborhood of a trajectory for density consideration). Min_{dst} (neighborhood radius) has been computed as per the suggestions in [5]. On applying the present framework on hurricane data, we found that out of the 16 directions (Fig.4.1 and Fig.4.2), clusters exists only in 6 directions, which are $(1, 9, 10, 11, 12, 13)$ (Fig. 4.15 – 4.26). This is an important result highlighting the fact that the hurricanes are mostly effective in the above mentioned 6 directions only. The remaining direction sub-trajectories were found to be all outliers at the outlier elimination stage of the proposed framework.

To find the significance of the our framework, we compare the clustering results on original hurricane trajectory data (before filtration and direction classification) Fig. 4.11, with clustering results obtained using our framework, Fig. 4.12. It is evident that the clusters in Fig. 4.11 are not very distict and are mostly overlapping. On the other hand the clusters in Fig. 4.12 are more distinct. Moreover, in Fig. 4.11 we see a

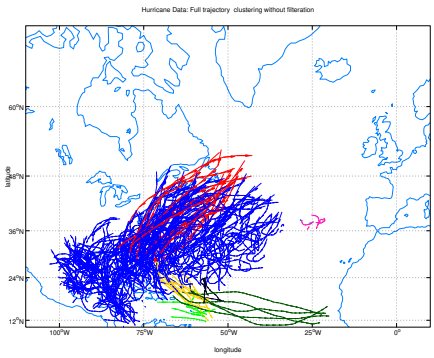


Figure 4.11: Trajectory clustering without filtering

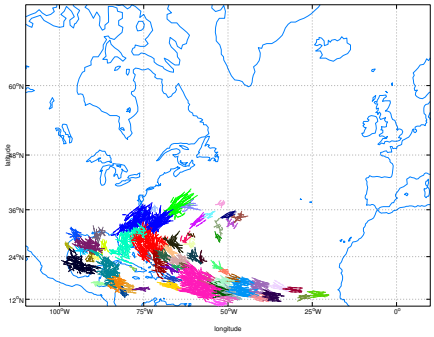


Figure 4.12: Hurricane Data combined sub-trajectory clustering

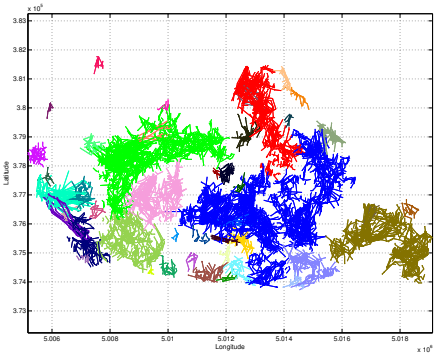


Figure 4.13: Trajectory original data

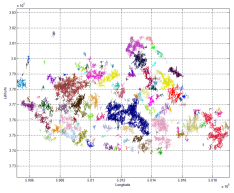


Figure 4.14: Trajectory clustering using the framework

big *blue* color cluster that spans a larger region, which only shows a region with high storm activity, whereas; in the same region in Fig. 4.12, we see many more compact clusters which carry an important information which is direction. Similarly we can see from Fig 4.14 and Fig. 4.13 that, the result obtained by the proposed framework Fig. 4.14 results is more number of distinct clusters as compared to the Fig. 4.13 (without applying the framework).

Fig. 4.15 – Fig. 4.26 provide the visualization of the final patterns obtained by the proposed framework on the 50 years hurricane data in the 6 significant directions. Here we have shown the comparative performance of the two outlier filtration mechanisms, which are the CH (proposed) and the MBR. Clearly we can see that the patterns (which are the directional clusters here in this work) are more prominent and compact using the CH based filtering technique.

Similarly we can see the same comparison on animal data from Fig 4.27 – Fig 4.42.

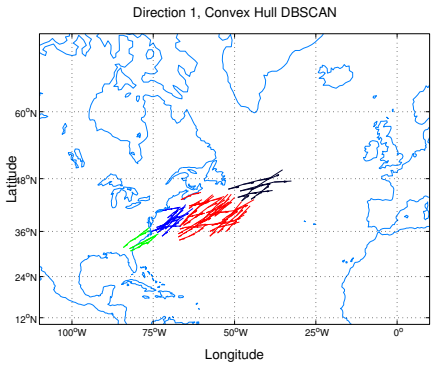


Figure 4.15: Convex Hull DBSCAN, Direction 1

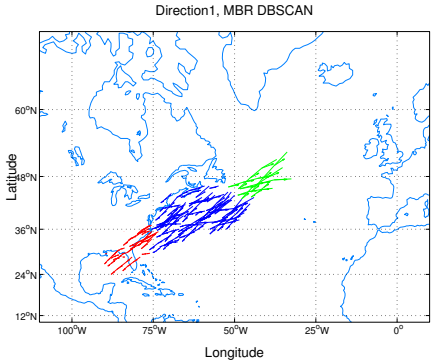


Figure 4.16: MBR DBSCAN, Direction 1

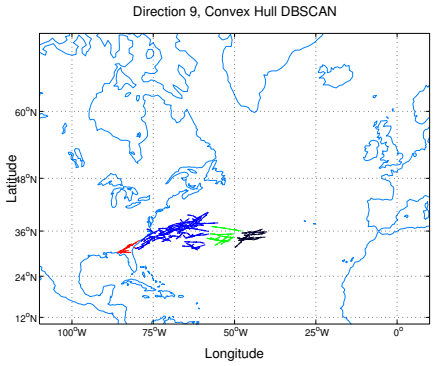


Figure 4.17: Convex Hull DBSCAN, Direction 9

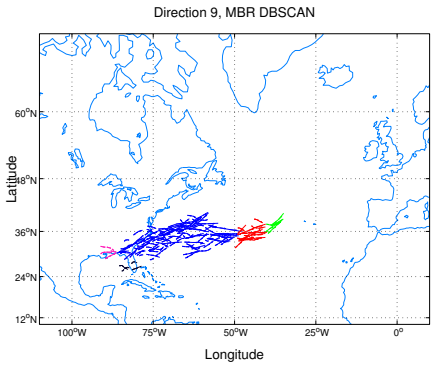


Figure 4.18: MBR DBSCAN, Direction 9

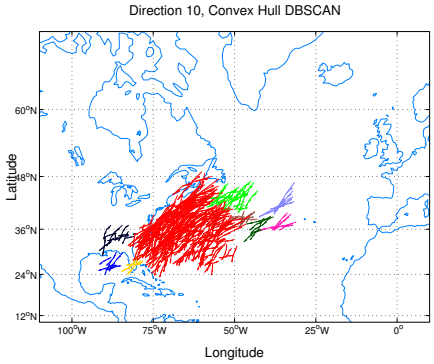


Figure 4.19: Convex Hull DBSCAN, Direction 10

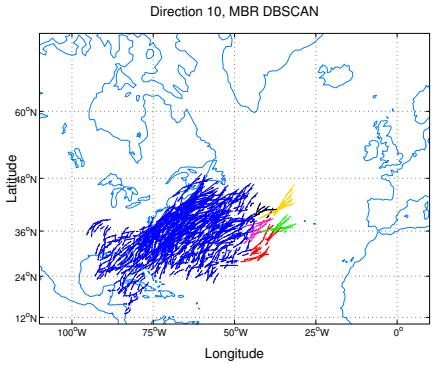


Figure 4.20: MBR DBSCAN, Direction 10

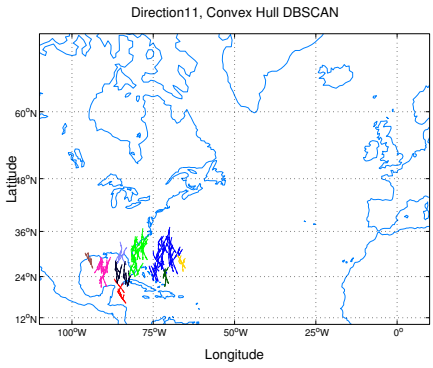


Figure 4.21: Convex Hull DBSCAN, Direction 11

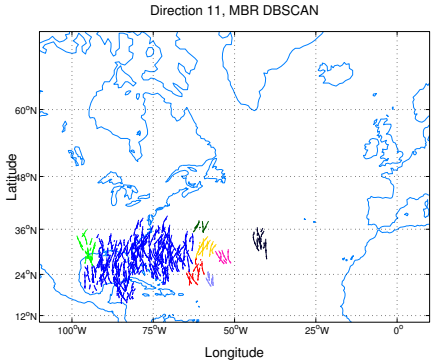


Figure 4.22: MBR DBSCAN, Direction 11

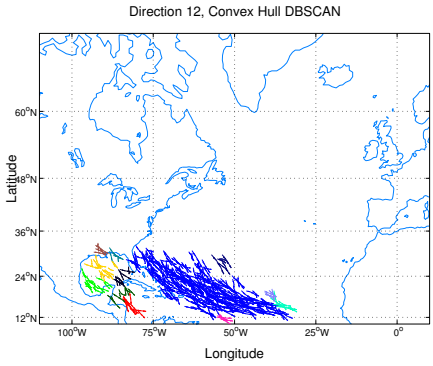


Figure 4.23: Convex Hull DBSCAN, Direction 12

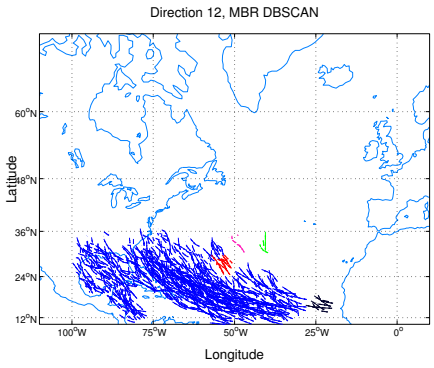


Figure 4.24: MBR DBSCAN, Direction 12

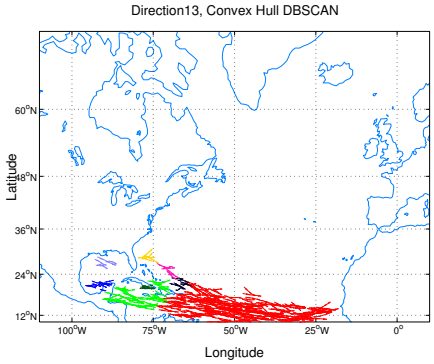


Figure 4.25: Convex Hull DBSCAN, Direction 13

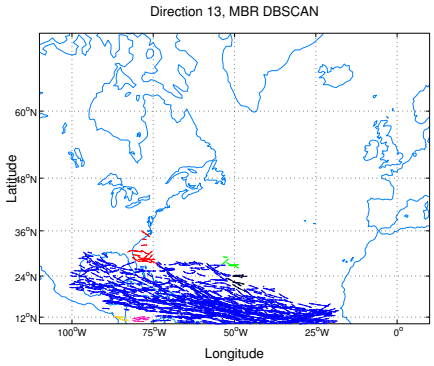


Figure 4.26: MBR DBSCAN, Direction 13

We did the qualitative analysis of the obtained results as given in the next sub section.

4.4.1 Qualitative analysis of clusters:

We used the quality measure proposed in [5]. This measure is given below:

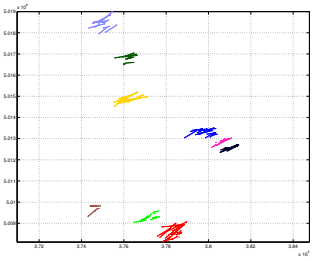
$$\sum_{i=1}^{num_{clus}} \left(\frac{1}{2C_i} \sum_{x \in C_i} \sum_{y \in C_i} dist(x,y)^2 \right) + \frac{1}{2|N|} \sum_{w \in N} \sum_{z \in N} dist(w,z)^2 \tag{4.2}$$

where, num_{clus} is the number of clusters, N is the set of noise sub-trajectories and C_i is the i^{th} cluster. This quality measure computes the sum of the square error (SSE), which means the smaller this value, the better will be the clustering result.

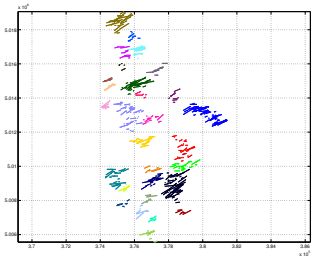
In order to show the effectiveness of the proposed CH based filtration method over the MBR based method we present the results in Fig. 4.43 and Fig. 4.44 using the evaluation method given in Equation 4.2. In Fig. 4.43, it can be seen that CH based filtration process resulted in lower values of $Qmeasure$ compared to its MBR counterpart. Similar trend can be seen for animal data also from Fig. 4.44. It implies that our framework produces better results for all the directions in terms of $QMeasure$ for both the data sets.

4.5 Summary

In this chapter we proposed a novel framework for the directional analysis of trajectory data. We proposed a new trajectory smoothing approach as well as a novel CH based filtration method using convex hulls of sub-trajectories. A novel technique for identifying the directional orientation of the trajectory data was proposed. Clustering is then applied to determine interesting directional clusters of trajectories. We demonstrate our approach on two data sets: hurricane tracks and animal trajectories. the results show that our clustering is good using standard cluster evaluation metric.

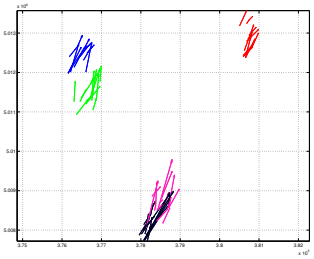


(a) Convex Hull DBSCAN, Direction 1

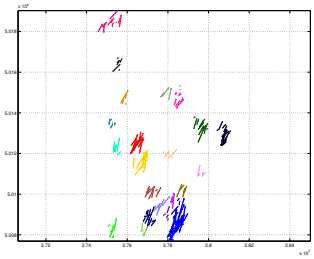


(b) MBR DBSCAN, Direction 1

Figure 4.27: Animal data : Relative performance Direction 1

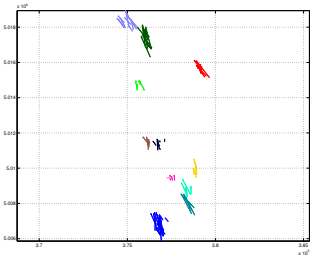


(a) Convex Hull DBSCAN, Direction 2

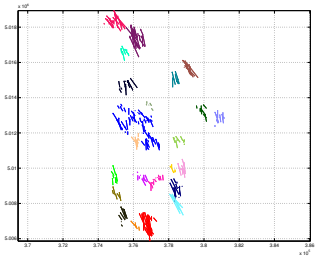


(b) MBR DBSCAN, Direction 2

Figure 4.28: Animal data : Relative performance Direction 2

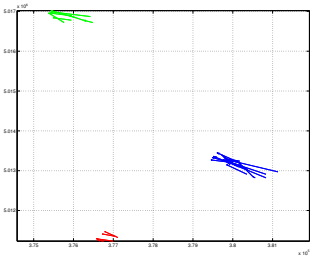


(a) Convex Hull DBSCAN, Direction 3

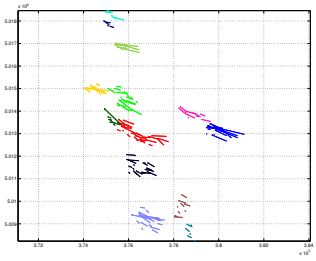


(b) MBR DBSCAN, Direction 3

Figure 4.29: Animal data : Relative performance Direction 3

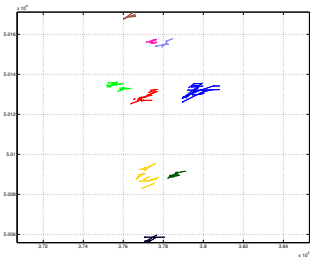


(a) Convex Hull DBSCAN, Direction 4

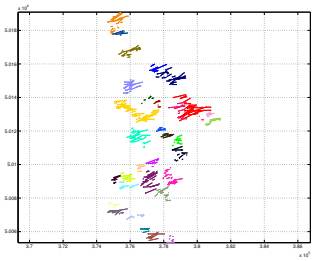


(b) MBR DBSCAN, Direction 4

Figure 4.30: Animal data : Relative performance Direction 4

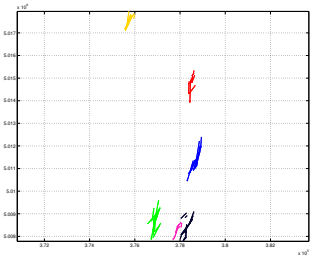


(a) Convex Hull DBSCAN, Direction 5

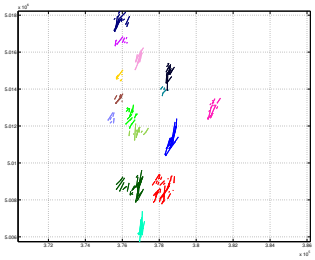


(b) MBR DBSCAN, Direction 5

Figure 4.31: Animal data : Relative performance Direction 5

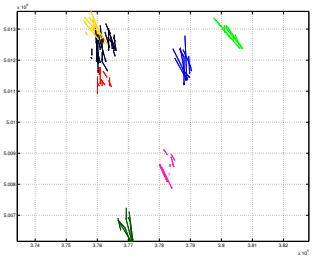


(a) Convex Hull DBSCAN, Direction 6

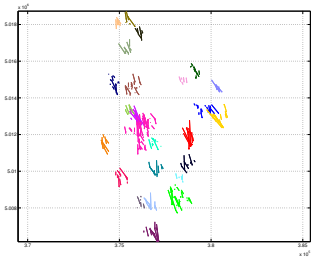


(b) MBR DBSCAN, Direction 6

Figure 4.32: Animal data : Relative performance Direction 6

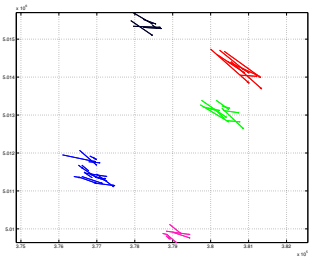


(a) Convex Hull DBSCAN, Direction 7

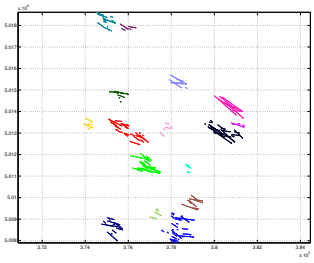


(b) MBR DBSCAN, Direction 7

Figure 4.33: Animal data : Relative performance Direction 7

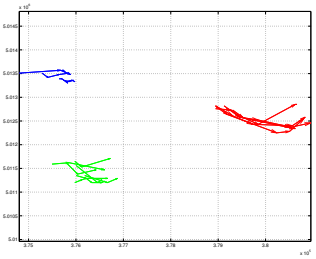


(a) Convex Hull DBSCAN, Direction 8

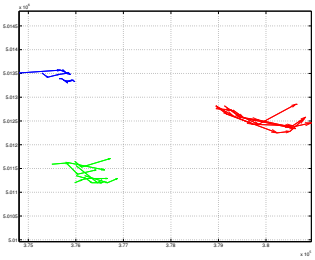


(b) MBR DBSCAN, Direction 8

Figure 4.34: Animal data : Relative performance Direction 8

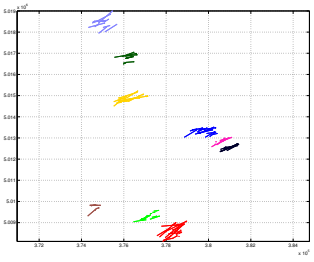


(a) Convex Hull DBSCAN, Direction 9

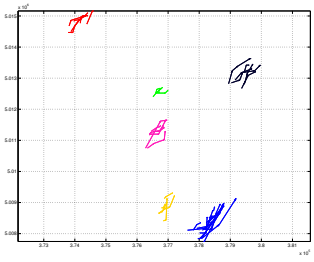


(b) MBR DBSCAN, Direction 9

Figure 4.35: Animal data : Relative performance Direction 9

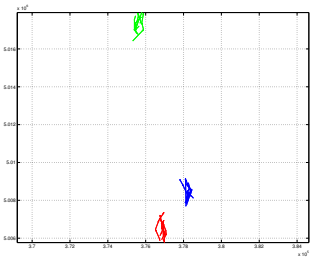


(a) Convex Hull DBSCAN, Direction 10

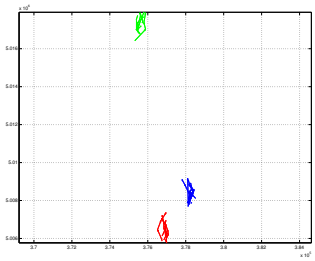


(b) MBR DBSCAN, Direction 10

Figure 4.36: Animal data : Relative performance Direction 10

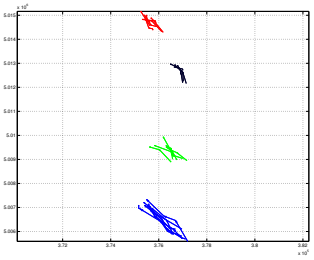


(a) Convex Hull DBSCAN, Direction 11

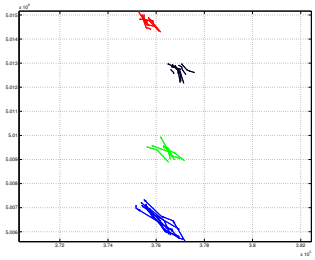


(b) MBR DBSCAN, Direction 11

Figure 4.37: Animal data : Relative performance Direction 11

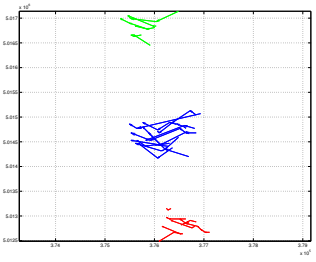


(a) Convex Hull DBSCAN, Direction 12

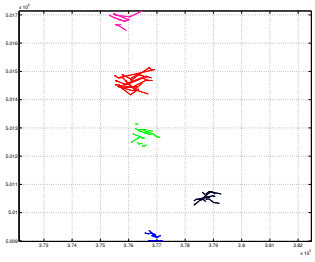


(b) MBR DBSCAN, Direction 12

Figure 4.38: Animal data : Relative performance Direction 12

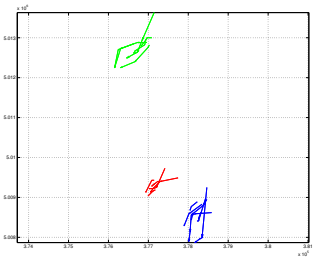


(a) Convex Hull DBSCAN, Direction 13

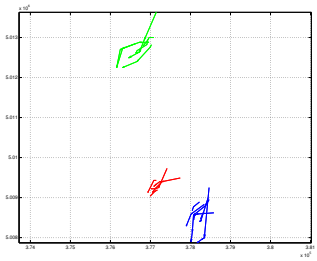


(b) MBR DBSCAN, Direction 13

Figure 4.39: Animal data : Relative performance Direction 13

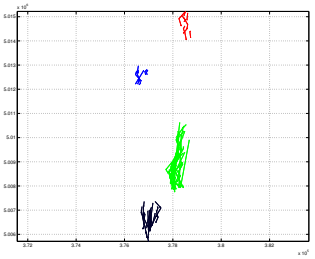


(a) Convex Hull DBSCAN, Direction 14

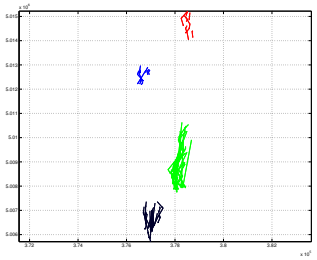


(b) MBR DBSCAN, Direction 14

Figure 4.40: Animal data : Relative performance Direction 14

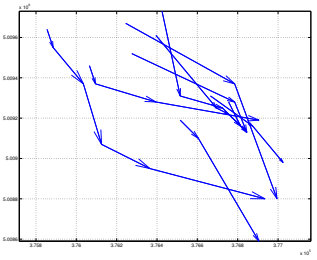


(a) Convex Hull DBSCAN, Direction 15

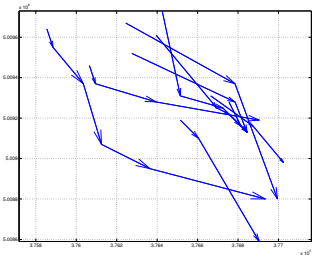


(b) MBR DBSCAN, Direction 15

Figure 4.41: Animal data : Relative performance Direction 15



(a) Convex Hull DBSCAN, Direction 16



(b) MBR DBSCAN, Direction 16

Figure 4.42: Animal data : Relative performance Direction 16

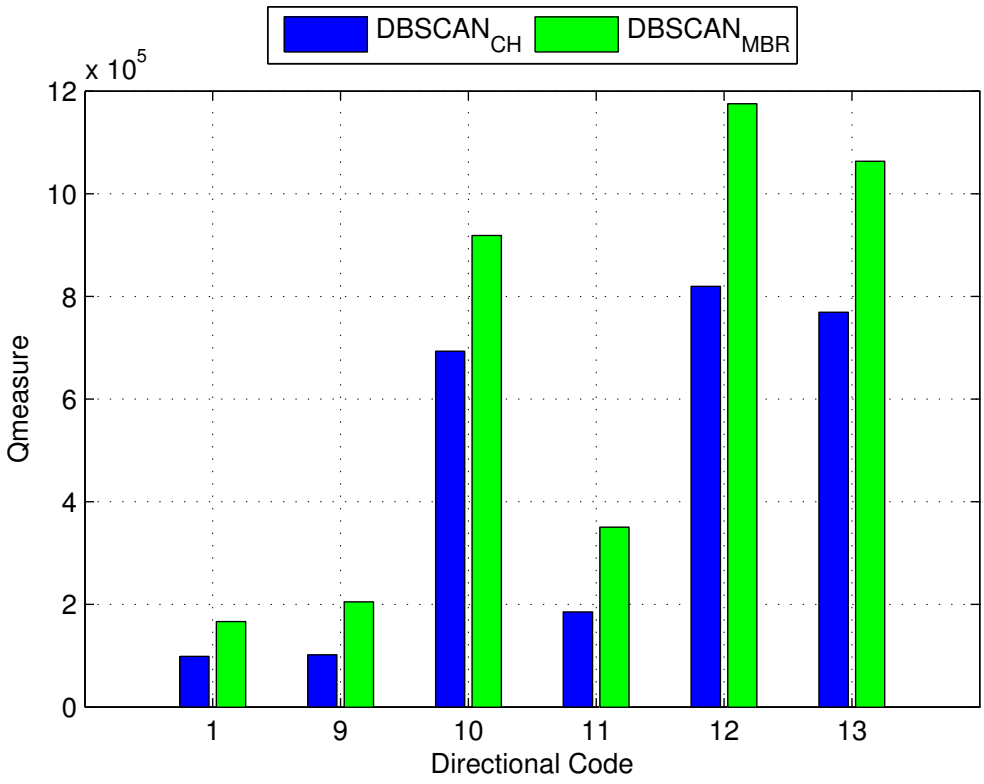


Figure 4.43: Hurricane Qmeasure

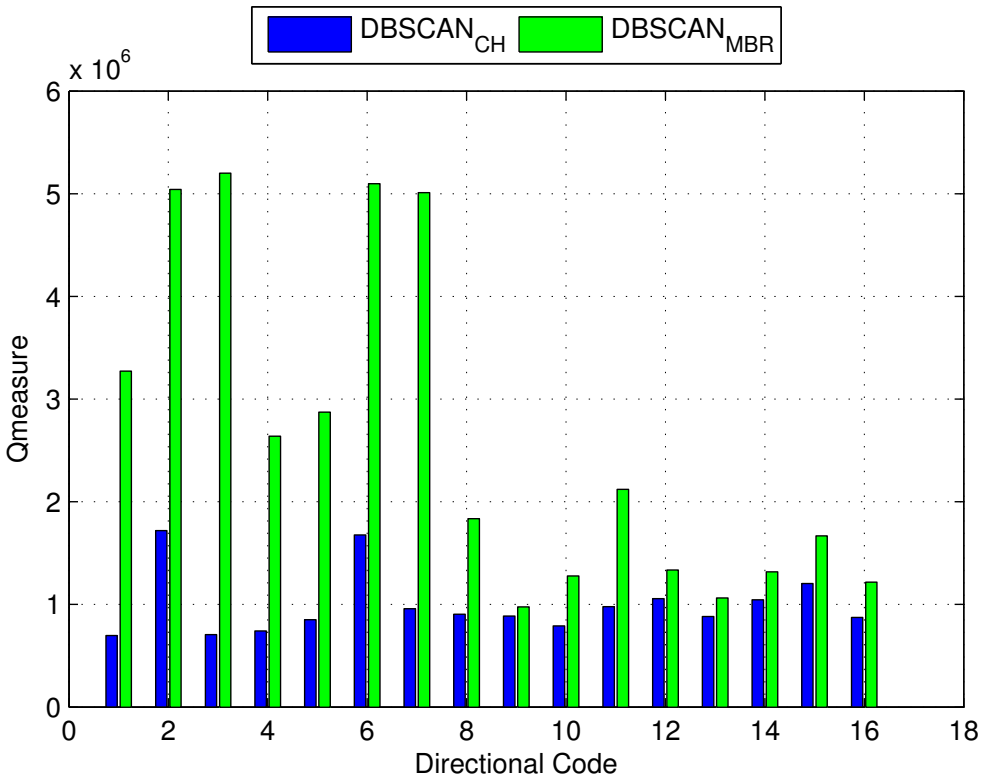


Figure 4.44: Animal Qmeasure

CHAPTER 5

Trajectory Smoothing and Simplification

5.1 Introduction

Trajectory smoothing is an important task in trajectory data analysis. A more general task, known as trajectory simplification [15–17], focuses on reducing the number of points in a trajectory; on the other hand trajectory smoothing focuses on the directional consistency of a trajectory. It is usually performed as a preprocessing task in most of the trajectory analysis. Usually, the non-smooth behavior of a trajectory can be caused by the presence of some noise in the data set, whereas in other cases it could be the characteristics of the trajectory itself. Therefore trajectory smoothing becomes imperative before any further analysis. As an example, in analyzing the global directional pattern of a trajectory, the original trajectory may have some local haphazard movements which is not relevant for the global analysis.

Trajectory simplification has been addressed in [15–17]. In these works on trajectory simplification, the authors did not address the non-smooth behavior of a trajectory caused by local sharp angular movements.

In this chapter we propose two novel trajectory smoothing methods. Proposed trajectory smoothing methods are relevant to the directional pattern analysis. The proposed smoothing algorithms are based on the internal angles of a trajectory. We identify that the non-smooth characteristics of a trajectory is mainly caused by the presence of some very small internal angles. These small angles are also responsible for a trajectory's self intersection. To the best of our knowledge this approach to smoothing a trajectory is novel and effective.

5.2 Preliminaries

Before we describe the proposed methods, we define a trajectory and its major defining characteristics. Let us consider, a trajectory data set D comprising of n trajectories viz., $D = (Tr_1, Tr_2, \ldots, Tr_n)$.

Trajectory Definition: A trajectory Tr_j of size s is a sequence of points $[p_1, p_2, \ldots, p_s]$, where p_1 is it's initial point and p_s is the final point. An i^{th} point p_i in Tr_j is associated with spatial co-ordinates (x_i, y_i) and the associated time t_i. For example, in Figure 5.1 *a*, $Tr = [p_1, p_2, p_3, p_4, p_5, p_6, p_7, p_8, p_9, p_{10}, p_{11}]$ is a trajectory of size $s = 11$. We define the size of a trajectory as the number of points that represent it.

Trajectory Segments (L_k): A trajectory Tr_j consists of line segments $L_k = \overline{p_k p_{k+1}}$ which are formed by joining the k^{th} and $(k+1)^{th}$ consecutive points in it, where $k \in [1, \ldots, s-1]$.

Angular Attribute (θ): This is an important attribute of a trajectory for directional analysis, which considers the angles between its L_k and L_{k+1} consecutive line segments. This involves three successive points: p_k, p_{k+1} and p_{k+2}.

$$\theta = min(\angle(\overline{p_k p_{k+1}}, \overline{p_{k+1} p_{k+2}}), 360 - \angle(\overline{p_k p_{k+1}}, \overline{p_{k+1} p_{k+2}})) \tag{5.1}$$

where, angle between the two line segments $\overline{p_k p_{k+1}}$ and $\overline{p_{k+1} p_{k+2}}$ is $\angle(\overline{p_k p_{k+1}}, \overline{p_{k+1} p_{k+2}})$. Angle is measured in anti-clockwise rotation from $\overline{p_k p_{k+1}}$ to $\overline{p_{k+1} p_{k+2}}$. We consider the smaller of the angles between the two line segments so that $0 \leq \theta \leq 180$. A large value of θ represents a small change in direction whereas a small value represents a sharp change. $\angle$ is the symbol for absolute value of an angle.

5.3 Trajectory Smoothing

Unlike hurricane trajectories [19], animal movement trajectories [23] tend to be very haphazard and are non-uniformly sampled. For meaningful trajectory data analysis, trajectory smoothing becomes imperative. Although, the smoothing of trajectories has been studied in [15–17], non-smooth trajectories (see Fig. 5.1) are not explicitly addressed.

5.3.1 Trajectory smoothness measure($sm\,(Tr)$):

Smoothness of a trajectory determines its directional consistency over its whole length. In this work as a smoothness measure, we consider the mean ($sm_{\mu}\,(Tr)$) as well as the standard deviation ($sm_{sd}\,(Tr)$) of the angular attributes of a trajectory. If $\overline{\Theta}$ is the vector of the angular attributes of a trajectory Tr, then : $sm_{\mu}(Tr) = mean(\overline{\Theta})$, $sm_{sd}(Tr) = sd(\overline{\Theta})$. The smooth trajectory should have better smoothness to highlight its main directional properties. If an initial trajectory is Tr, then the goal of the smoothing process is to simplify it to a trajectory Tr_s such that $sm_{\mu}(Tr) \leq sm_{\mu}(Tr_s)$ and $sm_{sd}(Tr) \geq sm_{sd}(Tr_s)$.

A large $sm_{\mu}(Tr)$ and a small $sm_{sd}(Tr)$ would indicate a smooth trajectory, whereas small values for $sm_{\mu}(Tr)$ indicate a very jagged trajectory.

We propose two algorithms for smoothing : *3-points smoothing* and *5-points smoothing* algorithms respectively. These algorithms are based on number of consecutive trajectory points that we consider for smoothing process.

5.3.2 3 - points trajectory smoothing : $3PtSpTraj$

Fig. 5.1 shows the application of $3PtSpTraj$. Here we consider 3 successive trajectory points at a time, for analyzing the smoothness of a trajectory. These three points will form an angle around the middle point. We can see from Fig. 5.1

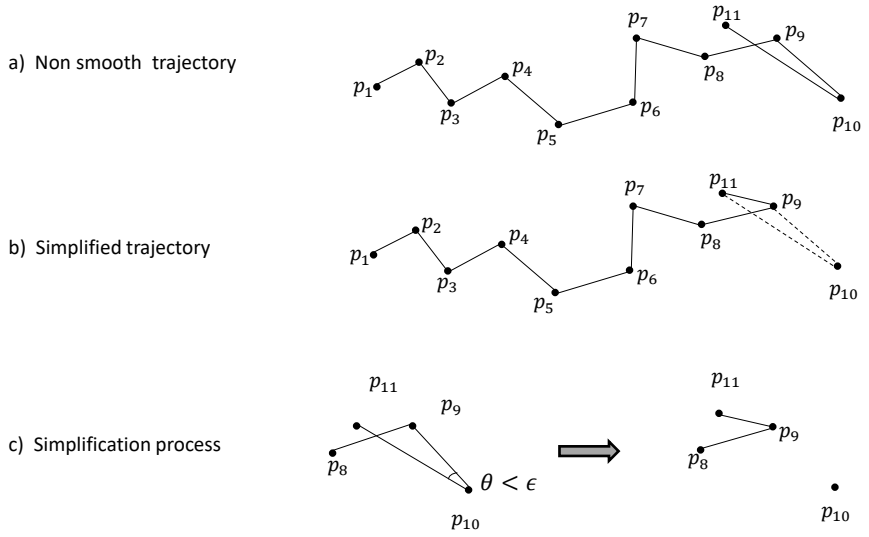


Figure 5.1: Trajectory simplification

that, the angle between the segment $\overline{p_9p_{10}}$ and $\overline{p_{10}p_{11}}$ is very small due to which the segment $\overline{p_{10}p_{11}}$ intersects the segment $\overline{p_8p_9}$. The small angles indicate sharp turns leading to non-smooth trajectories. This method needs an input parameter ϵ_θ, which is the smallest angle threshold. Therefore, if an intermediate angle is less than this threshold, (Fig. 5.1), we filter it out by discarding the intermediate point (p_{10} in Fig. 5.1). This will result in the formation of the new segment $\overline{p_9p_{11}}$. If the resulting angle is still less than ϵ, further smoothing is applied.

Algorithm 12 shows the formal steps of the trajectory smoothing method. In this algorithm first we compute the internal angles between the trajectory segments. This computation is done by the method $ComputeInternalAngles$. This method takes the input trajectory T as an input and returns a vector of angles as the output. We continue to smooth the input trajectory by dropping the smallest

Algorithm 12 : $3PtSpTraj(T, \epsilon_\theta)$ /* 3-Points trajectory smoothing */

$trajAngles = ComputeInternalAngles(T)$

$minAngle = min(trajAngles)$

while $minAngle < \epsilon_\theta$ **do**

$T = Remove(T, minAngle)$

$trajAngles = ComputeInternalAngles(T)$

$minAngle = min(trajAngles)$

end while

$return(T)$

Algorithm 13 : $5PtSpTraj(T, \epsilon_\theta)$: /* 5-points trajectory simplification */

$trajAngles = ComputeInternalAngles(T)$

$minAngle = min(trajAngles)$

while $minAngle < \epsilon_\theta$ **do**

$T_s = SubTrajectory(T, 5)$

for all T_i in T_s **do**

$T_i' = CaseSimplify(T_i)$

$T_i \leftarrow T_i'$

end for

$trajAngles = ComputeInternalAngles(T)$

$minAngle = min(trajAngles)$

end while

$return(T)$

angle $minAngle$ as long as this angle is less than threshold angle ϵ_θ. The method $Remove(T, minAngle)$ removes the angle $minAngle$ by removing the corresponding point from the trajectory T.

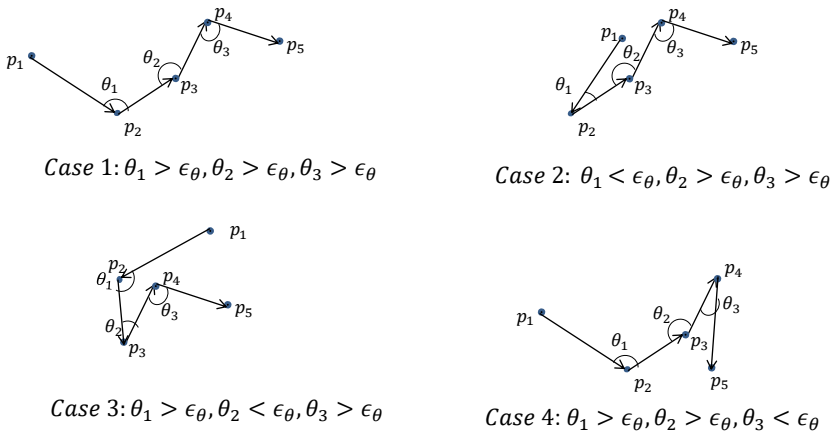


Figure 5.2: Case 1 - 4 for step simplification

5.3.3 5 - points trajectory smoothing : $5PtSpTraj$

We propose another algorithm called *5 - points smoothing* (Algorithm 13-14). We consider a window of size $w = 5$ trajectory points, hence corresponding to 3 angles. This approach is more advanced compared to the previous algorithm. In order to explain the algorithm we use the 8 cases explained in Figure 5.2 and Figure 5.3. In this algorithm we use $SubTrajectory(T, 5)$, to return sub-trajectories using a window of size 5. These sub-trajectories are in turn examined for smoothness using the 8 cases as described in Figure 5.2 and Figure 5.3. These 8 cases are based on each of the 3 angles being either $\leq \epsilon_\theta$ or $> \epsilon_{theta}$, resulting in $2^3 = 8$ cases. These individual cases are addressed by $CaseSimplify(T)$ (Algorithm 14). We can see from these figures that each case of non-smooth behavior in sub-trajectory can be addressed in corresponding way. In Figure 5.2 *Case* 1, since all the three internal angles are

Algorithm 14 : $CaseSimplify(T)$ /* Case specific simplification */

$caseInput = Find3\theta Case(T_i)$

switch $caseInput$ **do**

case 1 :

$DoNothing()$

case 2 :

$T_i \leftarrow DropPoint(T_i, p_2)$

case 3 :

$T_i \leftarrow DropPoint(T_i, p_3)$

case 4 :

$T_i \leftarrow DropPoint(T_i, p_4)$

case 5 :

$T_i \leftarrow DropPoint(T_i, p_2, p_3)$

case 6 :

$T_i \leftarrow DropPoint(T_i, p_2, p_4)$

case 7 :

$T_i \leftarrow DropPoint(T_i, p_3, p_4)$

case 8 :

$T_i \leftarrow DropPoint(T_i, p_2, p_3, p_4)$

end switch

$return(T_i)$

larger than the threshold ϵ_θ, we return the original sub-trajectory because it is already smooth. In Figure 5.2 *Case* 2, angle θ_1 is less than ϵ_θ, therefore we drop the point p_2 from the sub-trajectory. This will result in the smoothed trajectory as (p_1, p_3, p_4, p_5). Similarly in Figure 5.2 *Case* 3, we drop the trajectory point p_3 because the angle

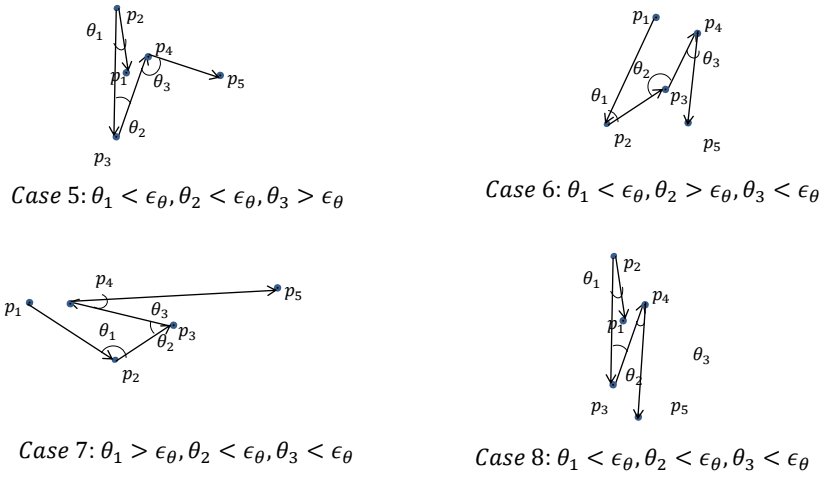


Figure 5.3: Case 5 - 8 for step simplification

$\theta_2 < \epsilon_\theta$, the resultant trajectory is (p_1, p_2, p_4, p_5). In Figure 5.2 $Case$ 4, $\theta_3 < \epsilon_\theta$ so we drop point p_4 so the resulting trajectory is (p_1, p_2, p_3, p_5). Figure 5.3 $Case$ 5, shows that the two internal angles which are θ_1 and θ_2, are less than ϵ_θ, here we discard points p_2 and p_3 which results in trajectory (p_1, p_4, p_5). Similarly in Figure 5.3 $Case$ 6, θ_1 and θ_3 are less than ϵ_θ, in this case we drop the trajectory points p_2 and p_4 resulting in the final trajectory (p_1, p_3, p_5). In Figure 5.3 $Case$ 7, θ_2 and θ_3 are less than ϵ_θ, therefore we drop the points p_3 and p_4 resulting in the final trajectory as (p_1, p_2, p_5). In the final case Figure 5.3 $Case$ 8, we have θ_1, θ_2 and θ_3 less than ϵ_θ, therefore we drop the points p_2, p_3 and p_4 resulting in the trajectory as (p_1, p_5).

$CaseSimplify$ method identifies different non-smooth cases and smooths those cases accordingly as mentioned above. It uses the method $DropPoint(T_i, p_j)$ to drop the point p_j from a sub-trajectory T_i, if the angle at p_j is less than ϵ_θ. The same

method $DropPoint(T_i, p_j, p_k, p_l)$ can be used to drop multiple points from the input sub-trajectory T_i. These multiple points are dropped to remove non-smoothness from the trajectory.

5.4 Experiments and Results

In this chapter we use the animal data [23] to show the effectiveness of our approach. This animal data contains movement data of elk, deer and cattle. We use the year 1993 data for elk, deer and cattle.

To address the unevenly sampled trajectories, we segment a bigger trajectory into sub-trajectories at a point where the time difference between the successive sampled points is larger than a given threshold (45 minutes for elk and deer movement data and 120 minutes for cattle data). The choice of different values of time threshold was picked on the basis of the nature of the data. The samples in elk and deer are in the interval of approximately 30 minutes, whereas in case of cattle it is approximately 90 minutes.

We present the results of the trajectory simplification on the three data sets (elk, deer and cattle) for original trajectories and simplified trajectories (using two methods as mentioned above) using the sm_μ and sm_{sd} smoothness measures. Table 5.1 to Table 5.11 show the results of trajectory smoothing. Each table shows the values of sm_μ and sm_{sd} for original trajectories, trajectory simplified by using 3-points and the 5-points smoothing. We present the results using different angular thresholds which are 5, 10, 15 and 30 degrees. From all these results we can see that the trajectory simplification has resulted in better values of the smoothness measure. Surprisingly, the trajectory smoothing method using 5-points smoothing has only performed marginally better than that with the 3-points smoothing. This behavior

Table 5.1: Elk trajectory data simplification $\epsilon_\theta = 5$

$OrigTraj$ (sm_μ)	$OrigTraj$ (sm_{sd})	$3PtSpTraj$ (sm_μ)	$3PtSpTraj$ (sm_{sd})	$5PtSpTraj$ (sm_μ)	$5PtSpTraj$ (sm_{sd})
94.59	55.00	99.08	52.16	99.17	52.13
89.68	55.47	95.02	51.45	95.17	51.46
85.56	55.92	93.79	52.88	93.92	51.91
89.41	58.50	97.71	56.07	97.84	56.04
83.68	56.50	89.37	53.73	89.92	54.03
91.31	54.95	97.51	52.49	97.62	51.51
97.50	56.85	102.45	50.79	102.60	50.83
90.56	56.69	98.41	54.00	98.52	54.05
87.28	52.80	92.02	50.25	92.42	49.89
86.23	56.74	91.02	55.02	91.44	54.25

Table 5.2: Elk trajectory data simplification $\epsilon_\theta = 10$

$OrigTraj$ (sm_μ)	$OrigTraj$ (sm_{sd})	$3PtSpTraj$ (sm_μ)	$3PtSpTraj$ (sm_{sd})	$5PtSpTraj$ (sm_μ)	$5PtSpTraj$ (sm_{sd})
94.59	55.00	99.17	51.85	99.25	51.82
89.68	55.47	96.54	51.08	96.69	50.09
85.56	55.92	93.71	50.96	93.84	50.99
91.31	54.95	98.97	50.85	99.42	50.09
97.50	56.85	104.60	48.58	104.74	47.64
90.56	56.69	104.16	48.81	104.28	48.85
89.99	56.24	97.92	51.16	98.08	49.25
87.28	52.80	95.11	48.72	95.51	48.35
94.96	53.53	101.67	49.81	101.74	49.82
86.23	56.74	92.41	52.12	92.80	52.20

is due to the fact that the 3-points smoothing method already removes most of the non-smooth characteristics of a trajectory.

5.5 Summary

In this chapter we propose two novel algorithms for the trajectory smoothing problem. These algorithms consider the internal angles of the trajectories for their non-smooth characteristics. We show the effectiveness of these algorithms using the

Table 5.3: Elk trajectory data simplification $\epsilon_\theta = 15$

$OrigTraj$ (sm_μ)	$OrigTraj$ (sm_{sd})	$3PtSpTraj$ (sm_μ)	$3PtSpTraj$ (sm_{sd})	$5PtSpTraj$ (sm_μ)	$5PtSpTraj$ (sm_{sd})
89.68	55.47	100.93	48.53	101.12	48.47
83.68	56.50	96.42	48.84	96.99	48.07
97.50	56.85	108.07	48.61	108.31	48.68
86.70	53.37	99.58	47.01	99.93	46.32
85.58	54.57	97.67	48.44	97.85	47.63
90.56	56.69	106.04	47.87	106.33	46.02
87.28	52.80	95.30	47.41	95.70	47.03
94.96	53.53	105.77	46.55	105.82	45.64
86.62	54.72	96.90	48.94	97.28	47.99
88.30	59.80	106.84	50.87	107.37	50.37

Table 5.4: Elk trajectory data simplification $\epsilon_\theta = 30$

$OrigTraj$ (sm_μ)	$OrigTraj$ (sm_{sd})	$3PtSpTraj$ (sm_μ)	$3PtSpTraj$ (sm_{sd})	$5PtSpTraj$ (sm_μ)	$5PtSpTraj$ (sm_{sd})
90.95	54.75	106.33	45.10	106.77	43.36
84.97	53.41	107.63	45.34	109.80	42.34
83.68	56.50	102.83	44.84	104.95	44.29
91.31	54.95	112.06	42.13	112.64	41.85
86.70	53.37	106.65	42.94	106.92	42.14
85.58	54.57	109.41	42.26	110.52	41.38
89.99	56.24	110.57	42.65	111.12	42.48
87.28	52.80	104.88	40.87	104.98	40.64
86.62	54.72	107.70	44.43	108.17	44.31
85.13	57.73	108.12	43.10	108.41	42.91

Table 5.5: Deer trajectory data simplification $\epsilon_\theta = 5$

$OrigTraj$ (sm_μ)	$OrigTraj$ (sm_{sd})	$3PtSpTraj$ (sm_μ)	$3PtSpTraj$ (sm_{sd})	$5PtSpTraj$ (sm_μ)	$5PtSpTraj$ (sm_{sd})
86.56	56.46	93.93	52.74	94.03	51.81
65.01	53.27	69.67	50.64	70.37	50.15
71.64	55.97	75.89	54.20	75.89	54.20

Table 5.6: Deer trajectory data simplification $\epsilon_\theta = 10$

$OrigTraj$ (sm_μ)	$OrigTraj$ (sm_{sd})	$3PtSpTraj$ (sm_μ)	$3PtSpTraj$ (sm_{sd})	$5PtSpTraj$ (sm_μ)	$5PtSpTraj$ (sm_{sd})
86.56	56.46	97.82	49.87	98.00	48.99
65.01	53.27	75.01	49.08	75.06	49.30
77.78	56.12	86.85	51.51	87.12	50.57

Table 5.7: Deer trajectory data simplification $\epsilon_\theta = 30$

$OrigTraj$ (sm_μ)	$OrigTraj$ (sm_{sd})	$3PtSpTraj$ (sm_μ)	$3PtSpTraj$ (sm_{sd})	$5PtSpTraj$ (sm_μ)	$5PtSpTraj$ (sm_{sd})
91.89	56.54	116.89	31.58	117.24	31.58
65.01	53.27	100.82	38.03	102.74	38.11
84.56	53.61	105.20	44.00	105.66	44.25

Table 5.8: Cattle trajectory data simplification $\epsilon_\theta = 5$

$OrigTraj$ (sm_μ)	$OrigTraj$ (sm_{sd})	$3PtSpTraj$ (sm_μ)	$3PtSpTraj$ (sm_{sd})	$5PtSpTraj$ (sm_μ)	$5PtSpTraj$ (sm_{sd})
75.25	50.72	81.99	49.49	82.06	48.52
76.75	53.05	82.02	47.97	82.27	47.07
77.99	51.56	82.54	49.21	82.69	48.76
82.84	52.66	85.56	51.61	85.78	51.73
83.10	55.65	86.97	54.66	87.66	54.66
74.12	57.19	82.24	52.50	82.85	50.61
81.96	56.90	90.77	52.97	91.05	52.08
78.43	54.69	82.11	51.21	82.77	50.71

Table 5.9: Cattle trajectory data simplification $\epsilon_\theta = 10$

$OrigTraj$ (sm_μ)	$OrigTraj$ (sm_{sd})	$3PtSpTraj$ (sm_μ)	$3PtSpTraj$ (sm_{sd})	$5PtSpTraj$ (sm_μ)	$5PtSpTraj$ (sm_{sd})
82.25	58.06	91.33	54.37	91.46	53.58
84.62	57.62	93.31	53.40	93.64	52.54
84.73	55.06	92.42	51.81	92.68	51.84
75.25	50.72	85.52	48.51	85.60	47.55
82.84	52.66	88.65	50.72	88.87	50.84
74.12	57.19	84.59	52.82	85.20	51.93
74.34	55.57	84.84	51.62	85.79	51.61
81.59	58.54	95.28	53.25	95.49	53.31

Table 5.10: Cattle trajectory data simplification $\epsilon_\theta = 15$

$OrigTraj$ (sm_μ)	$OrigTraj$ (sm_{sd})	$3PtSpTraj$ (sm_μ)	$3PtSpTraj$ (sm_{sd})	$5PtSpTraj$ (sm_μ)	$5PtSpTraj$ (sm_{sd})
79.11	52.06	93.79	46.69	94.56	45.81
82.25	58.06	95.08	49.81	95.20	49.03
81.60	60.35	100.30	51.82	100.37	50.86
75.25	50.72	86.40	44.70	87.43	44.13
76.75	53.05	86.13	43.41	86.52	43.21
74.12	57.19	88.46	51.16	89.07	50.28
74.34	55.57	84.77	48.64	85.72	48.63
82.65	54.12	91.10	50.25	91.70	50.23

Table 5.11: Cattle trajectory data simplification $\epsilon_\theta = 30$

$OrigTraj$ (sm_μ)	$OrigTraj$ (sm_{sd})	$3PtSpTraj$ (sm_μ)	$3PtSpTraj$ (sm_{sd})	$5PtSpTraj$ (sm_μ)	$5PtSpTraj$ (sm_{sd})
79.11	52.06	101.58	43.03	102.80	42.40
78.87	50.22	84.35	39.64	85.12	38.71
79.05	53.35	103.16	42.39	103.28	42.31
75.25	50.72	93.30	44.93	97.08	44.93
74.34	55.57	104.48	45.29	105.92	45.04
82.65	54.12	99.10	43.83	100.12	43.40
79.04	57.07	87.45	43.49	88.01	43.64
81.96	56.90	100.58	47.01	100.64	46.77

smoothness performance measure on animal movement data. The results show the effectiveness of the proposed algorithms.

CHAPTER 6

Conclusion

This dissertation is focused on three aspects of spatio-temporal data analysis: point based clustering for hot-spot detection; directional analysis of trajectory; and smoothing of trajectories.

In the point based clustering for hot-spot detection, we analyzed hurricane storm trajectory data to find areas of extreme hurricane density, as well as areas where hurricanes originate and land. We took a different approach to trajectory analysis by focusing on points along the trajectory, rather than line segments as in previous work. We used the DBSCAN algorithm for the clustering analysis. Initially the clusters are obtained on the basis of only the spatial attributes. After that, we looked at the influence of the non-spatial attributes, in particular wind speed, on the clusters obtained. We also propose a spatio temporal DBSCAN algorithm where the normalized relative time information with the point data has been considered as another non-spatial attribute for DBSCAN. We post processed the clustering results to obtain the storm starting, storm landing and storm tracking information. Our work differs from other work because of the focus on trajectory points, which results in identifying high-activity regions, as well as regions at start and end of storms.

In the directional analysis of trajectory, we proposed a novel framework for the directional analysis of trajectory data. We proposed a novel CH based filtration method using convex hulls of sub-trajectories. A novel technique for identifying the directional orientation of the trajectory data was proposed. Clustering is then applied to determine interesting directional clusters of trajectories. We demonstrate

our approach on two data sets: hurricane tracks and animal trajectories. the results show that our clustering is good using standard cluster evaluation metric.

Finally, in the work on smoothing of trajectories, we propose two novel trajectory smoothing methods which are based on the internal angles of trajectories. These two methods are called *3-points smoothing* and *5-points smoothing*. The effectiveness of these methods have been established on three real-life datasets.

REFERENCES

[1] P. K. Tripathi, M. Debnath, and R. Elmasri:, "Extracting dense regions from hurricane trajectory data," in *GeoRich'14, Proceedings of Workshop on Managing and Mining Enriched Geo-Spatial Data*, 2014.

[2] ——, "A direction based framework for trajectory data analysis," in *International Conference on PErvasive Technologies Related to Assistive Environments*, ser. PETRA '16, 2016.

[3] M. Vlachos, G. Kollios, and D. Gunopulos, "Discovering similar multidimensional trajectories," in *Data Engineering, 2002. Proceedings. 18th International Conference on*, 2002, pp. 673–684.

[4] D. Birant and A. Kut, "ST-DBSCAN: An algorithm for clustering spatial-temporal data," *Data Knowl. Eng.*, vol. 60, no. 1, pp. 208–221, Jan. 2007. [Online]. Available: http://dx.doi.org/10.1016/j.datak.2006.01.013

[5] J.-G. Lee and J. Han, "Trajectory clustering: A partition-and-group framework," in *In SIGMOD*, 2007, pp. 593–604.

[6] J. Gudmundsson, A. Thom, and J. Vahrenhold, "Of motifs and goals: mining trajectory data," in *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, ser. SIGSPATIAL '12. New York, NY, USA: ACM, 2012, pp. 129–138. [Online]. Available: http://doi.acm.org/10.1145/2424321.2424339

[7] M. Vieira, P. Bakalov, and V. Tsotras, "On-line discovery of flock patterns in spatio-temporal data," 2009.

[8] C.-S. Chen, C. F. Eick, and N. Rizk, "Mining spatial trajectories using non-parametric density functions," in *Machine Learning and Data Mining in Pattern Recognition*, ser. Lecture Notes in Computer Science, P. Perner, Ed. Springer Berlin Heidelberg, 2011, vol. 6871, pp. 496–510. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-23199-5_37

[9] J.-G. Lee, J. Han, X. Li, and H. Gonzalez, "Traclass: Trajectory classification using hierarchical region-based and trajectory-based clustering," *Proc. VLDB Endow.*, vol. 1, no. 1, pp. 1081–1094, Aug. 2008. [Online]. Available: http://dl.acm.org/citation.cfm?id=1453856.1453972

[10] Y. Huang, L. Zhang, and P. Zhang, "A framework for mining sequential patterns from spatio-temporal event data sets," *IEEE Transactions on Knowledge and Data Engineering*, vol. 20, no. 4, pp. 433–448, 2008.

[11] H. Jeung, M. L. Yiu, X. Zhou, C. S. Jensen, and H. T. Shen, "Discovery of convoys in trajectory databases," *Proc. VLDB Endow.*, vol. 1, no. 1, pp. 1068–1080, Aug. 2008. [Online]. Available: http://dl.acm.org/citation.cfm?id=1453856.1453971

[12] D. Patel, C. Sheng, W. Hsu, and M. L. Lee, "Incorporating duration information for trajectory classification," in *Proceedings of the 2012 IEEE 28th International Conference on Data Engineering*, ser. ICDE '12, 2012, pp. 1132–1143.

[13] M. Ester, H. peter Kriegel, J. S, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise." AAAI Press, 1996, pp. 226–231.

[14] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghavan, "Automatic subspace clustering of high dimensional data for data mining applications," in *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '98. New York, NY, USA: ACM, 1998, pp. 94–105.

[15] C. Long, R. C.-W. Wong, and H. V. Jagadish, "Direction-preserving trajectory simplification," *Proc. VLDB Endow.*, vol. 6, no. 10, pp. 949–960, Aug. 2013.

[16] D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 10, no. 2, pp. 112–122, 1973.

[17] M. Potamias, K. Patroumpas, and T. K. Sellis, "Sampling trajectory streams with spatiotemporal criteria." in *SSDBM*. IEEE Computer Society, 2006, pp. 275–284.

[18] *Data Mining: Concepts and Techniques, 2nd ed.* Morgan Kaufmann, 2006.

[19] http://weather.unisys.com/hurrricane/atlantic/index.html.

[20] M. Debnath, P. K. Tripathi, and R. Elmasri:, "A novel approach to trajectory analysis using string matching and clustering," in *ICDM Workshops*, 2013, pp. 986–993.

[21] S. Brakatsoulas, D. Pfoser, and N. Tryfona, "Modeling, storing, and mining moving object databases," in *Proceedings of the International Database Engineering and Applications Symposium*, ser. IDEAS '04. Washington, DC, USA: IEEE Computer Society, 2004, pp. 68–77.

[22] S. Brakatsoulas, D. Pfoser, R. Salas, and C. Wenk, "On map-matching vehicle tracking data," in *Proceedings of the 31st International Conference on Very Large Data Bases*, ser. VLDB '05. VLDB Endowment, 2005, pp. 853–864.

[23] http://www.fs.fed.us/pnw/starkey/data/tables/.

[24] T. H. Cormen, C. Stein, R. L. Rivest, and C. E. Leiserson, *Introduction to Algorithms*. McGraw-Hill Higher Education, 2001.

BIOGRAPHICAL STATEMENT

Praveen K. Trpathi was born in New Delhi, India, in 1975. He received his B.S. degree in Computer Science from G B Pant University of Agriculture and Technology, Pantnagar, India, in 1997. He did his M.S. degree in Computer Science from JNU New Delhi, India in 2002. He worked as a research fellow with Machine Intelligence Unit in Indian Statistical Institute Kolkata from 2003-2007. From 2007-2009 he worked as a System Analyst in Applied Research Group, Satyam Computers Limited in Bangalore India. He also worked as Senior Lecturer in department of Computer Science at Jaypee University of Information Technology, Solan, India from 2010-2011. He studied for his PhD in Computer Science at The University of Texas at Arlington from 2011 to 2016. During his PhD work; he worked as a Data Science intern at StumbleUpon Inc. San Francisco in Summer 2014, and as a Research Intern at Philips Research North America, New York in Summer 2015.