MARITIME TRAFFIC ANOMALY DETECTION FROM AIS SATELLITE DATA IN NEAR PORT REGIONS

by

Bo Liu

Submitted in partial fulfillment of the
requirements for the degree of
Master of Computer Science

at

Dalhousie University
Halifax, Nova Scotia
August 2015

Table of Contents

List of Tables

List of Figures

Abstract

Maritime traffic monitoring is an important aspect of safety and security in close to port operations. While there is a large amount of data with variable quality, decision makers need reliable information about possible situations or threats. In this thesis, we propose a two-component maritime traffic anomaly detection model. First, it extracts normal ship trajectory patterns using, besides ship tracing data, the publicly available IMO Rules. The main result of clustering is a set of generated lanes that can be mapped to those defined in the IMO directives. Then, we show how the second anomaly detection component detects anomalous navigational behaviors based on three specialized division distances with the clusters. It decides for each point if the vessel is anomalous, considering longitude, latitude, direction and speed. This point-based approach is applicable for real-time AIS surveillance; it is also feasible for analysts to set their own threshold for labeling whole trajectories.

List of Abbreviation and Symbols Used

$N_\epsilon(p)$	Eps-neighborhood of a point p
ADD	Absolute Division Distance
AIS	Automaitc Identification System
CDD	Cosine Division Distance
COG	Course Over Ground
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DBSCANSD	Density-Based Spatial Clustering of Applications with Noise considering Speed and Direction
EM	Expectation Maximization
GPS	Global Positioning System
GV	Gravity Vector
IMO	Iternational Maritime Organization
MMSI	Maritime Mobile Service Identity
RDD	Relative Division Distance
SOG	Speed Over Ground
SSP	Sampled Stopping Point
TSS	Traffic Separation Scheme Boundaries
UTC	Coordinated Universal Time

Acknowledgements

I would like to first express my deepest appreciation to my supervisor, Dr.Stan Matwin, for his support, guidance, encouragement and patience throughout my master program. His valuable suggestions and directions helped to improve the quality of this thesis. He also provides me with precious opportunities to work with multiple projects which can help me open my view.

Besides my supervisor, I would like to thank the rest of my thesis committee: Prof.Raza Abidi, Prof.Ronald Pelot and Prof.Dirk Arnold for their encouragement, insightful comments, and hard questions.

My sincere thanks also goes to Dr.Erico N.de Souza, my project leader. His advices and ideas guided me work on the right paths of the projects. He has also spent a lot time reviewing my papers and this thesis, which is a great help to me.

My parents, Baolong and Changling, who have always been supporting me and have never said no to my any decisions. Without them, this thesis cannot be finished.

I would also like to extend my gratefulness to all my friends and labmates in the Big Data Institute: Dr.Rob Warren, Dr.Xiaoguang Wang, Ahmad, Behrouz, Diana, Habibeh, Xuan, Eman, Baifan, David, Lulu and Vineeth for the stimulating discussions and for all the fun we have had in the past two years.

Last but not least, I would like to acknowledge the generous support of GSTS, Inc., ExactEarth, Inc., Marine Environmental, Observation, Prediction And Response Network (MEOPAR), and the Natural Science and Engineering Research Council of Canada (NSERC).

Chapter 1

Introduction

Today maritime transportation represents 90% of international trade volume [57] and there are more than 50,000 vessels sailing the ocean every day [31]. So the challenges related to safety and security aspects in the maritime domain are of high priority. Modeling ship behavior helps improve security for authorities and one crucial task of maritime surveillance is abnormal vessel behaviours detection. Anomalies in the maritime domain, such as unexpected stops, deviations from regulated routes, inconsistent speed or direction etc., may be related to risks like collisions[1], grounding[2], sea drunkenness[3], smuggling, piracy[4], etc. To address the issue, different data sources such as data from Automatic Identification System (AIS), synthetic aperture radar, high frequency surface wave radars, infra-red sensors, videos and intelligent reports [26] are used by the maritime authorities. Data generated by AIS are utilized in this thesis.

Automatic Identification System (AIS) is an automatic tracking system to identify and locate vessels by exchanging data with other nearby ships, AIS base stations and satellites. The use of this system has been required by the International Maritime Organization (IMO) since 2004 to enhance safety and efficiency of navigation and improve situational awareness and assessment [21]. IMO was established in Geneva in 1948 with an initial name called Inter-Governmental Maritime Consultative Organization (IMCO). Then in 1982, it was changed to International Maritime Organization [5]. The IMO's headquarter is in London, United Kingdom and it is a specialised agency of the United Nations with 171 Member States and three Associate Members. IMO is

[1]http://gcaptain.com/tag/ship-collision/
[2]http://www.wisegeek.com/what-is-ship-grounding.htm
[3]http://www.theglobeandmail.com/news/national/canadian-navy-officer-charged-with-drunkenness-disobeying-orders/article23023966/
[4]http://www.maritime-executive.com/article/pirates-attack-two-vessels-in-asia
[5]https://en.wikipedia.org/wiki/International_Maritime_Organization

the global standard-setting authority for the safety, security and environmental performance of international shipping [6]. According to the regulations defined by IMO, all ships of 300 gross tonnage and upwards engaged on international voyages and cargo ships of 500 gross tonnage and upwards not engaged on international voyages and passenger ships irrespective of size shall be fitted with an automatic identification system and over 400,000 ships worldwide have installed these transponders [6].

AIS messages are automatically broadcasted with a reporting frequency directly proportional to the speed of the vessel [44]. The data can certainly be viewed as big data: a stream of some 80 million messages are generated per day. Two categories of data (static and dynamic information) are transmitted by the AIS transponders. Static information, which must be programmed into the AIS transceiver, includes IMO number, Maritime Mobile Service Identity (MMSI) number, vessel call sign, vessel type, vessel dimensions (length and beam). The dynamic information, which needs to be updated automatically, includes the ship's location (Longitude and Latitude), Course Over Ground (COG), Speed Over Ground (SOG), heading, Rate of Turn (ROT), time in UTC (Coordinated Universal Time), navigational status etc [2]. In this thesis, both static and dynamic information is used to tackle the maritime anomaly detection task.

IMO also controls the navigational lanes that vessels must use when approaching some ports around the world. Specifically, the rules which regulate the general directions in the specific regions for the vessels to navigate are called Traffic Separation Schemes (TSS) [1]. The regulations to establish TSSs mandatory were first proposed by IMO's Maritime Safety Committee meeting in March 1971 and this recommendation was adopted by the IMO later the same year. The first corresponding mandatory traffic scheme is the Dover Straits scheme.[7] These navigational lanes are definitely good resources for maritime surveillance analysis, but unfortunately, current tools do not offer automatic ways to check whether the vessels are obeying these lanes.

Although TSS regulates the normal geographical lanes for ships to sail, it does not provide exact speed regulations in specific regions. More specifically, there is no rules to determine the maximum or minimum speed of a particular type of vessels to sail in one area. As a consequence, only checking if the ships are obeying the lanes

[6]http://www.imo.org/en/About/Pages/Default.aspx

[7]http://www.imo.org/OurWork/Safety/Navigation/Pages/ShipsRouteing.aspx

cannot satisfy the demand of maritime surveillance. Fortunately, one dynamic type of information, SOG (Speed Over Ground), is included in AIS messages and approaches that are able to utilize this historical speed information for anomaly detection are desirable.

1.1 Aim and Objectives

Based on the above introduction to the background of maritime traffic anomaly detection, the overall research aim of this thesis can be concluded as: *Review and analyse existing anomaly detection algorithms for trajectory data, and propose new or modified approaches that are well-suited for the task.*

In order to address the aim, the following objectives are identified:

1). Investigate and survey the related approaches proposed for anomaly detection in trajectory data.

2). Propose new algorithms fused with IMO rules to tackle anomaly detection task in maritime domain.

3). Demonstrate the effectiveness of the proposed algorithms on real world AIS data sets.

1.2 Research Methodology

To address the aim and objectives stated in Section 1.1, we propose a clustering-based maritime traffic anomaly detection framework. An overview of the entire framework is shown in Figure 1.1. The approach includes two components: the normal traffic patterns extraction model and the anomaly detection model. As can been seen from Figure 1.1, historical AIS data is first sent to the normal patterns extraction model which generates, as output, a set of Gravity Vectors (GV) and Sampled Stopping Points (SSP) via its two sub-components. Afterwards, when new ship trajectory data is to be judged, the next phase model will be applied based on the normal traffic patterns results to decide the new data's abnormality.

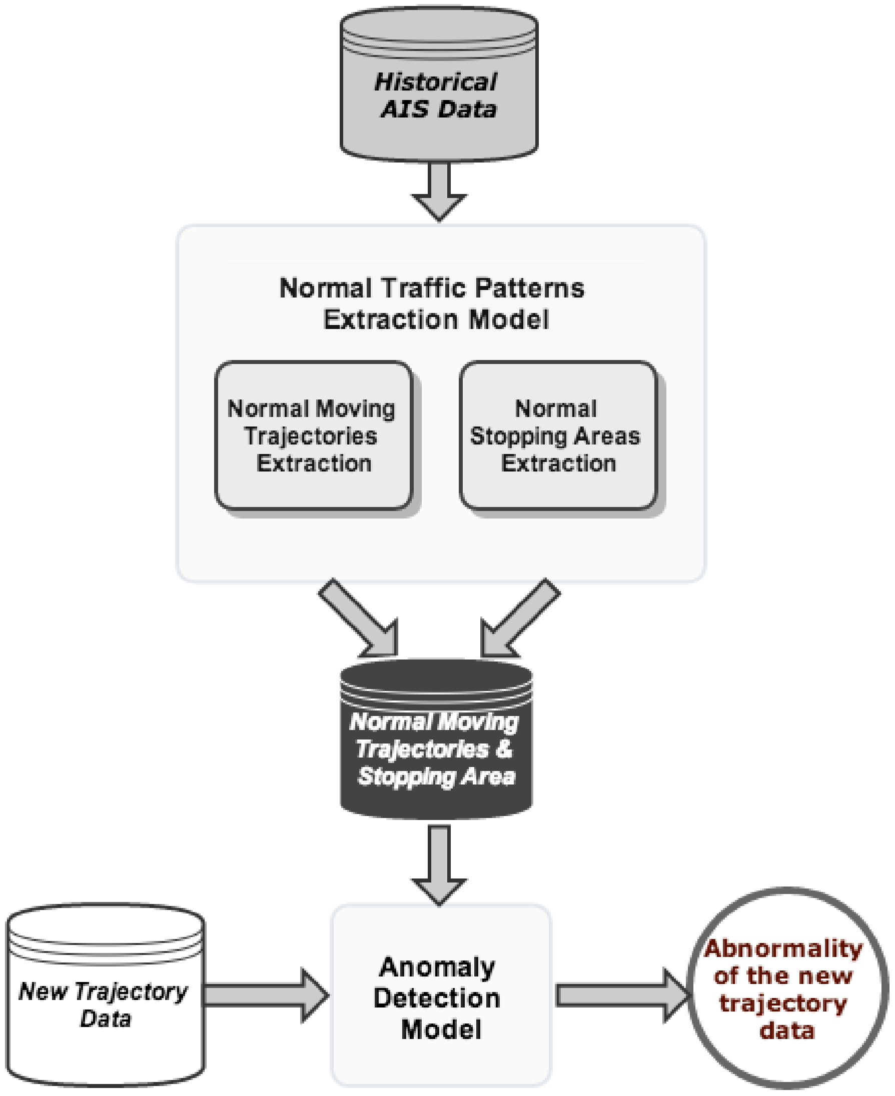

Historical AIS Data
Normal Traffic Patterns Extraction Model
Normal Moving Trajectories Extraction
Normal Stopping Areas Extraction
Normal Moving Trajectories & Stopping Area
New Trajectory Data
Anomaly Detection Model
Abnormality of the new trajectory data

Figure 1.1: The framework for Maritime Anomaly Detection. This can be roughly regarded as a two-step procedure. First, the normal traffic extraction model extracts the normal traffic patterns from the historical AIS data repository. Then the extracted Gravity Vectors (GVs) and Sampled Stopping Points (SSPs) are employed by the anomaly detection model to decide the abnormality of the new given trajectory.

The proposed clustering-based normal traffic patterns extraction model first divides the AIS data into moving and stopping parts, respectively, based on a stopping SOG (Speed Over Ground) threshold of 0.5 knots.

For the case where SOG is not less than 0.5 knots, we propose DBSCANSD (Density-Based Spatial Clustering of Applications with Noise considering Speed and

Direction) as the basic algorithm to detect the main traffic lanes within the data. DBSCANSD is based on DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [15] algorithm, modified to consider that for each point of a cluster, the neighborhood of a given radius has to contain at least a minimum number of points with similar SOGs (Speed Over Ground) and COGs (Course Over Ground). The experiments demonstrate that the clusters can reflect normal patterns of the vessels. However, it is not feasible to employ all the points of the clusters to decide the new coming trajectory data's abnormality. As a result, a representation called Gravity Vector (GV) is proposed. A GV is a vector composed of 5 features: average COG, average SOG, average Latitude, average Longitude and Median Distance. Thus, the output of the moving pattern extraction model can be a set of GVs which carries information about speed, direction, location and even the relative width of the lanes. More details about the process can be found in Section 3.2.

For the case where the SOG is less than 0.5 knots (stop areas), the original DBSCAN algorithm [15] is executed because speed (SOG) and direction (COG) are not important factors. Another type of vector is created as output, called Sampled Stopping Point (SSP), which is dependent only on the geographic shape of the region. The experiments show that a small number of SSPs can represent a stopping area well in terms of the shape of the area. More details about the process can be found in Section 3.3.

Once the normal patterns are extracted from the historical AIS data, the second phase is the anomaly detection model (presented in Chapter 4), can be applied for the new arriving data. In this process, three specialized distances including Absolute Division Distance (ADD), Relative Division Distance (RDD) and Cosine Division Distance (CDD) are proposed to decide the abnormality of one specific data point. Specifically, ADD is employed in the stopping points abnormality detection phase while RDD and CDD are used for the moving part. Besides, one anomaly detection algorithm using the three specialized division distances is proposed in Section 4.2. Although the approach is point-based, which is applicable for real-time AIS surveillance, it is also flexible enough for analysts to set their own threshold for labeling whole trajectories.

1.3 Scientific Contribution

This thesis is based on previously published work by the author. The published work has also been updated and extended with new theoretical and empirical results in this thesis. The main contributions of this thesis are listed as following:

- Proposal of the algorithm of DBSCANSD (Density-Based Spatial Clustering of Applications with Noise considering Speed and Direction), a novel point-based trajectory clustering algorithm which is applicable for various domains.
- Proposal of a normal maritime traffic extraction model based on $stops-and-moves$ approach [51].
- Proposal of a normal maritime traffic extraction model associating with the IMO rules.
- Proposal of two methods to represent the results of clustering, which are the Gravity Vector (GV) for moving clusters and the Sampled Stopping Point (SSP) for stopping clusters.
- Proposal of three specialized division distances measures for maritime anomaly detection.
- Implementation of the proposed maritime anomaly detection framework.
- Evaluation of the proposed maritime anomaly detection framework's effectiveness using both visualization and quantitative approaches.

1.4 Publications

In the following of this section, the publications published during the author's master study are listed. The first two publications are highly relevant to the thesis while the other two are less relevant to the thesis. Publication 2) first introduces the normal traffic extraction model and has been extended with more detailed explanation in Chapter 3. The anomaly detection model proposed in Chapter 4 is extended from the Publication 1).

1) Bo Liu, Erico N.de Souza, Cassey Hilliard and Stan Matwin, **Ship Movement Anomaly Detection Using Specialized Distance Measures**, IEEE International Conference on Information Fusion (FUSION 2015), 6-9 July 2015, Washington, DC, USA. (to appear)

2) Bo Liu, Erico N.de Souza, Stan Matwin and Marcin Sydow, **Knowledge-based Clustering of Ship Trajectories Using Density-based Approach**, Proceedings of IEEE International Conference on Big Data (IEEE BigData 2014), 27-30 October 2014, Washington, DC, USA. pp. 603-608.

3) Xiaoguang Wang, Xuan Liu, Bo Liu, Stan Matwin and Erico N.de Souza, **Vessel Route Anomaly Detection with Hadoop MapReduce**, Proceedings of IEEE International Conference on Big Data (IEEE BigData 2014), 27-30 October 2014, Washington, DC, USA. pp. 25-30.

4) Robert Warren and Bo Liu, **Language, Cultural Influences and Intelligence in Historical Gazetteers of the Great War**, Proceedings of IEEE International Conference on Big Data (IEEE BigData 2014), 27-30 October 2014, Washington, DC, USA. pp. 70-72.

1.5 Thesis Outline

The remainder of the dissertation is organized as follows. Chapter 2 presents the related work. It is divided into two parts The first part (Section 2.1) is about trajectory clustering in general and ship trajectory clustering in the maritime domain in particular. Section 2.2 is an overview and survey on anomaly detection techniques used for maritime security.

We present our clustering-based maritime anomaly detection framework in Chapter 3 and Chapter 4. In Chapter 3, we propose our maritime traffic patterns extraction model (Section 3.2 and Section 3.3) after giving the definition of the trajectory in maritime domain (Section 3.1). As we treat the ship trajectories as two different types based on a speed threshold, moving and stopping, which are explained in more details, Section 3.2 and Section 3.3 propose the normal moving patterns extraction model and normal stopping areas extraction model, respectively. Two different representative methods, Gravity Vector (GV) and Sampled Stopping Point (SSP) that represent the moving clustering results and stopping clustering results, are introduced accordingly in the two sections. Chapter 4 presents the anomaly detection model. As our method is a distance-based model, three division distances are first presented in Section 4.1 and then our abnormal detection algorithm is introduced in Section 4.2.

To evaluate the effectiveness of our framework, experiments are done in Chapter 5. First, in Section 5.1, to evaluate the normal traffic patterns extraction model, regions of Juan de Fuca Strait and Los Angeles Long Beach are selected and the results are presented in Section 5.1.1 and Section 5.1.2. Then in Section 5.2, we conducted another two experiments in the same region of Juan de Fuca Strait. The first one is conducted with the non-labeled data while the second one is done after labelling the data. The results of the first experiment (Section 5.2.1) are shown visually and the second experiment (Section 5.2.2) compares our model's results with the labels by the expert.

Finally, in Chapter 6 we conclude with a summary and discuss our method's limitations and the potential directions for future work.

Chapter 2

Related Work

The problem examined in this thesis can be roughly described as maritime data anomaly detection, the task which is rather specific. But the increasing demand for detecting and identifying anomalous events from maritime traffic data keeps attracting more and more researchers' attention.

The approach proposed in this thesis is an unsupervised model based on clustering, so in this chapter, a survey on trajectory clustering and other normal traffic extraction techniques are first given in Section 2.1. Since maritime anomaly detection can be considered as a specific type of anomaly detection tasks in maritime domain, we survey previous work in anomaly detection in Section 2.2 and also introduce some representative work done in the maritime domain.

2.1 Trajectory Clustering And Maritime Traffic Clustering

As stated in the previous chapter, our maritime anomaly detection framework is a clustering-based approach, so in this section, we first introduce the problem of trajectory clustering and then give a survey of various algorithms for solving it. Meanwhile, discussions on the works' limitations for maritime traffic anomaly detection are also conducted while introducing different techniques.

The use of satellites and tracking facilities, such as GPS (Global Positioning System), AIS (Automatic Identification System) and RFID (Radio Frequency Identification Devices) has increased over the past decades, which leads to an increasing number of tracking applications [19]. The data spread over various domains and examples include vehicle position data [18], airplane tracking data [19], ships or vessels tracking data [41][40][57][39][45] and even hurricane or animal movement data [32][17]. One crucial task of the applications is trajectory clustering, which is the process to discover the motion patterns from the tracking data. Exploring different trajectory clustering techniques to extract clusters can help analysts to gain better insights from the data.

Before investigating different clustering algorithms for trajectories, one point needs to be mentioned. Since the goal of trajectory clustering is to find normal paths by grouping similar trajectories together [31], there are lots of works aiming to find proper measures of similarity between trajectories, such as Euclidean distance [18], DTW (Dynamic Time Warping) [27] and LCSS (Longest Common Subsequence) [59]. A comparison between various similarity methods is conducted in [63] and majority of these measures are alignment based distances, which are not appropriate for the ship trajectory extraction task due to the sparseness of the AIS data. So in this thesis, we focus more on other clustering-based techniques to address the task.

According to [20], clustering is the process of grouping a set of physical or abstract objects into classes of similar objects and the proposed clustering algorithms are usually categorized into four types: partitioning methods (e.g., K-Means [46] and K-Medoids [25]), hierarchical methods like BIRCH (Balanced Iterative Reducing and Clustering using Hierarchies) [62], density-based methods including DBSCAN [15], OPTICS (Ordering Points To Identify the Clustering Structure) [5] and DENCLUE (DENsity-based CLUstEring) [22]), and grid-based methods such as STING (STatistical INformation Grid-based method) [60] and CLIQUE (CLustering In QUEst,) [61]). Among various types of clustering approaches, density-based clustering algorithms can be most suitable for this work because they can find arbitrary shapes of clusters. In the following paragraph, one representative approach is introduced and the reasons why this type of algorithms are suitable is discussed as well.

DBSCAN (Density-based spatial clustering of applications with noise) proposed by Martin *et al*. [15] in 1996 is one of the fundamental density-based clustering techniques. The key idea of this clustering algorithm is that for each point of a cluster the neighborhood of a given radius has to contain at least a minimum number of points. So the algorithm requires two additional parameters, eps and $minPts$, as input, where the eps is the radius and $minPts$ is the minimum number of points required to form a dense region. After applying DBSCAN on a set of points, the points are classified as core points, border points and noise points. Here both core points and border points form clusters while the noise points are filtered out. Compared to other types of clustering algorithm, DBSCAN has two main advantages which make it well-suited for the maritime trajectory clustering task. Firstly, the algorithm can provide good results in many cases since it is able to discover clusters of arbitrary shapes and it is robust to detect outliers [20]. Secondly, unlike K-Means [46], users are not required to provide the number of expected clusters before running the algorithm. However, one crucial limitation is that the original DBSCAN algorithm can only take into account location data (Longitude and Latitude), which cannot satisfy the demand of this anomaly detection task in maritime domain. Therefore, as part of our work, we extend DBSCAN to adopt other attributes, such as speed and direction, to find normal traffic patterns from moving trajectory dataset.

To handle specific issues of tracking data, researchers propose various approaches which extend the original DBSCAN in different ways. One method called DB-SMoT (Direction Based Stops and Moves of Trajectories) [45] is a clustering method based on the direction change in a minimal amount of time. For example, if there is a sequence of points with enough direction variance but the duration between the last point and the first point is less than the minimal time threshold, the sequence will not be considered as a final cluster. The main purpose of the algorithm is to find interesting places using single fishing ship trajectory data. The methodology introduced by the authors is simply using DBSCAN to cluster all the data points with a bigger direction variation than the pre-defined threshold. This algorithm is helpful in this specific scenario but cannot be fit for anomaly detection in consideration of the significance of the speed. Different from DB-SMoT, another work done by Palma *et al.* [41] called CB-SMoT (Clustering-Based Stops and Moves of Trajectories) takes speed into account. However, due to the goal of the paper which is also to find interesting places, the method only uses speed as a threshold and then conducts clustering work on the low-speed data set. In this case, the idea of CB-SMoT can be adopted for extracting stopping areas from AIS data. In Section 3.3, we present our approach to cluster and represent the stopping areas in the sea. However, CB-SMoT cannot be applied to extract different clusters with various normal speeds from navigational data set.

One partition-and-group framework, TRACLUS (TRAjectory CLUStering) [32], also uses density-based clustering to detect general lanes. TRACLUS is proposed by Lee *et al.* [32] and is one of the main references in the field of trajectory clustering.

The algorithm is trajectory-based. It tries to solve the clustering problem from the view of line segments of given whole trajectories. The general procedure of TRACLUS is to first partition each trajectory into line segments and then group close line segments into one particular cluster. The reason why they partition the whole track into short sub-segments before the clustering process is that clustering trajectories as a whole could lose similar portions of the trajectories [32]. Lastly, a representative trajectory is computed for each cluster. One limitation of this strategy is that it cannot consider the speed, which is very important to maritime traffic tasks. Another problem in the adaptation of this algorithm to our task is that TRACLUS is sensitive to the parameters (eps and $MinLns$). Although the paper provides an optimal estimation algorithm for thresholds, the experiments conducted in [32] demonstrate that the generated pair of parameters are not guaranteed to be optimal. When we tried to employ the algorithm for extracting normal ship trajectories, a satisfactory result could not be obtained even after a large number of trials. Fortunately, compared to TRACLUS, our model is less sensitive to the parameters as long as the analysts have some knowledge of the maritime domain.

One popular scenario of applying trajectory clustering techniques is video surveillance and monitoring systems. After distinguishing the foreground objects from the background [53] and identifying the positions of the moving objects (e.g. cars, pedestrians, etc.) in the foreground image, trajectory clustering approaches can be applied to group similar trajectories of the moving objects. In the work done by Paciarelli *et al.* [42][50], a dynamic model is presented to group similar trajectories into clusters. The model is essentially maintaining a forest of trees to represent the normal patterns. The prefix of one particular node, from the root node to the current node, stands for a specific normal path formed by all the clusters in the branch (including the root and the node itself). When a new trajectory is fed into the system, a prefix matching will be conducted and the best matching cluster will adopt the new arriving trajectory to update its properties, including physical coordinates and variances of the trajectory points inside the cluster. One issue of the approach is that a large number of trees need to be stored, because different trajectories may have different starting points and each start point needs a corresponding root node. Another limitation is that the prefix matching phase requires complete trajectories to be observed, which is infeasible in maritime traffic tracking. As we know, it is common to have gaps in the AIS data stream and the gaps can vary from milliseconds to days. In addition to the above two problems, similar to TRACLUS [32], it cannot take speed into account, which is a fundamental issue in coastal surveillance. While in this thesis, the point-based approach is proposed to address the issue of the inconsistent updated trajectory data and no concern about prefix matching exists.

Inspired by the work done by Paciarelli *et al.* [42][50], Guillarme *et al.* propose a trajectory-based "unsupervised normalcy model" [31] for S-AIS data. This work is also based on stop and move framework [51] and applies corresponding strategies on moving trajectories and stopping areas. Similar to TRACLUS [32], this technique is also a partition-and-group framework, which first partitions the historical trajectories into sub-segments. To extract normal patterns from moving trajectories, the authors introduce a technique based on the OPTICS algorithm [5]. OPTICS has the advantage of handling situations where clusters exist at different density levels, but it needs more experts' input knowledge for selecting relevant clusters from different clusters levels, which is more complicated than our non-hierarchical clustering approach. So this is less feasible for applying the algorithm in a large area with complicated traffic routes. Then for stopping areas extraction, the authors present a representation to estimate the locations and the spatial extent of a stopping area. This method is similar to their solution used for moving trajectories, that is, it tries to use one particular node to represent an arbitrary shape of region. This will cause a bias while it comes to coastal stopping areas since the areas might not be in circle shapes. Fortunately, the Sampled Stopping Points proposed in this work can represent arbitrary shapes of regions well.

Ferreira *et al.* [17] propose a strategy called VFKM (Vector Field K-Means) based on vector field fitting. The idea behind their method is to induce a similarity notion on the data set by using streamlines of a single vector field. Two main elements of this algorithm are fitting the vector fields and assignment of trajectories to clusters [17]. Since the approach assigns the trajectories to the vector fields, it is also a trajectory-based method. VFKM is an iterative model similar to original K-Means [46] and two main steps are involved in each iteration. First, the trajectories are partitioned randomly into K clusters (similar to the initialization of original K-Means). Then the clusters are updated by assigning each trajectory to the vector field that it fits best. These two steps are repeated iteratively until a convergence criterion is met. There are several limitation related to the algorithm. Firstly, as mentioned by the authors [17], the performance of the algorithm is sensitive to the choice of initial clusters. Secondly, although this method can consider other features (e.g. direction and speed) of the trajectory data, it cannot provide a representative trajectory. Instead, the algorithm output is k representative vector fields which cannot be employed in our case. Fortunately, in our thesis, the output of the maritime traffic extraction model is two sets of representative vectors (Gravity Vectors and Sampled Stopping Points), which is feasible for handling the anomaly detection problem.

One work which provides mechanisms to manage traffic in the ocean that is similar to ours is an unsupervised framework called TREAD (Traffic Route Extraction and Anomaly Detection), proposed by Pallotta *et al.* [40][57][39]. It is designed to detect low likelihood behaviors and predict future vessel positions from maritime traffic data. TREAD is a point-based framework and the traffic routes are built by the way-points generated by the clustering process, that is, the route objects are directly formed by the flow vectors of the vessels whose paths connect the derived way-points. The assumption of using waypoints to build the traffic lanes is that the vessels' trajectories are composed of a set of connected straight lines. Another similar methodology proposed by Gariel *et al.* [19] is also a way-point-based clustering framework for handling airspace monitoring. The objective of the method is to identify and group the turning points into way-points and to use the sequence of way-points to represent the aircraft's trajectories. To get the way-points from the historical flight data, two traditional clustering algorithms are employed. When the spatial distribution is sparse, K-MEANS [46] is used, and when the distribution is dense, DBSCAN [15] is used. This point-based idea has a practical advantage of handling trajectories of unequal length or with gaps, which is common due to the low satellite coverage of AIS around the world. The main difference in relation to TREAD [40] relies on the fact that our work tries to directly map the rules defined by IMO into real data available from AIS readings.

Another main advantage of the clustering framework proposed in this thesis is that in the clustering process the IMO rules, particular Traffic Separation Schemes, are taken into account, which is not the case in previous works. Therefore, our approach maps rules defined by IMO directly to the AIS data, and in that sense it is similar to the Candide [4] system, which is a digital face reconstruction mask. Candide uses a pre-defined digital face mask to be adjusted in a facial picture, and allows the generation of new facial expressions without any extra information. From the rule mapping perspective, this work does the same, with the difference that we use the set of rules defined by IMO and check if vessels are following them.

2.2 Anomaly Detection in Maritime Surveillance

The objective of anomaly detection is to find patterns in data that do not conform to expected behavior and these nonconforming patterns are often referred to as anomalies or outliers [10]. Anomalies or outliers are related to, but distinct from, noise in the data. As defined in [10], noise is some patterns or phenomenons in data that are not of interest to the analyst, but act as a hindrance to data analysis. Thus, activities on handling these unwanted noise involve noise removal [55] and noise accommodation [47]. But in anomaly detection tasks, the outliers or the anomalies are the patterns which attract much attention from the analysts in various domains, such as fraud detection for credit cards [52], insurance [58], or health care [34], intrusion detection for cyber-security [33] and military surveillance for enemy activities [10]. In this thesis, our work focuses on detecting the anomaly navigational behaviors of the ships or vessels.

Maritime Anomaly Detection techniques primarily fall into two categories: statistical modelling [29][44][13] and predictive modelling [40][38][28]. The general idea of statistical techniques for anomaly detection is to fit a statistical model for normal behaviors with the given data set and then apply a statistical inference test to determine if an unseen instance belongs to the model [10]. Approaches based on predictive models usually predict future status information (e.g. position, speed and course) of a particular vessel and then compare the real data with the prediction to decide the abnormality. Additionally, there is another type of approach trying to solve the problem based on predefined rules. One representative work was done by Roy *et al.* [48][49] and an expert system for automated anomaly detection was introduced. As we know, one limitation about these kind of systems is that they are relying on the knowledge or predefined rules, as a result, much effort needs to be done to categorize anomalies. In the papers [48][49], the authors identified a taxonomy of kinematic and geo-spatial concepts in the vessels tracking domain and the taxonomy has to be updated once a new rule is found. Since not much work focuses on rule-based type of techniques, the survey of maritime anomaly detection is more about the other two main categories.

In the maritime domain, the majority of statistical models are built upon the momentary kinematic features (position, course, speed and acceleration rate) of individual vessels.

Laxhammar [29] used a Gaussian Mixture Model (GMM) and a greedy version of Expectation-Maximization (EM) for clustering. A GMM can be regarded as an ensemble model of K multivariate Gaussian distributions (mixture components). GMM is an extension of single Gaussian and it has the advantage of approximating arbitrarily complex distributions in arbitrarily high dimensions. The greedy EM-learning (greedy Expectation-Maximization-learning) algorithm, an extension to the classical EM-algorithm, is employed to determine the optimal number of components (or distributions) and the parameter set for all the distributions. So the greedy EM algorithm is basically an iterative model with mainly two steps in each iteration; the first step is to insert a new component and the second step is to apply classical EM until convergence. Then in order to label new data, the likelihood of the new point will be first calculated based on the probability distribution obtained from clustering phase. Further, it will be classified as abnormal if the likelihood is below a certain predefined alarm threshold.

According to [30], GMM is not an optimal model because it needs an assumption that the distribution of vessel positions along the major axis of the sea lane segments is uniform. To tackle this issue, in [44], the authors propose to use adaptive Kernel Density Estimator (KDE) for estimating unknown probability densities and modelling arbitrary sea lanes. KDE, also known as the Parzen Window method, is a non-parametric model and the estimated PDF (Probability Density Function) can be determined merely based on the training data, which is superior to GMM. Then in the anomaly detection phase, the anomaly detector is sequentially applied to the incoming data. The value of new incoming point's density is calculated under the null hypothesis (no anomaly) and this value is then compared with a detector parameter related to false alarms for deciding the new point's abnormality.

A comparison between the two approaches above [29] and [44] is given in [30], demonstrating that the anomaly detection results from both models are not satisfactory. As the two models detect the anomalous segments after rather long distances (three kilometer and four kilometer respectively) while an expected effective anomaly detector should detect such behaviors at a shorter distance [30]. Another disadvantage of the two statistical methods is that it is hard to interpret, in terms of anomaly types, even if they could find anomaly patterns promptly. Once a point is decided as abnormal, the likelihood value in [29] and the density value in [44] cannot indicate the types of the abnormality. For example, the analysts cannot know whether the point deviates too far away from the normal lanes or the vessel sails too slow or too fast compared to normal speed.

In [13], Gerben *et al.* propose a technique based on Machine Learning models. In their work, different trajectory alignment kernels (Dynamic Time Warping [27] and Edit Distance [37]) are applied with one-class SVMs (Support Vector Machine) [11] for detecting the outlying trajectories. This trajectory-based method requires that the complete trajectory has been observed before it can be classified as normal or anomalous [43], which is not applicable for sequential anomaly detection in incomplete trajectories (real-time AIS surveillance), unlike our proposed point-based method.

Pallotta *et al.* [40] suggest the use of rule-based and low-likelihood models for anomaly detection. As stated in previous paragraph, the rule-based techniques generate alerts based on a set of pre-defined rules. So the rules defined in this approach are similar to those in other knowledge-based work, which require maritime domain experts' knowledge. As an example, no specific rules defined for maximum speed in different sea regions can be found in IMO publications [1][24], so the maximum speed pre-defined in a port area can only be accurately estimated by a specialist knowledgeable about the area. Similar to the work done in [29], the low-likelihood anomaly detection aims at detecting deviations from the normal patterns or distributions derived from the training AIS data. For this low-likelihood detection, a Weibull model was employed (a parametric exponential-like model), along with a sliding time window technique to avoid problems with incomplete and intermittent tracks.

Similar to the work done in [40], Nevell [38] proposes to use a Bayesian approach to predict the future route of a particular vessel for comparison. This methodology is based on a node-sparse network, built from different kinds of coastal nodes. The idea of pre-defining a global maritime traffic network is also adopted in [7] where the network is constructed based on the historical data. But instead of Bayesian approach, Soleimani *et al.* [7] employ $A*$ [14] algorithm to decide the optimal or expected route. In [28], the authors insist that an overall threat is indicated by a sequence of the individual behaviours. Therefore, five specific anomalies (deviation from standard route, unexpected AIS activity, unexpected port arrival, close approach with another ship and entering a zone known as illegal exchanges) are introduced to extend Nevell's work [38] to assess the probability of a higher-level threat based on a constructed Bayesian Network.

One issue with the work described in [38][28] is that it cannot incorporate speed into deciding if a trajectory is anomalous; instead the judgement is based only on position. Another problem is that a pre-defined network may not be applicable in many near-port regions due to the nature of port traffic. Traffic in near-port areas is usually variable and the vessels are not always following straight lanes (optimal routes in [38][28]). Fortunately, both of these problems are handled with our approach.

The anomaly detection model in this thesis uses the results of the clustering framework presented in Chapter 3, which generates normal moving patterns and arbitrary shapes of stopping areas. The proposed anomaly detection method is point based but is capable of handling trajectory tracks. For each track the algorithm will return an anomaly ratio. The ratio is based on three types of specialized distances to take position, direction and speed into consideration.

Chapter 3

Normal Traffic Patterns Extraction Model

Vessels always follow different movement patterns in different areas. For instance, cargo ships may travel along straight lines at high speed in the middle of the sea, while they may frequently adjust their directions at low speed in the port or offshore platform areas.

As a consequence, Spaccapietra [51] introduced a model to reason about trajectories, which is called stops and moves. So a trajectory can be treated as a sequence of moves and stops [51], and in the work done by [57][40][31], maritime trajectories analysis is conducted from these two different aspects as well, moving patterns and stopping patterns distinguished by a speed threshold. According to [31], the stop and move model has two main advantages: 1) stopping areas are interesting areas which need to be discovered; 2) stopping points are one importance source of noise during trajectory clustering and stopping points do not include any motion information for path modelling. In addition to the above mentioned advantages, another benefit is that it can help to decrease the search space complexity during moving trajectories clustering since the stopping points are filtered out before applying the algorithm.

In this work, we employ a similar method to extract different normal patterns from the historical AIS dataset with a stopping SOG (Speed Over Ground) threshold of 0.5 knots. More specifically, we first divide the historical AIS data into two subsets: the set of moving points (with SOG not less than 0.5 knots) and the set of stopping points (with SOG less than 0.5 knots). Instead of merely clustering the waypoints of the trajectory data [57][40], normal moving trajectories and arbitrary shapes of stopping regions will be extracted based on the entire moving and stopping dataset in the specific area, respectively.

In this chapter, a definition for the trajectory in maritime domain (Section 3.1) is first given and then the model for extracting normal patterns from the historical AIS data is proposed. For the case where SOG is not less than 0.5 knots (moving patterns), we propose DBSCANSD (Density-Based Spatial Clustering of Applications with Noise considering Speed and Direction) as the basic algorithm to detect the main traffic lanes within the data. The algorithm's output is a set of Gravity Vectors (GV), which are vectors formed by 5 features: average COG, average SOG, average Latitude, average Longitude and Median Distance. In Section 3.2, for the case where the SOG is less than 0.5 knots (stopping areas), the original DBSCAN [15] algorithm is executed because speed and direction are not important factors. Another type of vector is created as output, labelled Sampled Stopping Point (SSP), which is dependent only on the geographic shape of the region. (Section 3.3)

3.1 Representing Trajectory Data

Definition 1 *(Trajectory) A trajectory is a finite sequence $T = ((x_1, t_1), (x_2, t_2), \ldots, (x_m, t_m))$. Each data point x_i corresponds to a multi-dimensional feature vector of a moving object at time point t_i, where $t_i < t_{i+1}$ for i =1,..., m-1.*

A trajectory can be defined as a data type representing the movement of an object. A trajectory can be represented by a multidimensional time series [3] or a sequence of multi-dimensional points [32]. In this thesis, we modify the definition of its raw form (Definition 1) [43] to adapt it for our work. The new definition of a ship trajectory is as following:

Definition 2 *(Trajectory in Maritime Domain) A trajectory is a finite sequence $T = ((x_1, t_1), (x_2, t_2), \ldots, (x_m, t_m))$ where x_i is a set of $< Latitude, Longitude, COG, SOG >$ and t_i is the time-stamp.*

Here in Definition 2, each vector x_i is called a **trajectory point**, but in this thesis, we simply use **point** or **data point** where the context is clear.

3.2 Normal Moving Trajectories Extraction

As stated at the beginning of the chapter, given a data set of AIS historical tracking points, we divide them into stopping and moving parts based on the threshold of 0.5 knots. In this section, we present our moving trajectory extraction model.

As we know, compared to stopping patterns, the moving part plays a more important role in the anomaly detection task since there is a higher collision risk for a vessel sailing at high speed than at low speed in the stopping areas. To extract normal patterns from the moving AIS data, we first propose the new clustering algorithm DBSCANSD (Density-Based Spatial Clustering of Applications with Noise considering Speed and Direction).

3.2.1 Density-Based Spatial Clustering of Applications with Noise considering Speed and Direction

Density-Based Spatial Clustering of Applications with Noise considering Speed and Direction (DBSCANSD), shown in Algorithm 1, is a density-based clustering algorithm based on the algorithm DBSCAN [15]. The main observation behind our approach is that it is common for different types of ships to sail with different velocities in one similar area of the sea. For instance, the cruise speed of a cargo ship can be faster than a fishing ship. Even the same type of vessels can behave differently in relation to direction. Imagine that an oil tanker sails between two different countries. When the vessel is full, its speed is slower than when it is empty.

Definition 3 *(Eps-neighborhood of a point) The* $Eps-neighborhood$ *of a point* p*, denoted by* $N_\epsilon(p)$*, is defined by* $N_\epsilon(p) = \{q \in D \mid dist(p,q) < \epsilon\}$

As discussed in Chapter 2, the key idea of DBSCAN [15] is that for each point of a cluster the neighborhood of a given radius has to contain at least a minimum number of points. Here in this thesis, we adopt this idea but also consider two other factors, maximum speed variance ($MaxSpd$) and maximum direction variance ($MaxDir$). The intuition behind this is that the neighbors of a trajectory point should be not only near enough, but also with similar COG (Course Over Ground) and SOG (Speed Over Ground). Thus, we can modify the definition of Eps-neighborhood (Definition 3) in [15] to Definition 4.

Definition 4 *(Eps-neighborhood of a trajectory point) Given a database* D *of moving trajectory points in a specific area, the* $Eps-neighborhood$ *of a trajectory point* p*, denoted by* $N_\epsilon(p)$*, is defined by* $N_\epsilon(p) = \{q \in D \mid dist(p,q) < \epsilon \text{ and } |p.SOG - q.SOG| < MaxSpd \text{ and } |p.COG - q.COG| < MaxDir\}$

Note that $dist(p, q)$ is the Geographical Distance [56] between p and q, instead of Euclidean distance, because it is necessary to take in consideration the Earth's curvature to calculate distances.

Then in the following, we also give the formal definitions of other essential notions for our density-based algorithm. The definitions are changed from the ones originally defined for 2-Dimensional points in the algorithm of DBSCAN [15] to those for trajectory points.

Definition 5 *(Core trajectory point) Given a database D of moving trajectory points in a specific area, a trajectory point p in D is called a core trajectory point w.r.t. the parameters of* ϵ*, MinPts, MaxSpd and MaxDir if* $|N_\epsilon(p)| \geq MinPts$ }

Definition 6 *(Directly density-reachable) Given a database D of moving trajectory points in a specific area, a trajectory point p in D is directly density-reachable from another trajectory point q in D w.r.t. the parameters of* ϵ*, MinPts, MaxSpd and MaxDir if* $p \in N_\epsilon(q)$ *and* $|N_\epsilon(q)| \geq MinPts$ }

From Definition 6, we can find if point q is a core trajectory point according to Definition 5. Because $|N_\epsilon(q)| \geq MinPts$, q is a core trajectory point. If the point q is also directly density-reachable from the point p, the point q will be also a core trajectory point clearly. But when the point p is not a core point and it is directly density-reachable from core point q, the point p is called **Border Trajectory Point**.

Definition 7 *(Density-reachable) Given a database D of moving trajectory points in a specific area, a trajectory point p in D is density-reachable from another trajectory point q in D w.r.t. the parameters of* ϵ*, MinPts, MaxSpd and MaxDir if there is a chain of trajectory points* $p_1, ..., p_n, p_1 = q, p_n = p$ *such that* p_{i+1} *is directly density-reachable from* p_i *(* $1 \leq i < n$*)* }

From Definition 7, we can find the trajectory point p can be either a border trajectory point or core trajectory point if p is density-reachable from point q. But the point q and other points (except p) in the chain must be core trajectory points. Therefore, this relation is transitive, but it is not symmetric [15].

Definition 8 *(Density-connected) Given a database D of moving trajectory points in a specific area, a trajectory point p in D is density-connected to another trajectory point q in D w.r.t. the parameters of ϵ, MinPts, MaxSpd and MaxDir if there is a trajectory point o such that both p and q is density-reachable from o w.r.t. ϵ, MinPts, MaxSpd and MaxDir}*

However, if there exist multiple trajectory points (p_1, ..., p_n) that are density-reachable from one core trajectory point o, then points p_1, ..., p_n can be all border trajectory points. In such a cluster, one border trajectory point (p_1) is obviously not density-reachable from any other border points (p_2, ..., p_n). So it is necessary to define a relationship (Definition 8) for any pairs of two border trajectory points in the same cluster.

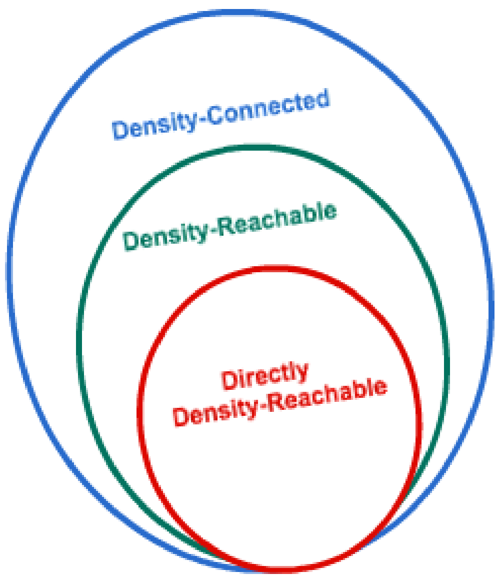


Figure 3.1: Three defined relations for the density-based clustering algorithm.

With the three density-based relations defined as above, it is easy to see the relationships between them. Figure 3.1 demonstrates these relationships. For example, if there are two trajectory points p and q, and p is directly density-reachable from q, p will be not only density-reachable but also density-connected from q. However, if p is density-reachable from q, it will be also density-connected from q but may be not directly density-reachable from q.

Now the concepts of **Cluster** and **Noise** (Definition 9 and Definition 10) can be defined based on the above three relations (Directly density-reachable, Density-reachable and Density-connected):

Definition 9 *(cluster) Given a database D of moving trajectory points in a specific area, a cluster C w.r.t.* ϵ *and MinPts is a non-empty subset of D satisfying the following two conditions:*

1) $\exists$ *p* $\in$ *C,* $\forall$ *q* $\in$ *D, if q is density-reachable from p w.r.t. the parameters of* ϵ*, MinPts, MaxSpd and MaxDir, then q* $\in$ *C*

2) $\exists$ *p* $\in$ *C,* $\forall$ *q* $\in$ *D, if p is density-connected to qw.r.t. the parameters of* ϵ*, MinPts, MaxSpd and MaxDir, then q* $\in$ *C* }

Definition 10 *(noise) Given a database D of moving trajectory points in a specific area, and* C_1*, ...,* C_k *are the clusters of Dw.r.t. the parameters of* ϵ*, MinPts, MaxSpd and MaxDir, i = 1, ..., k. Then the noise are the set of trajectory points in the database D not belonging to any cluster* C_i*, i.e. noise =* { *p* $\in$ *D* | $\forall$ *i: p* $\notin$ C_i } }

Algorithm 1 presents the procedures of DBSCANSD. It requires 5 parameters (*DatasetM*, *eps*, $MinPts$, $MaxDir$, $MaxSpd$) as input. *DatasetM* is a list of all the moving points in the trajectories. *eps* and $MinPts$ are the reachable distance and reachable minimum number of points (see Ester *et al* [15]). While the other two, $MaxDir$ and $MaxSpd$, are the two new parameters (maximum direction variance and maximum speed variance). The algorithm starts with a random selected trajectory point and then retrieve all its neighbour points. If its neighbour points' number is exceeding the threshold of $MinPts$, the point is marked as core trajectory point and all its neighbours are adopted in the cluster. This procedure then iterates until all the points have been visited and the output is a set of clusters.

Time Complexity Analysis of DBSCANSD algorithm:

`procedure QueryNeighbourPoints` (lines 15-25):

The dominating operation is the `if` test (lines 18-20), data size: $n =$`data.size`. The implementation presented in Algorithm 1 has linear time complexity $O(n)$ where n is the size of the `data` set. However, if some spatial index is used the time complexity can be reduced to $O(log(n))$.

Algorithm 1 DBSCANSD

```
1: procedure DBSCANSD($DatasetM$, $eps$, $MinPts$, $MaxDir$, $MaxSpd$)
2:     Mark all points in moving dataset $DatasetM$ as unclassified
3:     $clusterList \leftarrow$ empty list
4:     for each unclassified point $P$ in $DatasetM$ do
5:         Mark $P$ as classified
6:         $neighborPts \leftarrow$ queryNeighborPoints ($DatasetM$, $P$, $eps$, $MinPts$, $MaxDir$, $MaxSpd$)
7:         if $neighborPts$ is not $NULL$ then
8:             $clusterList$.add($neighborPts$)
9:     for each cluster $C$ in $clusterList$ do
10:        for each cluster $C'$ in $clusterList$ do
11:            if $C$ and $C'$ are different clusters then
12:                if mergeClusters($C$, $C'$) is TRUE then
13:                    $clusterList$.remove($C'$)
14:    return $clusterList$
15: procedure queryNeighborPoints($data$, $P$, $eps$, $MinPts$, $MaxDir$, $MaxSpd$)
16:    $cluster \leftarrow$ empty list
17:    for each point $Q$ in data do
18:        if distance($P$,$Q$) $< eps$ then
19:            if $|P.SOG - Q.SOG| < MaxSpd$ then
20:                if $|P.COG - Q.COG| < MaxDir$ then
21:                    $cluster$.add($Q$)
22:    if $cluster$.size $> MinPts$ then
23:        Mark $P$ as core point
24:        return $cluster$
25:    return $NULL$
```

```
26: procedure mergeClusters($clusterA$, $clusterB$)
27:     $merge \leftarrow FALSE$
28:     for each point $Q$ in $clusterB$ do
29:         if point $Q$ is core point and $clusterA$ contains $Q$ then
30:             $merge \leftarrow TRUE$
31:             for each point $Q'$ in $clusterB$ do
32:                 $clusterA$.add($Q'$)
33:             break
34:     return $merge$
```

`procedure MergeClusters` (lines 26-34):

The dominating operation is the `contains` test operation (line 29), the data size is the number of elements the second cluster $b =$`clusterB.size`. In the simple implementation presented in Algorithm 1 the worst time complexity is linear $O(b)$, if we assume the constant time cost of `contains` operation (this can be achieved by keeping a hash-set structure). However, this procedure can be accelerated with more sophisticated data structures, in particular avoiding adding elements one by one.

`procedure DBSCANSD` (lines 1-14):

Finally, the main procedure `DBSCANSD` is dominated by the two consecutive loops: point neighbourhoods computation (lines 4-8) and cluster merging (lines 9-13). The data size is the total number of elements $n =$`DataSetM.size`. The initialisation (line 3) is linear $O(n)$. Each of the two loops (lines 4-8 and lines 9-13) in the simple implementation presented in Algorithm 1 can be bounded with the quadratic time complexity $O(n^2)$, however, as we mentioned above, it is possible to accelerate it by applying more sophisticated data structures in the sub-procedures. To sum up, the `DBSCANSD` algorithm has at most quadratic time complexity $O(n^2)$ but there is definitely a room for improvement that will be further explored as the part of the future work.

We use external knowledge based on the Traffic Separation Schemes (TSS) [1] defined by IMO to adjust the parameters of the algorithm; therefore, we are using the lanes defined in IMO publications to fit the clustering results. As we know, once the area is determined, the two parameters $MaxDir$ and $MaxSpd$ should be fixed. During this thesis work, multiple pairs of values were tested to find the optimal $MaxDir$ and $MaxSpd$. Our experiments indicate that $5°$ and 5 knot are the best parameters for both tested ports (Los Angeles Long Beach area and Juan De Fuca Strait area).

On the other hand, the parameters *eps* and $MinPts$ must be defined for each port. Thus, another set of experiments were done to select the best pair for each area. For instance, when a lane generated by the clustering algorithm has any gaps compared to the one in the IMO TSS regulations, the analyst must increase the value of *eps* or decrease that of $MinPts$. In this way, the lanes with lower densities might be adopted as parts of the clusters to fill in the gaps. The two parameters *eps* and $MinPts$ selected in this procedure will be directly used in the next stopping area extraction phase.

After applying DBSCANSD to the ship trajectory dataset, those geographically close trajectory points with similar direction and speed will be grouped together to form a cluster. Then, we can not only get arbitrary shapes of clusters , but we can also separate those close points with great different normal SOGs and COGs into multiple clusters. Moreover, the algorithm can even treat a ship's acceleration or deceleration as a cluster, as long as the $MaxSpd$ is well defined. Similarly, if the $MaxDir$ is defined well, a curve shape of navigation will be also treated as one cluster.

3.2.2 Gravity Vector

Although the clustering results can reflect normal patterns of the vessels, it is not feasible to employ all the trajectory points in every cluster to decide the abnormality of the new coming trajectory data. Because the time complexity of comparing the new trajectory and the clusters is $O(n * m)$ where n is the total number of the trajectory points in the clustering results and m is the number of points in the new coming trajectory. As a result, an approach for clusters reduction is necessary.

Some techniques [12][36] that try to map the results to another representative objects are proposed for compressing the original clustering results. For example, a spline-based clustering approach is introduced in [12], however, since our approach is a point-based clustering algorithm and the output is multiple sets of points, the idea of using splines is not applicable in this work. Another idea called centroid and envelope has been used for path modeling in vision-based trajectory learning [36]. The centroid can minimally specify the corresponding path and the envelope is used for denoting the path extent. In our case, we adopt a similar idea as centroid and envelope. Here we call it **Gravity Vector (GV)**.

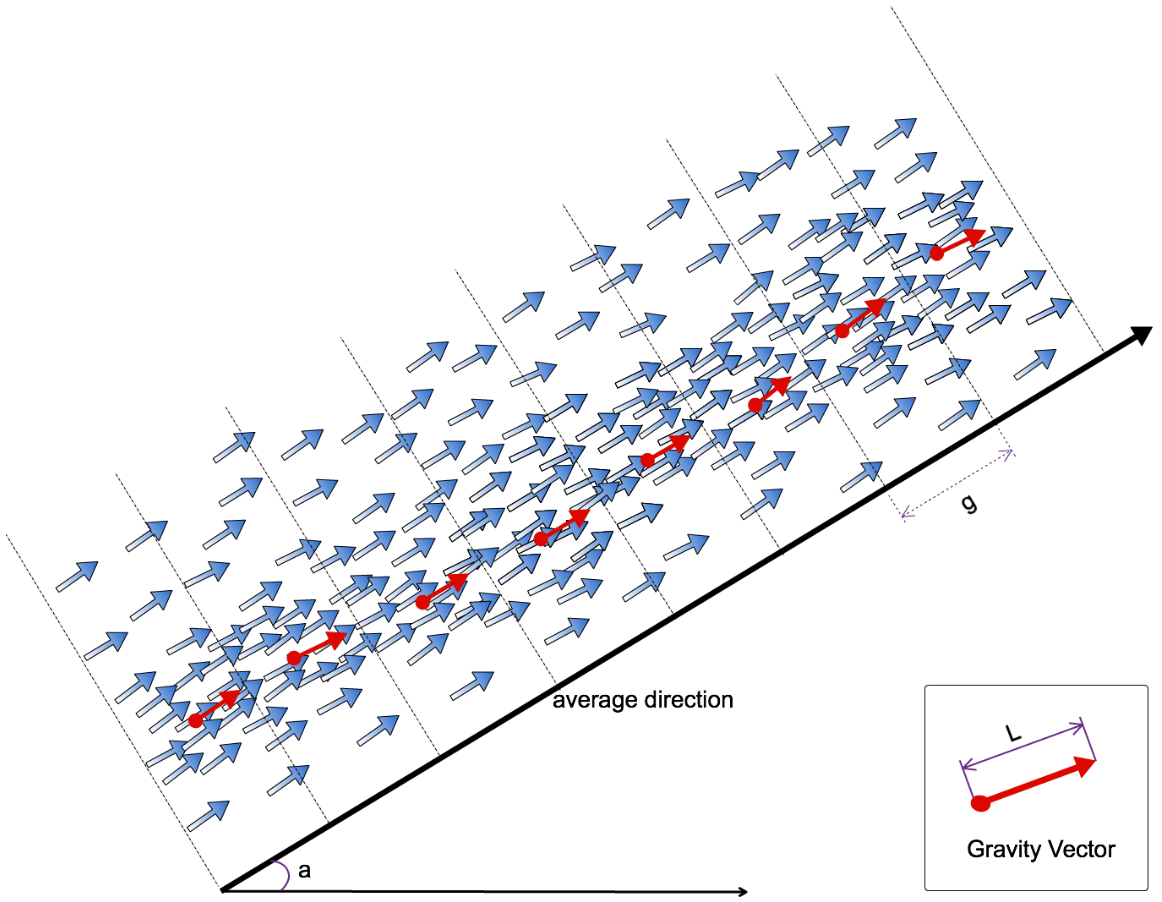


Figure 3.2: Calculate Gravity Vectors (GV) for one moving cluster. Blue arrows stand for the trajectory points of the cluster, red arrows are the final Gravity Vectors. The length (L) of the GV is the SOG_{avg} of the GV. The width of a grid g is the pre-defined length for partitioning the points.

A Gravity Vector is extracted by partitioning a cluster into multiple parts, therefore, each cluster can contain multiple Gravity Vectors. Figure 3.2 presents an example of calculating the Gravity Vectors of one cluster. To partition a cluster, a grid width needs to be first determined. The grid width (g in Figure 3.2) can be decided based on the domain knowledge or multiple experiments' results. Here in this thesis, we choose the length of *eps* used in Algorithm 1 as that of g. A Gravity Vector is a vector formed by 5 features: average COG, average SOG, average Latitude, average Longitude and Median Distance. Then one Gravity Vector GV_i can be denoted by:

$$GV_i =< COG_{avg}, SOG_{avg}, LAT_{avg}, LON_{avg}, D_{median} > \tag{3.1}$$

As seen in Figure 3.2, all the blue arrows in the figure are belonging to the same cluster and we can see that all the trajectory points in this cluster have similar COGs (directions). Although it is a bit hard to see the speeds from the figure, we know each point in the cluster has minor SOG variation compared with its neighbors' SOGs. In this example, since the cluster is partitioned into 8 grids, the final output for representing the cluster will be a set of 8 Gravity Vectors. The steps of calculating the Gravity Vectors are as following:

Step (1) Calculate the average COG of the whole cluster and let it be COG_w. Note that COG_w is not COG_{avg} in equation 3.1.

Step (2) Partition all the trajectory points in the cluster along the direction of COG_w by a pre-defined grid width (g in Figure 3.2).

Step (3) For each grid of the partitioning results, generate the gravity vector of the grid.

In step (3), assume there are k trajectory points in one grid and TP_i is the ith trajectory point, we can calculate the first 4 features using the following 4 formulas (3.2–3.5).

$$COG_{avg} = \frac{\sum_{i=1}^{k} TP_i.COG}{k} \tag{3.2}$$

$$SOG_{avg} = \frac{\sum_{i=1}^{k} TP_i.SOG}{k} \tag{3.3}$$

$$LAT_{avg} = \frac{\sum_{i=1}^{k} TP_i.Latitude}{k} \tag{3.4}$$

$$LON_{avg} = \frac{\sum_{i=1}^{k} TP_i.Longitude}{k} \tag{3.5}$$

Then to calculate the last feature D_{median}, we should first calculate the distances between all k points in the grid and the average geographical point (LAT_{avg}, LON_{avg})

generated by formula 3.4 and formula 3.5. After this, we can apply a linear time complexity algorithm for computing median D_{median} based on Hoare's PARTITION algorithm.

In Lemma 1, we present the time complexity of calculating GVs for a cluster.

Lemma 1 *The worst time complexity of calculating a specific cluster's Gravity Vectors is O(n), where n is the total number of points in the cluster.*

Proof. Assume there are n points in one cluster and the cluster is partitioned into m grids. Step (1) takes O(n) time to calculate the average COG of the whole cluster. Step (2) takes O(n) time to map all the points to the axis of average COG and O(n) time to partition them into m grids. In step(3), to calculate one particular GV for the ith grid with k_i points, it takes $O(k_i)$ time to calculate the first 4 features and $O(k_i)$ time to calculate the median distance in linear time using one of the algorithms based on the Hoare's partition algorithm [23], for example the Blum-Floyd-Pratt-Rivest-Tarjan algorithm [8] that has linear worst time.

And this procedure needs to be repeated for m times, the total time for step (3) can be calculated as following:

$$W(n) = \sum_{i=1}^{m} O(k_i) = O(n)$$

Thus the pessimistic total time complexity for the whole procedure from step (1) to step (3) is O(n).

■

The feature median distance can be used to measure the width variance of a moving cluster. The work done by Etienne *et al* [16] shows the choice of the statistical decile used to compute the spatio-temporal channel can give a tolerable estimate of this channel's width. However, we use median rather than ninth decile in [16] to provide a more robust width estimation metric. The distance is not the exact width of the channel, instead, it is a relative distance for outlier detection. More detailed information about how to use the GV for anomaly detection is presented in Chapter 4.

3.3 Normal Stopping Areas Extraction

In stopping areas, such as ports and wharfs, there may be different types of ships at anchor with zero-velocity or new coming vessels entering the area with extremely low speed. In this case, vessels can stop with their prows pointing to any directions and change their headings frequently to arrive at the anchorage berths. Therefore, direction is no longer an essential part for analyzing the risk of collision. The possibility of a low-speed vessel colliding with another ship is very low; therefore this situation was excluded from the clustering process. In other words, only when a ship sails with a higher speed than the stopping threshold in a stopping area can we label the data as outlier. So the task of this stage is to identify the locations and geographical shapes of the stopping areas from the historical data set.

Based on the analysis of stopping regions, the original DBSCAN [15] algorithm can be employed for stopping points clustering since speed or direction is no longer essential factors. The output is a set of stopping clusters with high density of trajectory points. Intuitively, there are a relative large number of vessels gathered in each stopping cluster, which indicates the region is a reasonable place for ships to anchor.

It should be noted that it is common to have both stopping clusters and moving clusters in a specific region at the same time, e.g., a shipping lane can pass through a stopping area. So the phenomenon that a vessel is approaching the port (stopping area) with a low speed is not supposed to be considered abnormal. Our approach can handle these kinds of issues because both types of clusters can be generated during the normal traffic patterns extraction step and the abnormality of a point will be decided based on both types clusters.

Although DBSCAN [15] can generate stopping points clusters, these original clustering results are still not adaptable for the next anomaly detection phase because the relatively large number of points in each cluster may pose serious constraints regarding the computational feasibility. Consequently, just like GV, a similar representative point called **Sampled Stopping Point (SSP)** is proposed. Algorithm 2 shows the whole procedure for generating the final SSPs given a set of stopping trajectory points in a specific region.

As shown in Algorithm 2, stopping points are first clustered by the DBSCAN [15] algorithm (line 1). The same parameters (eps and $MinPts$) used by DBSCANSD

Algorithm 2 Extract SSP From Stopping Dataset

```
Input: The list of the stopping points of the region, $DatasetS$; Reachable distance, $eps$; Reachable minimum number of points, $MinPts$
Output: The list storing the region's SSP, $resultSSP$
1: $StoppingPointsClusters \leftarrow$ DBSCAN($DatasetS$, $eps$, $MinPts$)  ▷ see DBSCAN algorithm in [15]
2: $resultSSP \leftarrow$ empty list
3: for each cluster $C$ in $StoppingPointsClusters$ do
4:     $lat1, lat2 \leftarrow$ minimum and maximum of all the points' Latitude values in cluster $C$
5:     $lon1, lon2 \leftarrow$ minimum and maximum of all the points' Longitude values in cluster $C$
                                                   ▷ estimate the sample size for cluster $C$
6:     $area \leftarrow |(lat1 - lat2) * (lon1 - lon2)|$
7:     if $area = 0$ then
8:         $sample\_size \leftarrow 1$
9:     else
10:        $sample\_size \leftarrow$ Ceiling($area/(\pi * eps^2)$)
                                                   ▷ sample the stopping area points
11:    $count \leftarrow 0$
12:    while $count < sample\_size$ do
13:        randomly select one point $P$ from cluster $C$
14:        if $P$ is far from all points in $resultSSP$ then
15:            $resultSSP$.add($P$)
16:            $count$++
17: return $resultSSP$
```

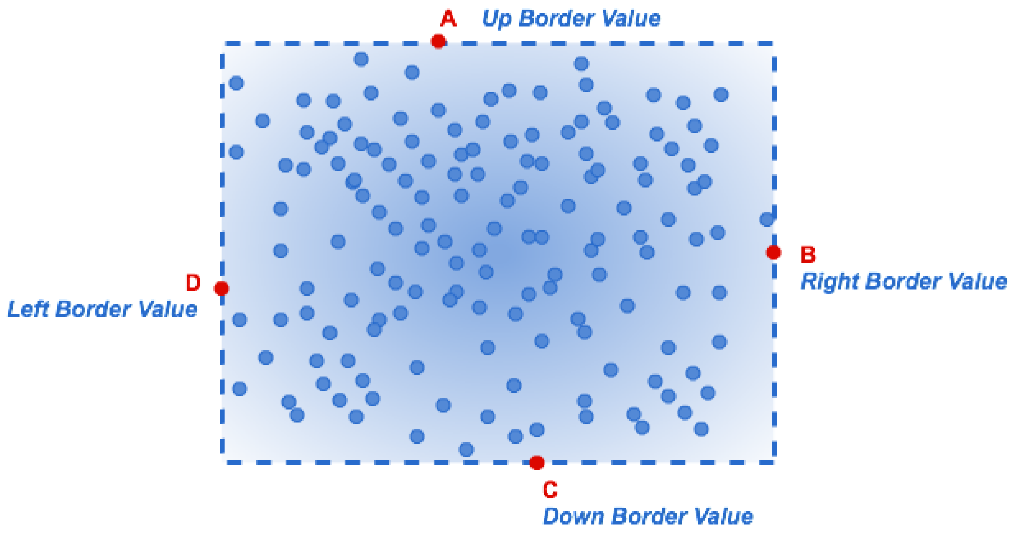


Figure 3.3: Border Values of a stopping cluster

(Section 3.2) are still employed in this stopping areas extraction phase. Then the algorithm starts to generate the SSPs for every cluster. The key idea about SSP is to find the representative points that reflect the shapes and locations of the stopping areas. But before extracting the points from the clustering results, an estimation of the number of SSPs needs to be conducted. To address this issue, lines 4−10 first estimate the area of the region and then calculate the sample size. Lines 4 and 5 calculate the four border values of the cluster. Border values are extreme values of one cluster. Figure 3.3 shows an example of border values in a stopping cluster. In Figure 3.3, 4 border points A-D are in red color and Up Border Value is the maximum of all points' Latitude values while Down Border Value is the minimum. Then B and D can be extracted in a similar way as A and C. Any two or three of them (or even all the four points) can be the same point. It is noteworthy that if one cluster crosses the Greenwich meridian, the way to calculate the two border values, $lon1$ and $lon2$, will be changed.

After extracting the 4 border values, the area of the border (blue border lines shown in Figure 3.3) is calculated. With this area result and the *eps* used for clustering, the sampling number is estimated (lines 6−10). Lastly, we use a random selection strategy to sample the stopping points. However, there still exists some sub-regions with higher density of points compared with other sub-regions in the same cluster

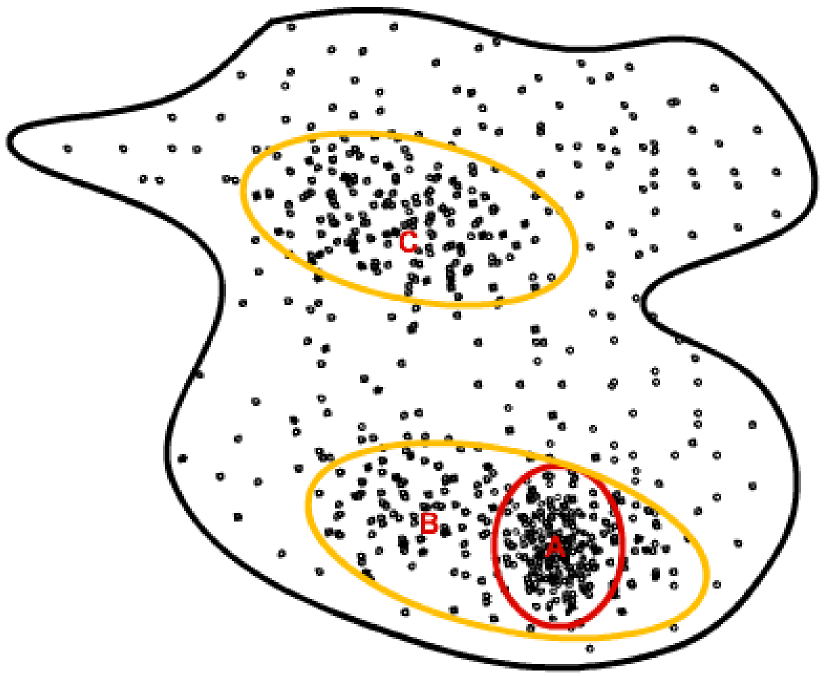


Figure 3.4: An example of a stopping area that simply random sampling cannot work. There are three regions (A,B and C) with higher density than the whole area's average density.

and as a result, random selection tends to select points from these higher-density micro-regions.

An example is shown in Figure 3.4. Figure 3.4 shows a stopping area with three higher densely sub-regions and all the black points are the stopping points of the clustering result. We can see that the sub-region A has the highest density while B and C have relatively lower density which are still higher than the whole average. So in this example, if we simply employ a random selection technique, the majority of the final points shall be extracted from region A and then regions B and C. There might be only a limited number of points that are sampled from the rest part of the whole stopping area, which is not supposed to happen in terms of representing the whole area's shape. As a consequence, we need to modulate the sampling process to stratified sampling to give each sub-region equal probability to be selected. This can be easily done with lines $11-16$; the distance condition in line 14 can be defined by *eps* as well. The algorithm has linear time complexity.

Thus, with the Gravity Vectors and Sampled Stopping Points extracted from the moving and stopping AIS data separately, the anomaly detection task can be conducted, the techniques for which are presented in the next chapter.

Chapter 4

Anomaly Detection

In this chapter, we present our anomaly detection model. Similar to the normal traffic pattern extraction model, we also treat stopping and moving separately. A ship trajectory in a particular area, especially in the near-port areas, can consist of both stopping points and moving points. Hence, given one incoming trajectory data set (one sequence of trajectory points with same identification), every point in this trajectory should first be labeled as stopping or moving according to the SOG threshold (0.5 knots) and then be checked for abnormality. In the following part of the section, three division distances are first presented in Section 4.1 and then our abnormal detection model is introduced in Section 4.2.

4.1 Three Division Distances

In this section, three specialized division distances are proposed to label every data point,in our approach. Absolute Division Distance (ADD), Relative Division Distance (RDD) and Cosine Division Distance (CDD). ADD is employed in the stopping points abnormality detection phase while RDD and CDD are used for the moving part. The definitions of the three division distances are given in the following. Here we use **target point**s to represent the points of a new coming trajectory to be labeled.

The **Absolute Division Distance** between a target point P_t and a Sampled Stopping Point P_s is defined as:

$$D_{absolute} = Distance((P_t.Lat, P_t.Lon), (P_s.Lat, P_s.Lon)) \tag{4.1}$$

As can be seen in Definition 4.1, ADD is actually the Geographical Distance [56] between Latitude and Longitude values of the target point(P_t) and the sampled stopping point(P_s).

The **Relative Division Distance** between a target point P_t and a Gravity Vector GV is defined as:

$$D_{relative} = \frac{Distance((P_t.Lat, P_t.Lon), (GV.Lat, GV.Lon))}{GV.MedianDistance} \tag{4.2}$$

For moving trajectory points, we adopt RDD ($D_{relative}$) rather than ADD ($D_{absolute}$) because the normal moving lanes could have different widths according to their surrounding geographical environment. The routes in a narrow strait can be more cramped than those in the open sea. Relative distance, the ratio of a point's distance from the centroid to the median distance of all the points in one cluster from the centroid, is one efficient metric in clustering-based approaches for detecting outliers [54]. We replace the centroid of each cluster with our Gravity Vectors, which can be regarded as surrogates for a centroid when we only consider Longitude and Latitude. As stated in [35] and previous section, every cluster can have more than one GV, in which case the median distance of one GV can be used to measure the width variance of a moving cluster. Another work that confirms this approach is presented by Ethiene *et al.* [16] in which it is shown that the choice of the statistical decile used to compute the spatio-temporal channel can give a tolerable estimate of this channel's width.

The next division distance is CDD, which is employed to involve direction and speed of moving objects. The **Cosine Division Distance** between a target point P_t and a Gravity Vector GV is defined as:

$$D_{cosine} = \cos\alpha \times \frac{min(P_t.SOG, GV.SOG)}{max(P_t.SOG, GV.SOG)} \tag{4.3}$$

where:

α is the angle between the two directions, that is, the difference between P_t's COG and GV's COG.

The intuition behind CDD is to combine the angle ($\leq 180°$) between the two directions (angle α is defined by COG differences between P_t and GV) and the difference between the two speeds. Figure 4.1 shows two abnormal cases that consider

COG and SOG. In Figure 4.1(a), we can see that the speeds of the two vectors are in the same length L while the angle a is too large. This could be explained as a ship crossing a normal lane in a nearly opposite direction, which may cause a collision. In Figure 4.1(b), the speed of the target point P_t is much lower than GV's although they have similar COG. In this case, the slowly sailing vessel may also be in a high risk of collision in relation to other ships moving at a normal rate of speed. As a consequence, Cosine Division Distance (CDD) is proposed. It can be easily seen that $cos\ \alpha$ accounts for the angles' difference and the ratio between two SOGs can reflect the speeds' difference.

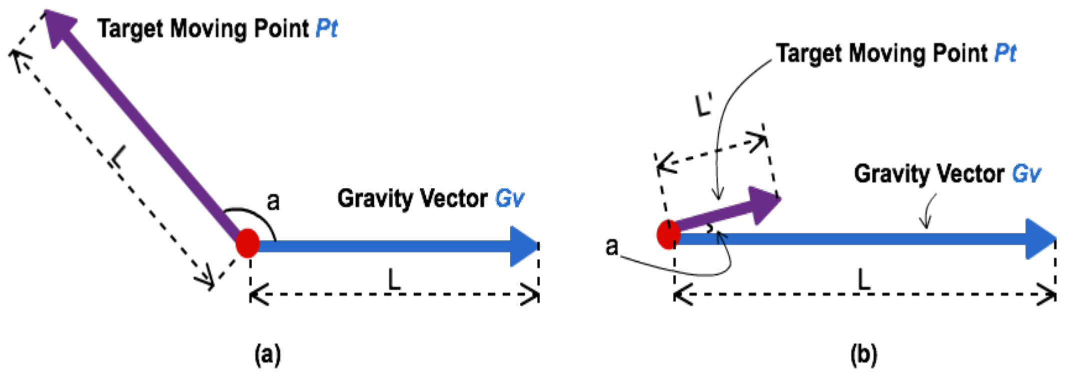


Figure 4.1: Two Abnormal Cases after considering COG and SOG

The Cosine Division Distance proposed in this work combines both COG and SOG in its calculation. A possible alternative is to calculate the distance of each component separately and then check the normality, but this alternative may increase the misclassification error. One example is presented in Figure 4.2. In this scenario, it is assumed that a certain point P_t is in the middle of two different grids belonging to different clusters. If this approach of separately calculating each component is employed, the target point, P_t, will first be considered normal with respect to G_v' on the basis of direction, and also with respect to G_v on the basis of length (value of speed). However, P_t is an abnormal point because it is in the direction of G_v', but with a much faster speed.

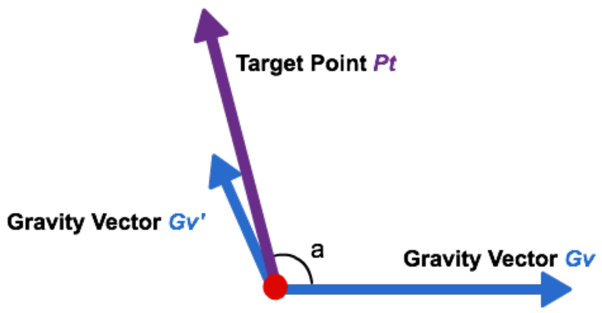


Figure 4.2: One case that the simple combination of the two division distances cannot work. There are two moving clusters in one specific area and the two clusters are with two different average COGs.

4.2 Anomaly Detection Model

After having the three division distances defined, we can present our anomaly detection model.

As shown in Algorithm 3, the anomaly detection process is completed in two steps. Lines 3-6 employ ADD to label the stopping points of the trajectory and Lines 7-14 use RDD and CDD to decide the moving points' labels. Lastly, the ratio of abnormal points to the number of all points is returned as the value of abnormality (Lines 15-18). The abnormality can be interpreted as a confidence ratio in a trajectory point being abnormal and it is beneficial for the end users to choose a level of confidence while retrieving the abnormal results of the whole system.

This algorithm requires three thresholds as inputs, which can be estimated through experiments. In this work, we first take out one month of data in a specific area and calculate all the division distances, then select specific values based on distribution and quartile analysis.

Figure 4.3 shows the distribution of the maximum CDD of moving points in the area of Juan de Fuca Strait. Figure 4.4 illustrates the CDF (Cumulative Distribution Function) of the maximum CDD in the area. The red dotted line in Figure 4.3 is the 5% separation line, which means the percentage of all the CDDs not greater than the separation coordinate is 5%. Similarly, in Figure 4.4, the red dotted line is also a 5% separation line and we can see the corresponding coordinate in y-axis is 0.05. It is obvious that over 90% of the data points' CDD are greater than 0.5. Thus, in this case, we can choose 0.5 or a smaller value as the CDD threshold for this area and

Algorithm 3 Detect abnormality of the target trajectory

```
Input: (1) The target trajectory dataset, $D$; (2) The lists of Sampled Stopping
    Points and Gravity Vectors from the previous model, $SSP$ and $GV$; (3) Three
    thresholds, $add\_threshold$, $rdd\_threshold$ and $cdd\_threshold$
Output: The abnormality rate, $abnormality$
 1: Separate D into two sub-datasets based on the speed threshold, moving dataset
    $D\_m$ and stopping dataset $D\_s$
 2: Initialize all labels of points in $D\_m$ and $D\_s$ as Normal
                                        ▷ label stopping points of the target trajectory
 3: for each data point $S$ in $D\_s$ do
 4:     $ADD\_s \leftarrow$ minimum(ADD($S$,$SSP$))
 5:     if $ADD\_s > add\_threshold$ then
 6:         $S.label \leftarrow$ Abnormal
                                        ▷ label moving points of the target trajectory
 7: for each data point $M$ in $D\_m$ do
 8:     $RDD\_m \leftarrow$ minimum(RDD($M$,$GV$))
 9:     if $RDD\_m > rdd\_threshold$ then
10:         $M.label \leftarrow$ Abnormal
11:     else
12:         $CDD\_m \leftarrow$ maximum(CDD($M$,$GV$))
13:         if $CDD\_m < cdd\_threshold$ then
14:             $M.label \leftarrow$ Abnormal
15: $count\_ab \leftarrow$ the number of the abnormal points in $D$
16: $count\_all \leftarrow$ the total number of all points in $D$
17: $abnormality \leftarrow count\_ab/count\_all$
18: return $abnormality$
```

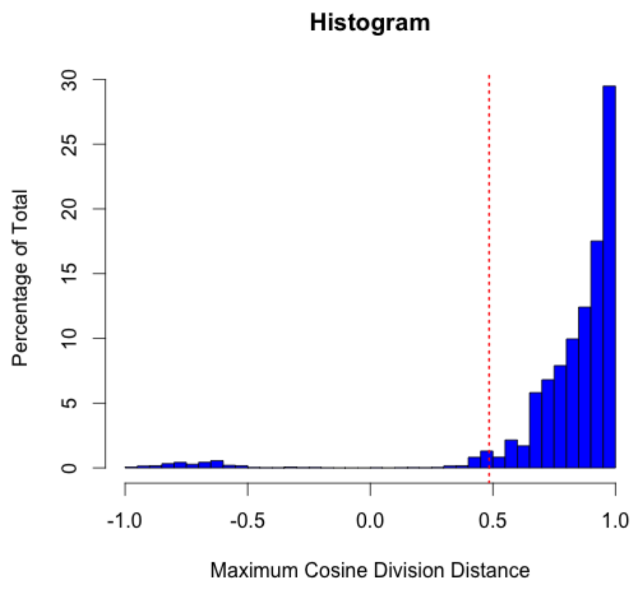


Figure 4.3: The maximum CDD distribution in the area of Juan de Fuca Strait.

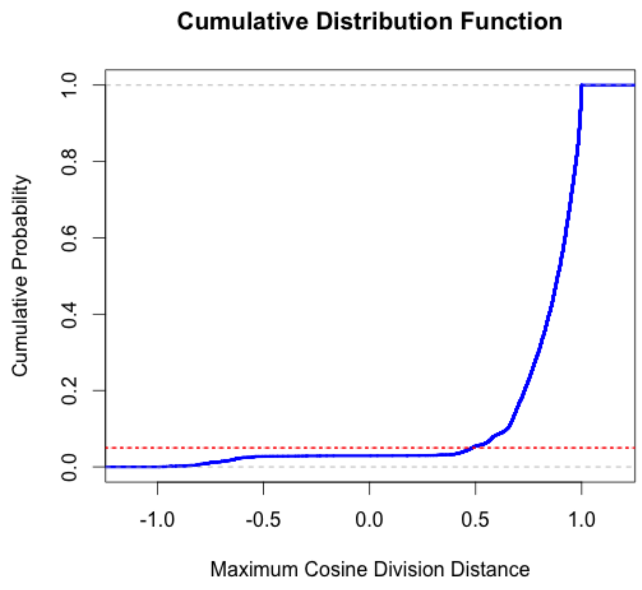


Figure 4.4: The Cumulative Distribution Function of the maximum CDD in the area of Juan de Fuca Strait.

then when a new trajectory dataset is given, we can use this value as the basis for anomaly detection.

Chapter 5

Evaluation

In this chapter, experiments are done to evaluate the effectiveness of the proposed approach. First, to evaluate the normal traffic patterns extraction model, regions of Juan de Fuca Strait (Section 5.1.1) and Los Angeles Long Beach (Section 5.1.2) are selected and the results are presented in Section 5.1. Then in Section 5.2, we conducted two other experiments in the same region of Juan de Fuca Strait. The first one is conducted with the non-labeled data while the second one is done after labelling the data. The results of the first experiment (Section 5.2.1) are shown visually and the second experiment (Section 5.2.2) compares our models results with the labels by the expert.

5.1 Normal Traffic Patterns Extraction Model

In this section, we evaluate the effectiveness of our maritime normal patterns detection model in two regions. The data set contains non-anonymized messages from ships in the region of Juan de Fuca Strait and the region of Los Angeles Long Beach. The movement rules for port areas defined by IMO achieved from the IMO publication [24] are shown in Figure 5.1 and 5.4. Evaluation of the normal patterns detection model is related to the problem of evaluating clusters quality and this is always not a trivial task. Our strategy is to use our approach to generate the lanes and then compare the results with these rules.

5.1.1 Juan de Fuca Strait

The Strait of Juan de Fuca separates the south coast of Vancouver Island from the north coast of State of Washington. The entrance of it lies between Cape Flattery (48°23′N.,124°44′W.) and Carmanah Point (48°37′N.,124°45′W.) [9]. Figure 5.1 shows a map of this area and the predefined rules for it.

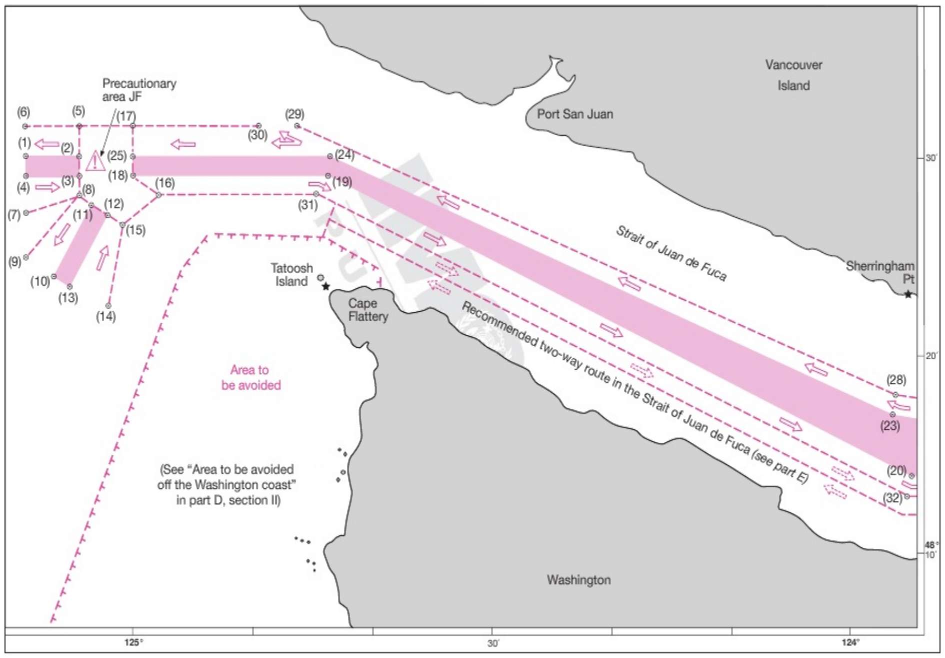


Figure 5.1: Juan de Fuca Strait and its approaches (west) [24]

The data set prepared for this experiment is two-months of trajectory data from November 1 to December 31 in 2012. It consists of 67,850 trajectory points. The whole dataset is not used; instead 46,000 records (40,000 moving points and 6,000 stopping points distinguished by the SOG threshold 0.5 knot) are selected randomly.

After applying the two clustering algorithms, 14 different clusters including 13 moving clusters and 1 stopping cluster are extracted. The result can be seen in Figure 5.2. Points in blue are the original traffic points while others are the cluster points. The size (total number of the points composing the clusters) of moving clustering results is 20,817 while that of the stopping cluster is 4,800.

Figure 5.3 shows the gravity vectors of the moving clusters and the sampled stopping point of the stopping cluster. After this extraction phase, only 302 points are generated which include 301 GVs and only one SSP. From the two figures we can see that the clustering results can reflect the traffic trends in this strait and the GVs as well as the SSP can represent the clustering results too.

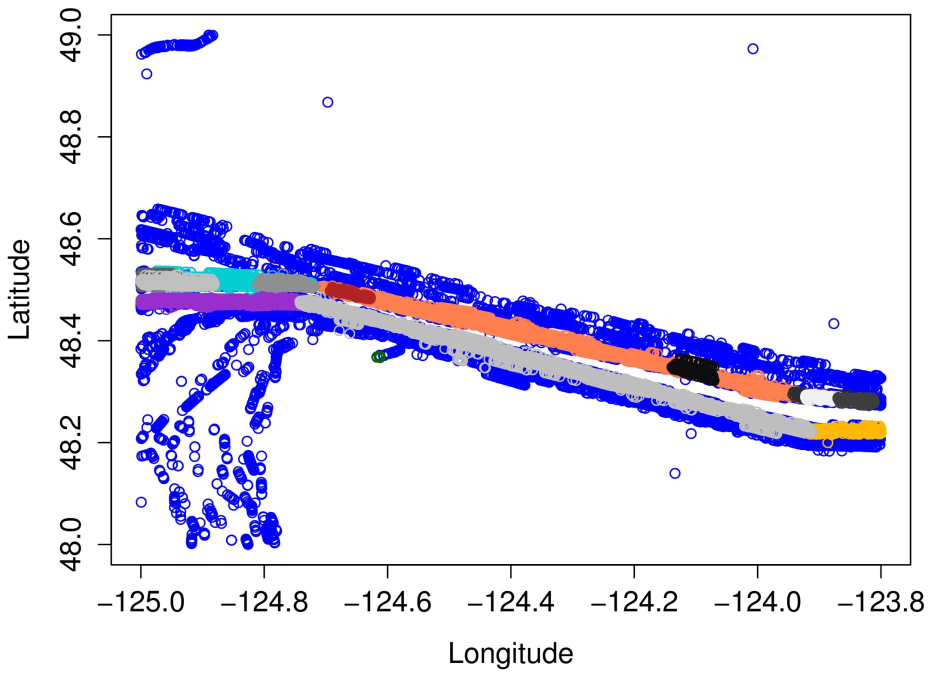


Figure 5.2: The clustering results of JUAN DE FUCA STRAIT area. Different colors stand for different clusters. The clusters in dark green is stopping clusters.

5.1.2 Los Angeles Long Beach

The approach has also been tested in the area of Los Angeles Long Beach, the traffic of which is more complicated and heavier. Figure 5.4 shows a map of the area with two rules defined. In this experiment, the same period (November 1 to December 31 in year 2012) dataset has been prepared. There are 327,694 records (99,937 moving points and 227,757 stopping points) in this dataset. We adopt all the moving points in this dataset for moving clusters generation and the first 20,000 stopping points for stopping area generation.

Figure 5.5 shows the result of stopping clusters sampling procedure in the area of Los Angeles Long Beach. There are 7 different stopping clusters in Figure 5.5(a) with 19,089 stopping points generated by DBSCAN algorithm [15]. Then after applying Algorithm 2, only 26 Stopping Sampled Points (shown in green color in Figure 5.5(b) and 5.5(c)) are selected with excellent quality in terms of the representativeness.

Then the algorithm DBSCANSD is applied to moving points. Finally, 48,404

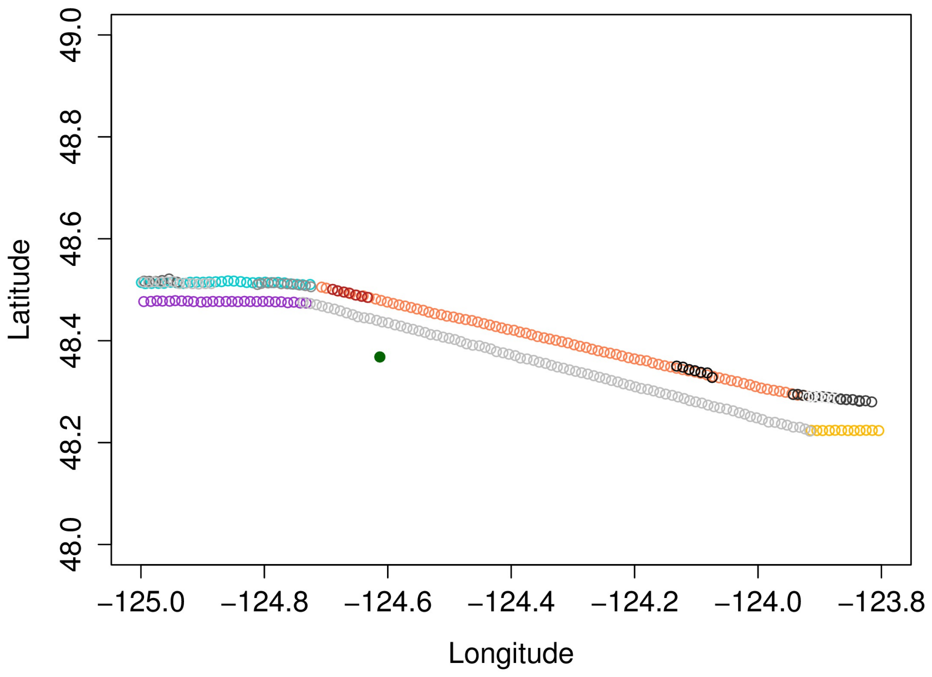


Figure 5.3: The Gravity Vectors and Sampled Stopping Points extracted from the clusters in Juan de Fuca Strait area.

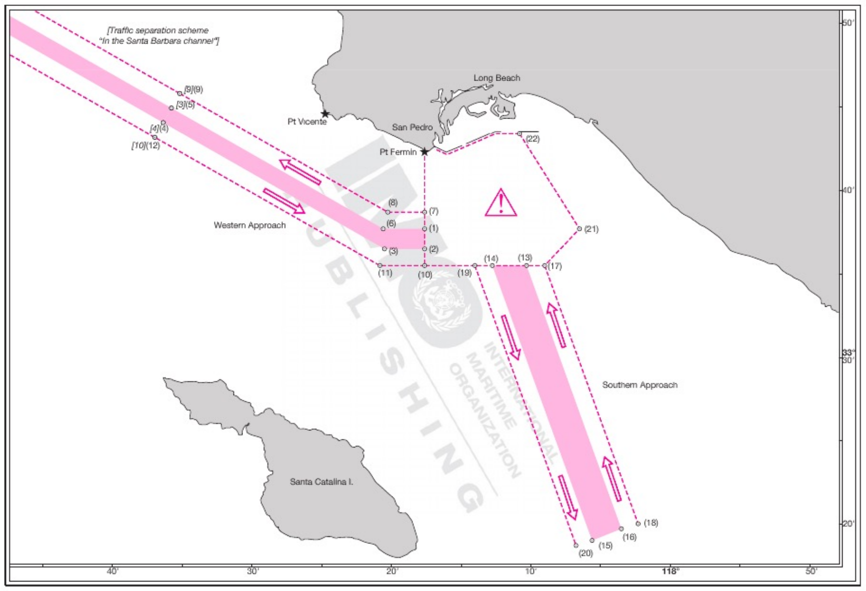
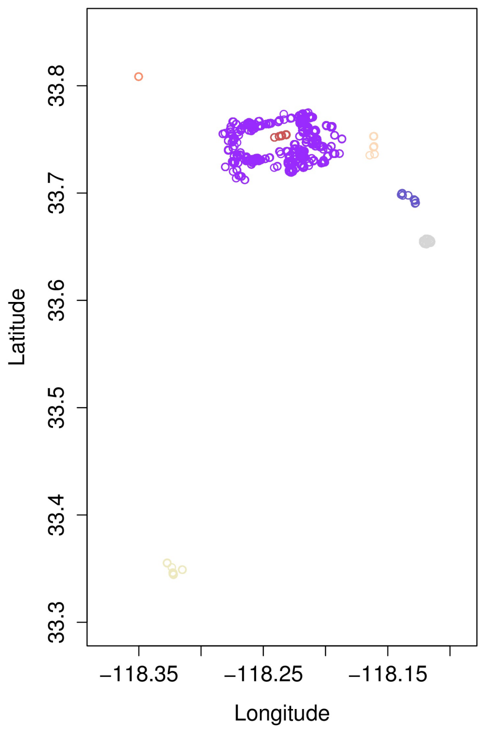
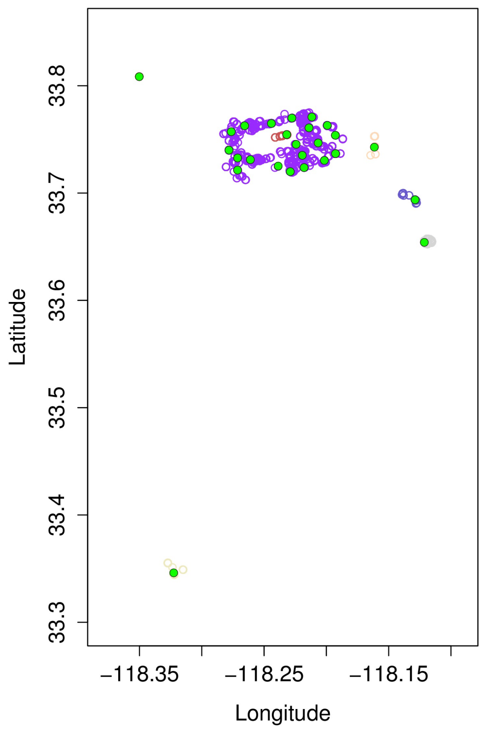


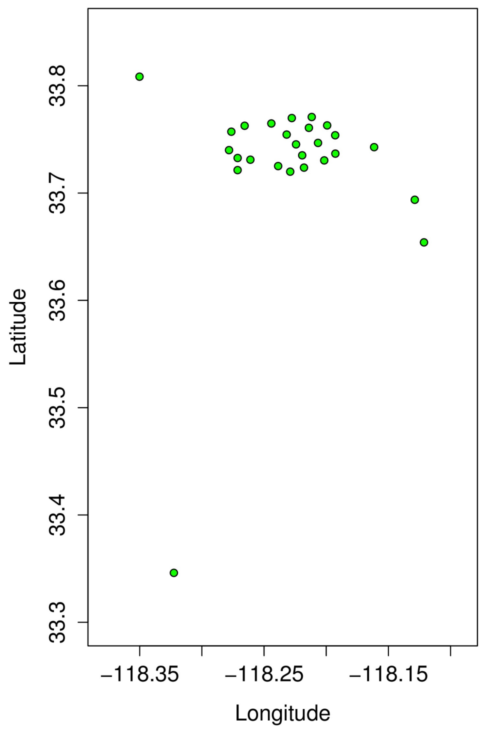
Figure 5.4: Map of Los Angeles Long Beach and the rules defined [24]



(a) Stopping clusters by DBSCAN [15]



(b) Sampled Stopping Points generating



(c) Sampled Stopping Points generated

Figure 5.5: Extract Stopping Sampling Points (SSP) from stopping clusters in Los Angeles Port Area. The stopping clusters are first extracted using DBSCAN [15] algorithm shown in (a) and different colors stand for different clusters. Then Algorithm 2 is used for getting the SSPs (shown in figures (b) and (c))

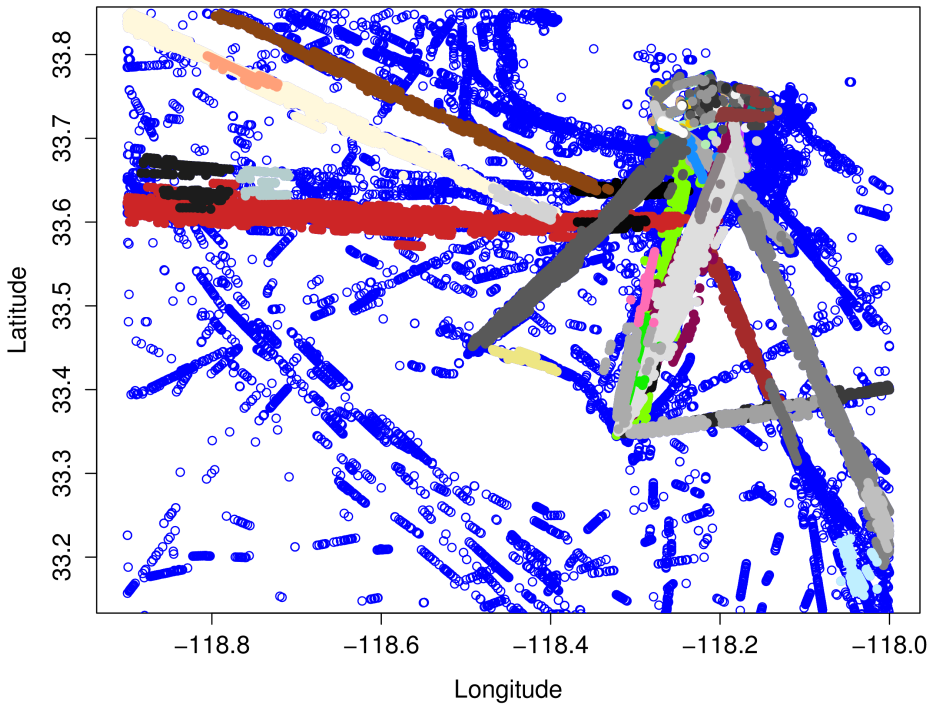


Figure 5.6: The clustering results of Los Angeles Long Beach area. This only shows the moving points(in blue) and the corresponding moving clusters (in different colors).

points are selected to form 51 moving clusters from the original 99,937 points. This result is shown in Figure 5.6. From the figure we can see that in the stopping area, there are multiple moving clusters too. This is because vessels in the area have to change their headings frequently to arrive at the specified anchor location and our algorithm will treat this curve-shape movement as multiple moving clusters with slight COG differences. The last step for this normal traffic pattern extraction is to calculate the Gravity Vectors of the moving clusters. The GVs of the area can be seen in Figure 5.7 and the SSPs (filled circles in dark green color) extracted in the previous step are also shown in the same figure .

5.2 Anomaly Detection Model

In this section, we evaluate the effectiveness of our maritime anomaly detection model in the region of Juan de Fuca Strait. The evaluation work contains two parts, the

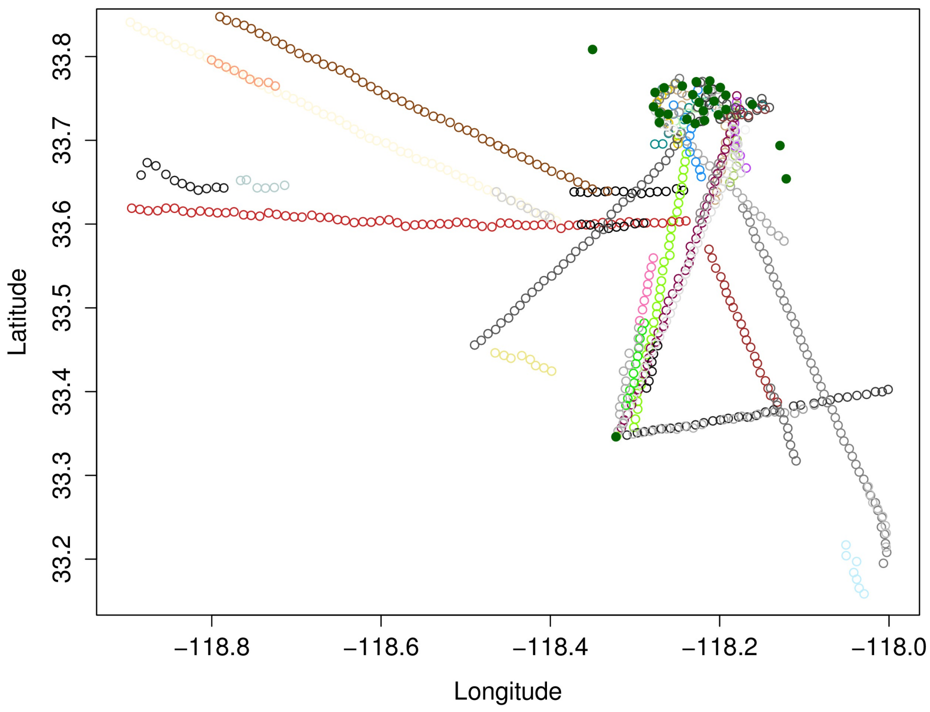


Figure 5.7: The Gravity Vectors and Sampled Stopping Points extracted from the clusters in Los Angeles Long Beach area.

first one is conducted with the non-labeled data while the second one is done after labelling the data. The results of the first experiment are shown visually and the second experiment compares our model's results with the labels by the expert.

The data set which was prepared for normal traffic patterns extraction phase comprises two months of trajectory data from November 1 to December 31 in 2012 and contains 67,850 trajectory points. The whole data set is not used for extracting normal patterns, instead 46,000 records (40,000 moving points and 6,000 stopping points distinguished by the SOG threshold 0.5 knots) are selected and then the rest of the data set are used for estimating the three thresholds for the anomaly detection phase.

Afterwards, to evaluate our anomaly detection model in both experiments, we chose the first half of January (January 1st to January 15th) in 2013, as our target trajectory data set. This second dataset consists of 284 different trajectories with 17,431 points.

5.2.1 Experiment On Unlabeled Data Set

After applying the normal traffic extraction model, 16 different clusters, including 15 moving clusters and 1 stopping cluster are identified. Figure 5.8 shows the gravity vectors of the moving clusters and the sampled stopping point of the stopping cluster. After this extraction phase, only 388 points are generated which include 388 GVs and only one SSP.

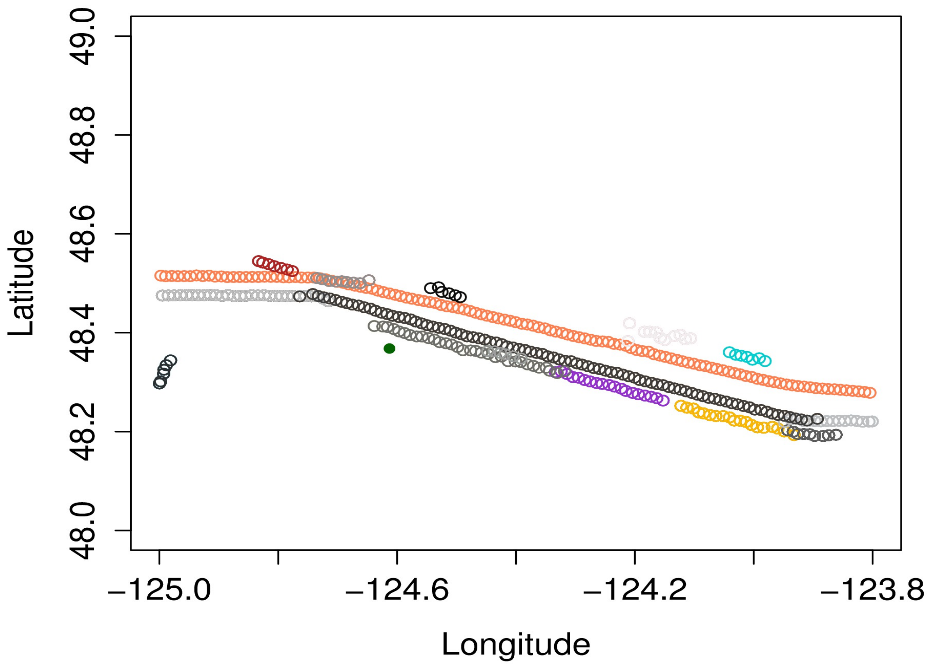


Figure 5.8: The Gravity Vectors (open circles) and Sampled Stopping Points (filled circles) extracted from the clusters in JUAN DE FUCA STRAIT area.

The next step before detecting anomalous trajectories is to estimate the three thresholds (*add_threshold*, *rdd_threshold*, *cdd_threshold*) to be used in Algorithm 3. As stated before, the remaining 23,850 trajectory records are chosen for this phase. There are 10,825 stopping points and 11,025 moving points in this subset.

The quartile values of ADD and RDD of the subset are shown in Table 5.1 (column 1 and column 2). Authors tested different thresholds to be used as anomaly detector. After various tests, for the area of Strait of Juan de Fuca, the best threshold value was to consider 95% of the data as normal in relation to distance, and from this sub-set

Table 5.1: Quartile statistics of the three division distances

Statistic	ADD	RDD	CDD
Min	0.13	0.00537	-0.9937
1st Quartile	3.00	0.70300	0.7642
Median	4.46	1.04000	0.8876
Mean	36.97	1.81500	0.8104
3rd Quartile	6.89	1.58400	0.9612
Max	44250.00	52.86000	0.9999

another 95% of the data to be considered normal in relation to speed and direction. The main objective is to reduce the number of false alarms (vessels considered abnormal, while they are normal), and this filtered data will later be evaluated by a human expert that will give the final decision. The model is flexible to allow changing this threshold value depending on the geographical area under evaluation.

So we choose the sample quantiles of 0.95 for both ADD and RDD. The corresponding thresholds in this case, *add_threshold* and *rdd_threshold* in Algorithm 3, are 97.290 and 5.938. After calculating the RDD threshold, the statistic for CDD is obtained (shown in the 3rd column in Table 5.1). Then we select 0.05 as the possibility to decide the third threshold (0.485) which can be employed as our CDD threshold (*cdd_threshold* in Algorithm 3).

With the extracted normal patterns and the thresholds estimated, we start to evaluate the capacity of detecting abnormal trajectories.

In this step, we first apply our model to the trajectory data points; the labeling results of the data points are shown in Figure 5.9. The red points stand for the GVs and the SSP in this area. The green points are normal, while the blue and purple ones are abnormal. More specifically, a blue point means it is too far away from the corresponding GV or SSP, while a purple point represents that its speed or direction is too aberrant in the specific location. In this case, 1,534 points (872 in blue and 662 in purple) of the 17,431 points are finally considered as abnormal.

After finishing the labeling process, we calculate the abnormality ratio of each trajectory. We use a threshold of 0.5 as the minimum confidence rate to extract the abnormal trajectories. Noteworthy, the number 0.5 is adjustable and was chosen

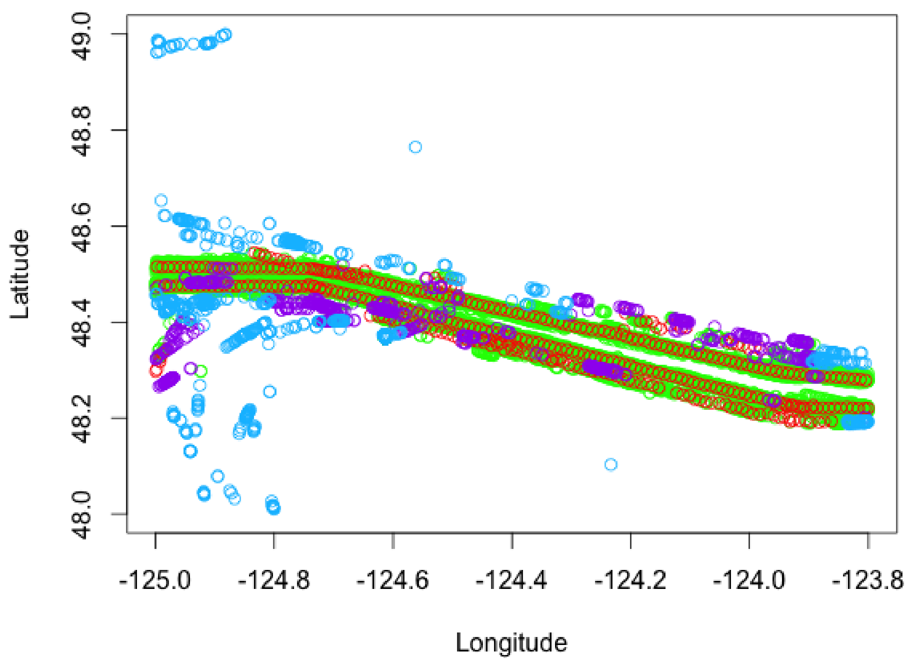


Figure 5.9: The anomaly labeling results of the trajectory data points in JUAN DE FUCA STRAIT area. GVs and SSPs are in red and the normal points are in green. The two types of abnormal points are in blue (abnormal in relation to ADD or RDD) and purple (abnormal in relation to CDD).

based only on the experiments to reduce the presence of false alarms (normal trajectories considered abnormal by our model). The result is that 22 trajectories are labeled abnormal among the total 284 trajectories, in other words, the abnormality rates of the 22 trajectories are over 0.5. From Figures 5.10 to 5.15, six examples of abnormal cases are illustrated. As can be seen in the figures, both purple points and blue points contribute to the final abnormality of one trajectory.

5.2.2 Experiment On Labeled Data Set

In this experiment, the same data set is labeled by an expert who has multiple years of experience in maritime data analysis. The labelling process is not biased by our model's results since only the raw AIS data has been provided to the expert.

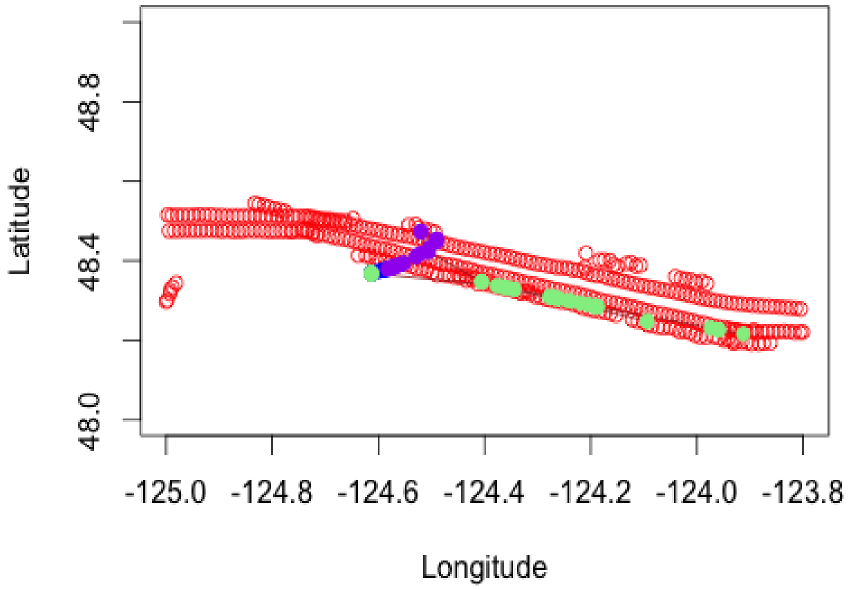


Figure 5.10: First example of the abnormal trajectories detected by our framework.

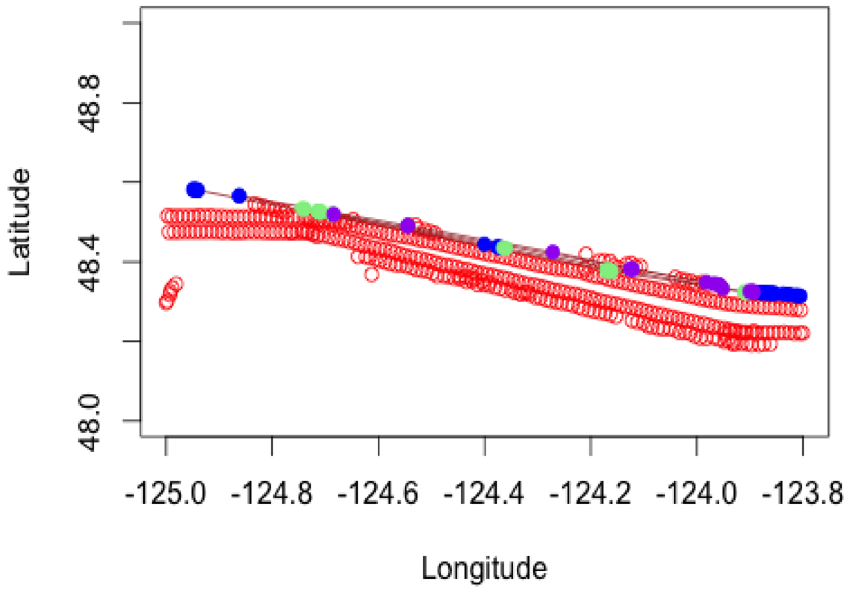


Figure 5.11: Second example of the abnormal trajectories detected by our framework.

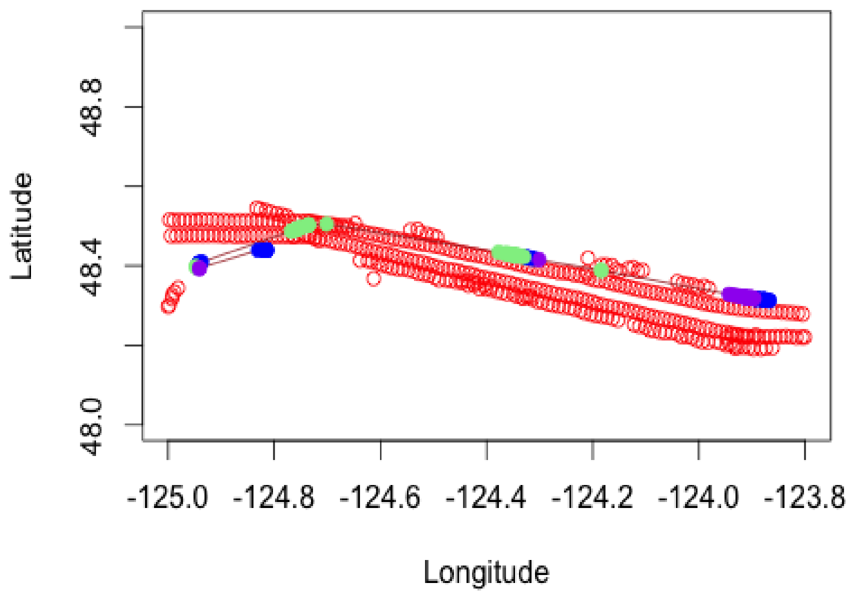


Figure 5.12: Third example of the abnormal trajectories detected by our framework.

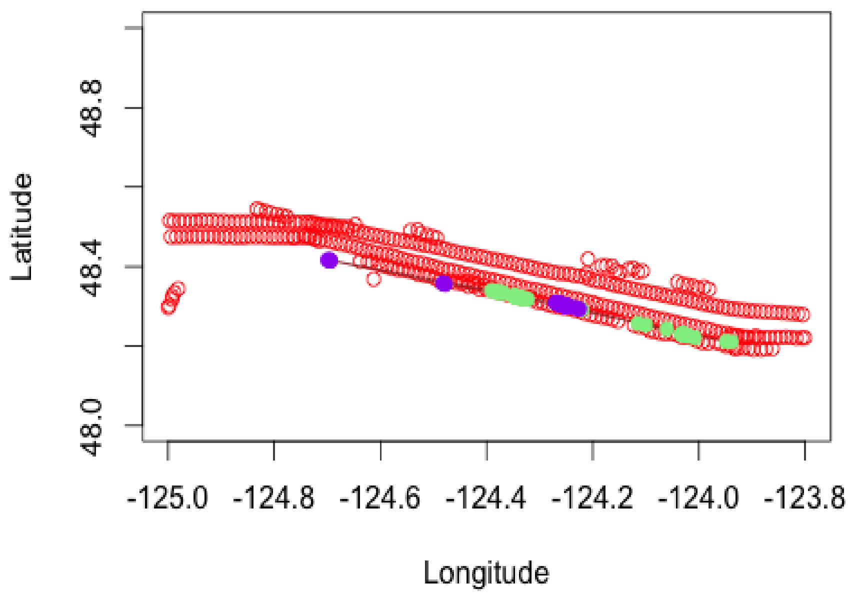


Figure 5.13: Fourth example of the abnormal trajectories detected by our framework.

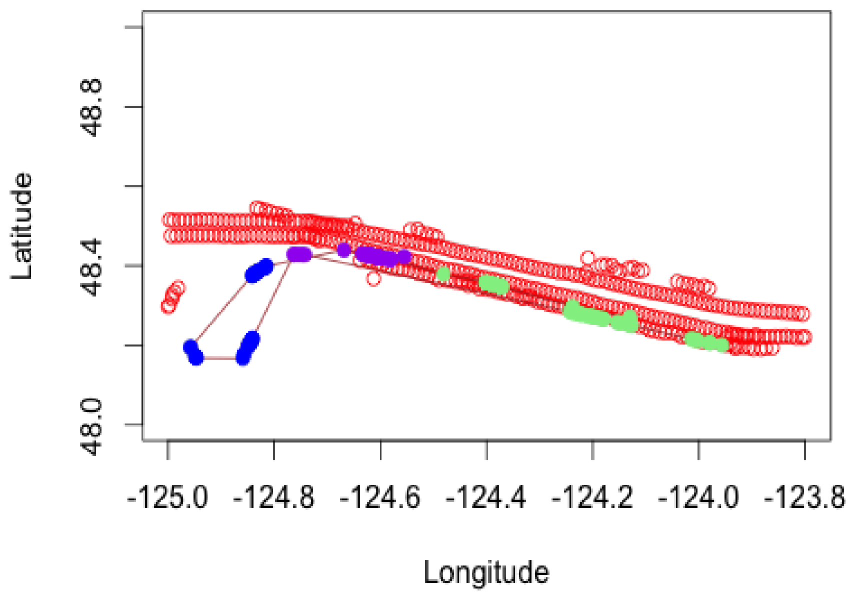


Figure 5.14: Fifth example of the abnormal trajectories detected by our framework.

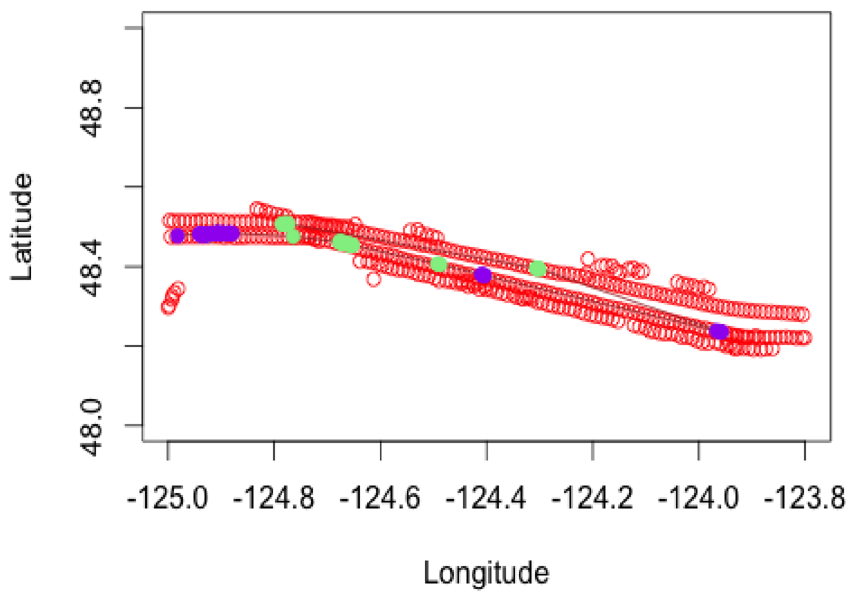


Figure 5.15: Sixth example of the abnormal trajectories detected by our framework.

Table 5.2: Labels and descriptions by the expert

Label	Description
Bad_Pos	Track contains questionable point, far outside track, looks like bad GPS return
In_Excl_Zn	Track has significant portion within the exclusionary zone between traffic lanes
XING_TSS	Track appears to be crossing lanes of TSS [1]
XING_NShor	Track appears to be crossing lanes of near shore two way traffic area
Odd_Mvmt	Track shows unusual movement without other explanation
Leave_Lane	Vessel was in traffic lane, then veered outside
Harbour	Track seems to describe in-harbour navigation or moored vessel
Normal	Normal Movement

A track division method is employed by the expert before labelling. More specifically, the AIS data points are divided into distinct tracks on the basis of vessel ID (MMSI), and where temporally sequential points are separated by no more than 4.5 minutes of time. Thus one track, as defined above, may be divided into multiple-sub tracks on the basis of time. Additionally, this process can result in tracks comprised of single points. For the one-point track case, the length of the track is 0 nautical mile and the track is not assigned with any labels. The final labels and their descriptions provided by the expert are shown in Table 5.2.

The labels assigned by the expert are not based on the points; instead, they are based on the whole sub-tracks. In other words, as long as one sub-track shows an anomalous pattern, the whole set of points inside the sub-track will be assigned as one kind of abnormal label. Another noteworthy point is that the expert has not taken SOG (speed) into account except for Harbour behavior during his labeling process. That is, whether the speed of the vessel is too fast or too slow near the lane is not considered, but our algorithm can take this into account.

From Table 5.2, we can firstly assume the label of *Normal* as normal patterns based on the description. Then the label *Harbour* can also be considered as a normal case because it is reasonable for a vessel to moor in harbour. Lastly, we observe that all the sub-tracks with the label of *Leave_Lane* only have tiny changes from their

Table 5.3: Confusion Matrix I

	Abnormal (Our Model)	Normal (Our Model)
Abnormal (Expert's Label)	4	10
Normal (Expert's Label)	127	1301

route and they still navigate strictly within the normal lanes. So we also classify the label of *Leave_Lane* as normal.

After dividing the 284 tracks (284 different MMSIs), 2,122 sub-tracks are generated. Among them, 680 sub-tracks contain only one point (Length=0 nautical mile). Then 14 sub-tracks are classified as abnormal labels (other than *Leave_Lane*, see Table 5.2) by the expert and the remaining 1,428 tracks are all normal patterns. Thus we can see that the data set is a highly imbalanced data set which can make our work extremely challenging.

At this point we can use our algorithm to label the data set and compare the results with the expert's labels. To compare the results, we first apply the same division method to separate the tracks. We can then use a threshold to decide the whole sub-track's label. In this experiment, we employ 60% as the threshold value. That means that if the portion of abnormal points in one track is greater than 60%, we will label this whole track as abnormal. Using this approach, we find that 131 sub-tracks are classified as abnormal and the remaining 1,311 sub-tracks are normal. Table 5.3 is the confusion matrix for the experiment.

From Table 5.3 we can see that 4 sub-tracks are classified as abnormal by both the expert and our model and 1301 sub-tracks are classified as normal by both too. On the other hand, another 10 sub-tracks are labeled as abnormal by the expert while normal by our model. The remaining 127 sub-tracks are classified as abnormal by our model while normal by the expert. And the overall accuracy of this detection result is 90.49%.

To improve the result, we investigate the 14 sub-tracks designated as abnormal by the expert. Among these, we find that the 4 which are further designated as abnormal by the algorithm are so labeled solely because of their direction. After investigating other tracks, we find that even if the ships deviate far from the lane the

Table 5.4: Confusion Matrix II

	Abnormal (Our Model)	Normal (Our Model)
Abnormal (Expert's Label)	4	10
Normal (Expert's Label)	52	1376

expert may still label them as normal. The intuition behind this is straight forward, the labelling process is based on Traffic Separation Scheme (TSS) [1] boundaries and the expert cannot affirm that a trajectory point far from the TSS Boundaries is abnormal. Considering this fact, in the following experiment, we ignore the abnormal labels caused by RDD or ADD during the evaluation and we choose a lower threshold for deciding whole sub-tracks' labels. In the previous experiment we use 60% while we choose 10% here. It should be noted that the threshold can be adjusted based on the input from domain experts. In real-time application, it is not necessary to have the threshold while labelling the new incoming points instead of tracks.

Table 5.4 presents the improved results and we can see that the overall accuracy has been increased from 90.49% to 95.70% while keeping the same recall for abnormal cases.

Chapter 6

Conclusion

In this thesis, a maritime traffic pattern extraction model has been first proposed. The approach first separates the trajectory data set into moving and stopping subsets, and then employ different strategies in corresponding subsets. The main advantage of this work is that two attributes, speed and direction, are taken into account during the clustering phase. In this way, geographically close trajectory points with similar direction and speed can be grouped together to form a cluster. We also present two methods to represent the results of clustering; that is, Gravity Vector for moving clusters and Sampled Stopping Point for stopping clusters.

Another contribution of the proposed normal traffic extraction model is that it can be easily extended for other scenarios. Besides the maritime anomaly detection task addressed in this thesis, the moving trajectories clustering algorithm (DBSCANSD) and Gravity Vectors are also applicable for other trajectory clustering tasks (e.g. vehicle position data, animal movement data, hurricane monitoring data). Thus, more experiments can be done with other domains' data sets to verify our method's applicability in the future study.

To show the effectiveness of our approach, two real data sets from different regions (Juan de Fuca Strait area and Los Angeles Long Beach area) are used. We have compared the generated results with the rules defined by IMO and it shows that the results can be successfully mapped to the rules. This demonstrates that our model can effectively mine normal patterns in the two areas, which is another contribution of this work.

One limitation of the normal traffic extraction model is that it is sensitive to parameters. The model requires four parameters (eps, $MinPts$, $MaxDir$, $MaxSpd$) during the moving traffic patterns extraction phase and two parameters (eps and $MinPts$) for stopping area generation. So, if the model is applied in another area, a new set of parameters need to be given to achieve a good clustering result. Thus, to achieve satisfactory clustering results, it is necessary to possess a good maritime background. Specifically, when we distinguish moving points from stopping points in the first step, a reasonable SOG threshold is required. Here in this thesis, we adopt 0.5 knots as the threshold for distinguishing stopping and moving. Besides, before applying DBSCANSD to moving trajectory points, we cannot simply assume the tolerated angle ($MaxDir$) between two close points' directions and the tolerated difference ($MaxSpd$) between two close points' speeds.

The Gravity Vectors and Sampled Stopping Points detected by the proposed approach can help authorities update their rules (e.g. re-position the buoy markers). Some other research studies like route planning and vessel position prediction can also be conducted based on our model's results.

Based on the Gravity Vectors and Sampled Stopping Points extracted from the clustering phase, we then propose the anomaly detection model for maritime traffic data. An abnormality detection algorithm is presented based on three division distances (Absolute Division Distance, Relative Division Distance and Cosine Division Distance). This model is a fairly straightforward point-based approach, and is capable of handling complicated maritime traffic situations. One advantage is that the clustering process is associated with TSS Boundaries [1], which can assure a reliable clustering result for the following anomaly detection work. Another critical advantage is that besides position information (Longitude and Latitude), the model can also take speed and direction into account while deciding the abnormality of a single trajectory point. The model is also flexible enough for analysts to set their own thresholds to label whole trajectories.

In order to evaluate the effectiveness of the anomaly model, a highly imbalanced data set from Juan de Fuca Strait area is used. There are 2,122 trajectories while only 14 of them are abnormal (imbalance rate $\approx 0.66\%$). Fortunately, as shown in Table 5.4, our model can detect 28.57% of the abnormal tracks while maintaining a relatively high overall accuracy (95.70%).

6.1 Future Work

One of possible directions of future work is to improve the efficiency of the presented algorithms by applying more sophisticated data structures (like spatial indexes, for example). This issue is important in the context of big sizes of the processed data. In this thesis the focus was not on optimising the efficiency of the algorithms but to present the ideas. We envisage algorithm optimisation in the future work.

One may argue about the use of an unsupervised learning approach instead of supervised learning methods. The reason for this is that there is no sufficient labeled data to train an effective model. Once we get enough labeled data we will explore other supervised learning techniques for this anomaly detection task. For example, we can try some classification algorithms designed for handling imbalanced data sets and the proposed specialized division distances can be used as the features of the classification models. Then comparisons between the results and our model's could be done to illustrate the effectiveness of the proposed division distances.

One limitation of the anomaly detection evaluation model is that the labelling process applied by the expert does not consider speed while our work takes this into account. This leads to another possible future direction, that is, more work should be done by the experts while labelling the data set to consider speed. In this way, the false alert rate can also be reduced while the recall of the anomaly trajectories is improved. Another limitation is that the experiments are only conducted with data from Juan de Fuca Strait area and as a result, more experiments in other regions should be done to better illustrate the effectiveness of our approach.

As discussed before, this is a point-based method and the output is two sets of representative vectors (Gravity Vectors and Sampled Stopping Points). As a consequence, one limitation of our clustering algorithm, compared to trajectory-based ones, is that ours cannot take into account the behavior over time. One example of this limitation is illustrated in Figure 6.1. Assume there are two main lanes in the region and they are Lane A and Lane B (shown in red color and purple color separately). Then a vessel enters the region and follows a path same as Trajectory C. Since our framework treats the trajectories as sequences of data points, the trajectory will be divided into two sub-sequences and during anomaly detection phase, the first part will be associated with Lane B while the second part will be associated with Lane A. In this sense, the new trajectory will be assigned as normal. However, the vessel is not supposed to change its navigational lane once it starts its trip on Lane B because there are only two shipping routes pre-defined in this region. So in this situation, our

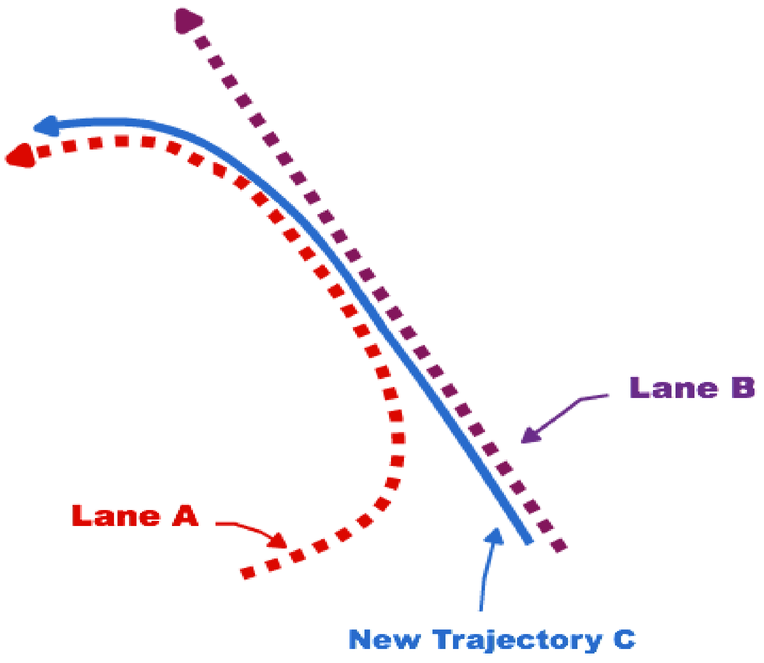


Figure 6.1: One situation that our algorithm cannot detect the anomalies. Two normal lanes (Lane A and Lane B) detected by DBSCANSD are in red and purple. A new anomaly trajectory C is in blue.

model fails to detect the new anomaly trajectory C, and this can be improved after considering time, which is one of our future works.

The clustering results of the first phase can be used for predicting vessels' following routes. Since the GVs extracted carry not only the position information (Longitude and Latitude) but also the kinematic characteristics (course and average speed), it is feasible for researchers to obtain the corresponding expected speed and heading for the specific predicted position. Two potential ways for handling this can be association rules mining techniques or Bayesian approaches. As stated in Chapter 2, anomaly detection approaches based on predictive models usually predict future status information of a particular vessel and then compare the real data with the prediction to decide the abnormality. Thus, to improve the performance of the model, an ensemble model which incorporates the predictive model can be developed in the future. Specifically, once a trajectory point needs to be labeled, we can consider both its anomalous score (output of our proposed model) and its deviation from the predicted position to get a more confident result.

Bibliography

[1] Colreg.2/circ.57 new and amended existing traffic separation schemes. `http://www.imo.org/blast/blastDataHelper.asp?data_id=14761&filename=57.pdf`. Accessed: 2006-05-26.

[2] *IALA Guidelines on the Universal Automatic Identification System (AIS), Volume 1, Part II-Technical Issues Edition 1.1.* IALA/AISM-20ter rue Schnapper, 78100 Saint Germain en Laye, France, 2002.

[3] Charu C Aggarwal and Chandan K Reddy. *Data Clustering: Algorithms and Applications.* CRC Press, 2013.

[4] Jrgen Ahlberg. Candide-3 - an updated parameterised face. Technical report, 2001.

[5] Mihael Ankerst, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: ordering points to identify the clustering structure. In *ACM Sigmod Record*, volume 28, pages 49–60. ACM, 1999.

[6] Heather Ball. *Satellite AIS for Dummies.* Wiley, Mississauga, ON, 2013.

[7] Casey Hilliard Behrouz Haji Soleimani, Erico N. De Souza and Stan Matwin. Anomaly detection in maritime data based on geometrical analysis of trajectories. In *Information Fusion, 2015 18th International Conference on.* IEEE, 2015.

[8] Manuel Blum, Robert W. Floyd, Vaughan Pratt, Ronald L. Rivest, and Robert E. Tarjan. Time bounds for selection. *J. Comput. Syst. Sci.*, 7(4):448–461, August 1973.

[9] United States. Defense Mapping Agency. Hydrographic/Topographic Center. *Sailing Directions (enroute) British Columbia.* Pub. (United States. Defense Mapping Agency. Hydrographic/Topographic Center). Defense Mapping Agency, Hydrographic/Topographic Center, 2012.

[10] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(3):15, 2009.

[11] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[12] Anders Dahlbom and Lars Niklasson. Trajectory clustering for coastal surveillance. In *Information Fusion, 2007 10th International Conference on*, pages 1–8. IEEE, 2007.

[13] Gerben Klaas Dirk De Vries and Maarten Van Someren. Machine learning for vessel trajectories using compression, alignments and domain knowledge. *Expert Systems with Applications*, 39(18):13426–13439, 2012.

[14] Rina Dechter and Judea Pearl. Generalized best-first search strategies and the optimality of a*. *J. ACM*, 32(3):505–536, July 1985.

[15] Martin Ester, Hans peter Kriegel, Jrg S, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. pages 226–231. AAAI Press, 1996.

[16] Laurent Etienne and Thomas Devogele. Spatio-temporal trajectory analysis of mobile objects following the same itinerary. *Advances in Geo-Spatial Information Science*, 17(1):11–34, 2012.

[17] Nivan Ferreira, James T. Klosowski, Carlos Eduardo Scheidegger, and Cláudio T. Silva. Vector field k-means: Clustering trajectories by fitting multiple vector fields. *CoRR*, abs/1208.5801, 2012.

[18] Zhouyu Fu, Weiming Hu, and Tieniu Tan. Similarity based vehicle trajectory clustering and anomaly detection. In *Image Processing, 2005. ICIP 2005. IEEE International Conference on*, volume 2, pages II–602. IEEE, 2005.

[19] Maxime Gariel, Ashok N Srivastava, and Eric Feron. Trajectory clustering and an application to airspace monitoring. *Intelligent Transportation Systems, IEEE Transactions on*, 12(4):1511–1524, 2011.

[20] Jiawei Han. *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2005.

[21] Abbas Harati-Mokhtari, Alan Wall, Philip Brooks, and Jin Wang. Automatic identification system (ais): data reliability and human error implications. *Journal of navigation*, 60(03):373–389, 2007.

[22] Alexander Hinneburg and Daniel A Keim. An efficient approach to clustering in large multimedia databases with noise. In *KDD*, volume 98, pages 58–65, 1998.

[23] C. A. R. Hoare. Algorithm 63: Partition. *Commun. ACM*, 4(7):321–, July 1961.

[24] IMO. *Ships' Routeing*. International Maritime Organization, 2013.

[25] Leonard Kaufman and Peter Rousseeuw. *Clustering by means of medoids*. North-Holland, 1987.

[26] Samira Kazemi, Shahrooz Abghari, Niklas Lavesson, Henric Johnson, and Peter Ryman. Open data for anomaly detection in maritime surveillance. *Expert Systems with Applications*, 40(14):5719–5729, 2013.

[27] Eamonn J Keogh and Michael J Pazzani. Scaling up dynamic time warping for datamining applications. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 285–289. ACM, 2000.

[28] Richard O Lane, David A Nevell, Steven D Hayward, and Thomas W Beaney. Maritime anomaly detection and threat assessment. In *Information Fusion (FUSION), 2010 13th Conference on*, pages 1–8. IEEE, 2010.

[29] Rikard Laxhammar. Anomaly detection for sea surveillance. In *Information Fusion, 2008 11th International Conference on*, pages 1–8. IEEE, 2008.

[30] Rikard Laxhammar, Göran Falkman, and Egils Sviestins. Anomaly detection in sea traffic-a comparison of the gaussian mixture model and the kernel density estimator. In *Information Fusion, 2009. FUSION'09. 12th International Conference on*, pages 756–763. IEEE, 2009.

[31] Nicolas Le Guillarme and Xavier Lerouvreur. Unsupervised extraction of knowledge from s-ais data for maritime situational awareness. In *Information Fusion (FUSION), 2013 16th International Conference on*, pages 2025–2032. IEEE, 2013.

[32] Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. Trajectory clustering: A partition-and-group framework. In *Proceedings of the 2007 ACM SIGMOD Inter. Conf. on Management of Data*, SIGMOD '07, pages 593–604, 2007.

[33] Kingsly Leung and Christopher Leckie. Unsupervised anomaly detection in network intrusion detection using clusters. In *Proceedings of the Twenty-eighth Australasian conference on Computer Science-Volume 38*, pages 333–342. Australian Computer Society, Inc., 2005.

[34] Jing Li, Kuei-Ying Huang, Jionghua Jin, and Jianjun Shi. A survey on statistical methods for health care fraud detection. *Health care management science*, 11(3):275–287, 2008.

[35] Bo Liu, Erico N de Souza, Stan Matwin, and Marcin Sydow. Knowledge-based clustering of ship trajectories using density-based approach. In *Big Data (Big Data), 2014 IEEE International Conference on*, pages 603–608. IEEE, 2014.

[36] Brendan Tran Morris and Mohan M. Trivedi. A survey of vision-based trajectory learning and analysis for surveillance. *IEEE Trans. Circuits Syst. Video Techn.*, (8):1114–1127.

[37] Gonzalo Navarro. A guided tour to approximate string matching. *ACM computing surveys (CSUR)*, 33(1):31–88, 2001.

[38] David Nevell. Anomaly detection in white shipping. *Mathematics in Defence*, 2009.

[39] Giuliana Pallotta, Michele Vespe, and Karna Bryan. Traffic knowledge discovery from ais data. In *Information Fusion (FUSION), 2013 16th International Conference on*, pages 1996–2003. IEEE, 2013.

[40] Giuliana Pallotta, Michele Vespe, and Karna Bryan. Vessel pattern knowledge discovery from ais data: A framework for anomaly detection and route prediction. *Entropy*, 15(6):2218–2245, 2013.

[41] Andrey Tietbohl Palma, Vania Bogorny, Bart Kuijpers, and Luis Otavio Alvares. A clustering-based approach for discovering interesting places in trajectories. In *Proceedings of the 2008 ACM Symposium on Applied Computing*, SAC '08, pages 863–868, New York, NY, USA, 2008. ACM.

[42] Claudio Piciarelli, Gian Luca Foresti, and Lauro Snidaro. Trajectory clustering and its applications for video surveillance. In *Advanced Video and Signal Based Surveillance, 2005. AVSS 2005. IEEE Conference on*, pages 40–45. IEEE, 2005.

[43] Laxhammar Rikard. Anomaly detection in trajectory data for surveillance applications. *Studies from the school of science and technology at rebro university 19*, 2011.

[44] Branko Ristic, Barbara La Scala, Mark Morelande, and Neil Gordon. Statistical analysis of motion patterns in ais data: Anomaly detection and motion prediction. In *Information Fusion, 2008 11th International Conference on*, pages 1–7. IEEE, 2008.

[45] Jose Antonio MR Rocha, Gabriel Oliveira, Luis O Alvares, Vania Bogorny, and VC Times. Db-smot: a direction-based spatio-temporal clustering method. In *Intelligent systems (IS), 2010 5th IEEE international conference*, pages 114–119. IEEE, 2010.

[46] Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

[47] Peter J Rousseeuw and Annick M Leroy. *Robust regression and outlier detection*, volume 589. John Wiley & Sons, 2005.

[48] Jean Roy. Rule-based expert system for maritime anomaly detection. In *SPIE Defense, Security, and Sensing*, pages 76662N–76662N. International Society for Optics and Photonics, 2010.

[49] Jean Roy and Michael Davenport. Categorization of maritime anomalies for notification and alerting purpose. In *NATO workshop on data fusion and anomaly detection for maritime situational awareness, La Spezia, Italy*, pages 15–17, 2009.

[50] Lauro Snidaro, Claudio Piciarelli, and Gian Luca Foresti. Fusion of trajectory clusters for situation assessment. In *Information Fusion, 2006 9th International Conference on*, pages 1–7. IEEE, 2006.

[51] Stefano Spaccapietra, Christine Parent, Maria Luisa Damiani, Jose Antonio de Macedo, Fabio Porto, and Christelle Vangenot. A conceptual view on trajectories. *Data Knowl. Eng.*, 65(1):126–146, April 2008.

[52] Abhinav Srivastava, Amlan Kundu, Shamik Sural, and Arun K Majumdar. Credit card fraud detection using hidden markov model. *Dependable and Secure Computing, IEEE Transactions on*, 5(1):37–48, 2008.

[53] Chris Stauffer and W Eric L Grimson. Learning patterns of activity using real-time tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 22(8):747–757, 2000.

[54] Pang-Ning Tan, Michael Steinbach, and Vipin Kumar. *Introduction to Data Mining, (First Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2005.

[55] Henry S Teng, Kaihu Chen, and SC Lu. Adaptive real-time anomaly detection using inductively generated sequential patterns. In *Research in Security and Privacy, 1990. Proceedings., 1990 IEEE Computer Society Symposium on*, pages 278–284. IEEE, 1990.

[56] Chris Veness. Calculate distance, bearing and more between latitude/longitude points.

[57] Michele Vespe, Ingrid Visentini, Karna Bryan, and Paolo Braca. Unsupervised learning of maritime traffic patterns for anomaly detection. In *Data Fusion & Target Tracking Conf. (DF&TT 2012): Algorithms & Applications, 9th IET*, pages 1–5. IET, 2012.

[58] Stijn Viaene, Richard A Derrig, Bart Baesens, and Guido Dedene. A comparison of state-of-the-art classification techniques for expert automobile insurance claim fraud detection. *Journal of Risk and Insurance*, 69(3):373–421, 2002.

[59] Michail Vlachos, George Kollios, and Dimitrios Gunopulos. Discovering similar multidimensional trajectories. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 673–684. IEEE, 2002.

[60] Wei Wang, Jiong Yang, Richard Muntz, et al. Sting: A statistical information grid approach to spatial data mining. In *VLDB*, volume 97, pages 186–195, 1997.

[61] Mohammed Javeed Zaki, Srinivasan Parthasarathy, Mitsunori Ogihara, Wei Li, et al. New algorithms for fast discovery of association rules. In *KDD*, volume 97, pages 283–286, 1997.

[62] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: an efficient data clustering method for very large databases. In *ACM SIGMOD Record*, volume 25, pages 103–114. ACM, 1996.

[63] Zhang Zhang, Kaiqi Huang, and Tieniu Tan. Comparison of similarity measures for trajectory clustering in outdoor surveillance scenes. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 3, pages 1135–1138. IEEE, 2006.

Appendix A

Source Code of the Algorithms

In this appendix, some key source code implemented during the thesis is shown. The whole framework is divided into two parts: normal traffic patterns extraction model and anomaly detection model. The first part shown in A.1 has been mostly implemented in Java with a small portion of R code. While the second part shown in A.2 has been implemented totally in R.

A.1 Source Code of Normal Traffic Patterns Extraction Model

As discussed in Chapter 3, the normal traffic extraction model includes two different clustering components (DBSCANSD and DBSCAN) to handle moving trajectory points and stopping areas respectively. Before presenting the code for clustering, it is necessary to first define the classes of **Trajectory Point** and **Cluster**.

```java
public class TrajectoryPoint {
    private String mmsi;           //mmsi, the id of the vessel
    private long timestamp;          //second of UTC time stamp
    //the following protected variables are to be inherited by GV
    protected double longitude;          //longitude
    protected double latitude;        //latitude
    protected double SOG;             //speed over ground
    protected double COG;             //course over ground
    //the following two are useful during the clustering process
    private boolean isVisited;        //whether it has been visited
    private boolean isCorePoint;      //whether it is a core point
    public TrajectoryPoint() {
        this.isVisited = false;       //initialize the point unvisited
    }
     //generate getter and setter functions ...
}
```

Listing A.1: Class of TrajecotryPoint

The **cluster** defined in Definition 9 is basically a set of trajectory points. So the Class of Cluster can be written as follows:

```java
public class Cluster {
   private ArrayList<TrajectoryPoint> cluster ;
   private double avgCOG;  //the average direction (COG) of the whole
       cluster
    public Cluster() {}
    public double calculateAverageDirection() {
        double sum = 0;
      for(int i=0; i<this.cluster.size();i++) {
         sum = sum+this.cluster.get(i).getCOG();
      }
      double avg = sum/(double)(this.cluster.size());
      return avg;
   }
    //generate getter and setter functions ...
}
```

Listing A.2: Class of Cluster

A.1.1 Code of Clustering Process

The clustering process includes two steps which are DBSCANSD for moving points clustering and DBSCAN for stopping points clustering. The two steps have been implemented together in terms of the code's reusability (DBSCANSD can be regarded as an extension of DBSCAN).

```java
public class DBScanSD {
    //final clustering results, a global variable
    private ArrayList<Cluster> resultClusters = new
        ArrayList<Cluster>();
    /**
     * Apply DBSCANSD on the data set. If the data is stoppoing points,
        set isStopPoint as true and it will execute the original
        DBSCAN directly. If the data is moving points, set isStopPoint
        as false and it will execute DBSCANSD considering speed and
        direction.
     */
    public ArrayList<Cluster> applyDBScanSD(ArrayList<TrajectoryPoint>
        pointsList, double eps, int minPoints, double maxSpd, double
        maxDir, boolean isStopPoint) {
      for(int index=0;index<pointsList.size();index++) {
        //we should mark the point as visited, it has no problem
            because here we use a for loop, it can stop
          ArrayList<TrajectoryPoint> tmpLst = new
              ArrayList<TrajectoryPoint>();
          TrajectoryPoint p = pointsList.get(index);
          if(p.isVisited()&&index!=(pointsList.size()-1)&&index%4096!=0)
              continue;
          tmpLst = isCorePoint(pointsList, p, eps, minPoints, maxSpd,
              maxDir, isStopPoint);
          if(tmpLst!=null||index==(pointsList.size()-1)||index%4096==0){
           Cluster c = new Cluster();
           c.setCluster(tmpLst);
           if(tmpLst!=null)  resultClusters.add(c);
           int length=resultClusters.size();
           boolean flag = true;
           if((index%4096==0)||(index==(pointsList.size()-1)))  {
               while(flag) {
                   flag = false;
                    for(int i=0;i<length;i++){
                             for(int j=0;j<length;j++){
                                 if(i!=j){
                                     if(i == length) {
                                         flag = true;
                                         continue;
                                     }
                                     if(mergeClusters(resultClusters.get(i),
                                        resultClusters.get(j))) {
                                         resultClusters.remove(j);
                                         j--;
                                         length--;
                                     }
                                 }
                             }
                         }
               }
            }
          }
      }
      return resultClusters;
  }
  /**
    * Merge two clusters into one cluster
    */
  public boolean mergeClusters(Cluster clusterA, Cluster clusterB) {
     boolean merge = false;
     if(clusterA.getCluster() == null || clusterB.getCluster() ==
        null) {
        return merge;
     }
     for(int index = 0; index < clusterB.getCluster().size(); index++)
        {
        TrajectoryPoint p = clusterB.getCluster().get(index);
        if(p.isCorePoint() && clusterA.getCluster().contains(p)) {
           merge = true;
           break;
        }
     }
     if(merge) {
        for(int index=0; index<clusterB.getCluster().size();index++) {
           if(!clusterA.getCluster().contains(clusterB.getCluster()
              .get(index))) {
              clusterA.getCluster().add(clusterB.getCluster().get(index));
           }
        }
     }
     return merge;
  }
  /**
    * Decide if the point p is core point, if yes, return the list
       with p and its neighbors, if no, return null.
    */
    public ArrayList<TrajectoryPoint>
        isCorePoint(ArrayList<TrajectoryPoint> lst, TrajectoryPoint p,
        double eps, int minPoints, double maxSpd, double maxDir, boolean
        isStopPoint) {
        int count = 0;
        ArrayList<TrajectoryPoint> tmpList = new
            ArrayList<TrajectoryPoint>();
        for(Iterator<TrajectoryPoint> it = lst.iterator(); it.hasNext();)
            {
            TrajectoryPoint q = it.next();
            if(isDensityReachable(p, q, eps, minPoints, maxSpd, maxDir,
                isStopPoint)) {
                count++;
                if(!tmpList.contains(q)) {
                    tmpList.add(q);
                }
            }
        }
        if(count>=minPoints) {
            p.setCorePoint(true);
            p.setVisited(true);
            return tmpList;
        }
        return null;
    }
     /**
      * Decide if the two points are density reachable.
      */
     public boolean isDensityReachable(TrajectoryPoint p1,
         TrajectoryPoint p2, double eps, int minPts, double maxSpd,
         double maxDir, boolean isStopPoint) {
         boolean result = false;
         if(gpsDistance(p1.getLatitude(), p1.getLongitude(),
             p2.getLatitude(), p2.getLongitude()) <=eps) {
             //if they are stopping points, we can directly use original
                 DBSCAN algorithm without considering speed or direction
             if(isStopPoint) return true;
             if(Math.abs(p1.getCOG()-p2.getCOG())<maxDir) {
                 if(Math.abs(p1.getSOG()-p2.getSOG())<maxSpd) {
                    result = true;
                }
            }
        }
        return result;
    }
    /**
     * calcualte the gps distance between two trajectory points
         considering the curve of the earth.
     */
    public static double gpsDistance(double lat1, double lng1, double
        lat2, double lng2) {
        double earthRadius = 3958.75;
        double dLat = Math.toRadians(lat2-lat1);
        double dLng = Math.toRadians(lng2-lng1);
        double a = Math.sin(dLat/2) * Math.sin(dLat/2) +
        Math.cos(Math.toRadians(lat1)) * Math.cos(Math.toRadians(lat2))
            *
        Math.sin(dLng/2) * Math.sin(dLng/2);
        double c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
        double dist = earthRadius * c;
        int meterConversion = 1609;
        return (double) (dist * meterConversion);
    }
}
```

Listing A.3: The code of clustering process

A.1.2 GVs Calculation

In this section, the code of the algorithm to calculate a cluster's Gravity Vectors is given. First, as defined in Chapter 3, a Gravity Vector is a vector formed by 5 features: average COG, average SOG, average Latitude, average Longitude and Median Distance, the class of the **Gravity Vector** needs to be first presented. Here the GV is inherited from the Class of Trajectory Point (Listing A.1).

```
public class GravityVector extends TrajectoryPoint{
    private double medianDistance;      //the median distance feature
```

```java
    public GravityVector(double longitude, double latitude, double COG,
        double SOG, double medianDistance) {
        this.latitude = latitude;
        this.longitude = longitude;
        this.COG = COG;
        this.SOG = SOG;
        this.medianDistance = medianDistance;
    }
     //generate getter and setter functions ...
}
```

Listing A.4: Class of Gravity Vector

As demonstrated in Chapter 3, the process of calculating GVs of a cluster needs to map the trajectory points to an axis of the average direction. So here a Class called Mapping Point is also given and it also inherits from the Class of Trajectory Point:

```java
public class MappingPoint extends TrajectoryPoint{
    private double mappingtude; //mapping point's coordinate on the
        exact axis
     /**
      * Mapping Point is the point that is mapped onto the axis (average
          direction of the whole cluster)
      */
    public MappingPoint(double longitude, double latitude, double
        mappingtude, double COG, double SOG) {
        this.longitude = longitude;
        this.latitude = latitude;
        this.mappingtude = mappingtude;
        this.COG = COG;
        this.SOG = SOG;
    }
     /**
      * map the trajectory point to the particular axis and return the
          mapping point
      */
    public MappingPoint convertPointToMappingPoint(TrajectoryPoint p,
        double avgCOG) {
        double mappingtude = 0;
        double angle = (avgCOG/(double)180) * Math.PI;
        if((avgCOG>=0&&avgCOG<90)) {
            mappingtude = (p.getLongitude() + (1.0/Math.tan(angle)) *
                p.getLatitude()) * Math.sin(angle);
        } else if((avgCOG >= 270&&avgCOG<360)) {
            mappingtude = (p.getLatitude() - (Math.tan(Math.PI*2-angle))*
                p.getLongitude()) * Math.cos(Math.PI*2 - angle);
        } else if(avgCOG >= 90&&avgCOG<180) {
            mappingtude = ((Math.tan(Math.PI-angle))*p.getLongitude() -
                p.getLatitude()) * Math.cos(Math.PI - angle);
        } else if(avgCOG >= 180&&avgCOG<270) {
            mappingtude = -(((double)1/Math.tan(angle - Math.PI)) *
                p.getLatitude() + p.getLongitude()) * Math.sin(angle -
                Math.PI);
        }
        MappingPoint mp = new MappingPoint(p.getLongitude(),
            p.getLatitude(), mappingtude, p.getCOG(), p.getSOG());
        return mp;
    }
    // generate setter and getter functions ...
}
```

Listing A.5: Class of Mapping Point

After having the Gravity Vector and Mapping Point classes, we can start to write the code for extracting the GVs from a given cluster.

```
public class GravityVectorExtraction {
    /**
     * extract the GVs from the moving cluster
     * @param cluster: input a cluster for extracting the GVs
     * @return an arraylist of GVs
     */
    public ArrayList<GravityVector> extractGravityVector(Cluster
        cluster) {
        //Step(1): Calculate the average COG of the whole cluster
        double avgCOG = cluster.calculateAverageDirection();
        //Step(2) Map all the points to the axis and partition the points
            based on 0.01 grid width
        ArrayList<MappingPoint> mpLst = new ArrayList<MappingPoint>();
```

```java
        for(int i=0; i<cluster.getCluster().size(); i++) {
            MappingPoint mp =
                MappingPoint.convertPointToMappingPoint(cluster.getCluster()
                .get(i), avgCOG);
            mpLst.add(mp);
        }
        insertionSort(mpLst);
        ArrayList<GravityVector> ppL = new ArrayList<GravityVector>();
        int count = 0;
        int k = 0;
        double sum_x = 0;
        double sum_y = 0;
        double sum_SOG = 0;
        double sum_COG = 0;
        ArrayList<MappingPoint> traPointsTMP = new
            ArrayList<MappingPoint>();
        double medianDistance = 0;
        //partition the points and calculate each GV for each cell
        while(count<=mpLst.size()) {
            if(count < mpLst.size() && (mpLst.get(count) .getMappingtude()
                - mpLst.get(k). getMappingtude() < 0.01)) {//0.01 as the
                pre-defined grid width
                sum_x = sum_x+mpLst.get(count).getLongitude();
                sum_y = sum_y+mpLst.get(count).getLatitude();
                sum_SOG = sum_SOG+mpLst.get(count).getSOG();
                sum_COG = sum_COG+mpLst.get(count).getCOG();
                traPointsTMP.add(mpLst.get(count));
                count++;
            } else {
                double x = 0;
                double y = 0;
                double sog = 0;
                double cog = 0;
                x = sum_x/(double)(count-k);
                y = sum_y/(double)(count-k);
                sog = sum_SOG/(double)(count-k);
                cog = sum_COG/(double)(count-k);
                //insert median distance calculation
                double[] distances = new double[traPointsTMP.size()];
            for(int i=0; i<traPointsTMP.size();i++) {
                double lon=traPointsTMP.get(i).getLongitude();
                double lat=traPointsTMP.get(i).getLatitude();
                double dist = gpsDistance(lat, lon, y, x);
                distances[i]=dist;
            }
            //medianDistance
            medianDistance = quartile(distances, 50);
                 //for each cell of the grid, calculate its GV
            GravityVector gv = new
                GravityVector(x,y,cog,sog,medianDistance);
            ppL.add(gv);
            sum_x = 0;
            sum_y = 0;
            sum_COG = 0;
            sum_SOG = 0;
            k = count;
            traPointsTMP.clear();
            if(count==mpLst.size()) break;
        }
    }
    return ppL;
}
 /**
  * insertion sorting
  */
public void insertionSort(ArrayList<MappingPoint> mpl) {
    for(int i=1; i<mpl.size(); i++) {
        int k = i;
        MappingPoint mp = mpl.get(i);
        boolean insertAlready = false;
        while(mpl.get(i).getMappingtude()<mpl.get(k-1).getMappingtude())
            {
            if(k==1) {
                mpl.remove(i);
                mpl.add(0, mp);
                insertAlready = true;
                break;
            }
```

```
                k--;
            }
            if(!insertAlready) {
                mpl.remove(i);
                mpl.add(k,mp);
            }
        }
    }
     /**
      * Retrive the quartile value from an array, used for generating
         relative distance.
      * @param values The array of data
      * @param lowerPercent The percent cut off. For the lower quartile
         use 25, for the upper-quartile use 75
      * @return the quartile value
      */
    public double quartile(double[] values, double lowerPercent) {
          if (values == null || values.length == 0) {
              throw new IllegalArgumentException("The data array either
                  is null or does not contain any data.");
          }
          // order the values
          double[] v = new double[values.length];
          Arrays.sort(v);
          int n=0;
          if(v.length==1) {
           n = 0;
          }
          else n = (int) Math.round(v.length * lowerPercent / 100);
          return v[n];
     }
}
```

Listing A.6: Extract GVs from a cluster

A.1.3 SSPs Calculation

SSP (Sampled Stopping Point) is a type of points to represent the geo-spatial shape of a cluster and it does not need to consider the factors of speed or direction (See Section

3.3). Here the algorithm of extracting SSPs from stopping clusters (Algorithm 2) has been implemented in R.

```r
## The main function of SSP extraction process.
## The parameter of stoppingClusters is the stopping clustering results generated by DBSCAN;
## The parameter of radius here is the eps defined during the clustering process
stopPointsSampling<-function(stoppingClusters,radius) {
  clusterIDs<-unique(stoppingClusters[,"clusterindex"]);
  end=length(clusterIDs);
  result<- data.frame(Longitude=numeric(),Latitude=numeric());
  for(i in 1:end) {
    id=clusterIDs[i];
    clusterData<-stoppingClusters[stoppingClusters$clusterindex==id,];
    sample_num<-sampleNumberEstimate(clusterData,radius);
    sampledPoints<-samplePointsFromData(clusterData,sample_num,radius);
    result<-rbind(result,sampledPoints);
  }
  return(result);
}
## Sample points from stopping clusters such that every sampled point is far enough from other sampled points
samplePointsFromData<-function(data,sample_num,radius) {
  result<- sampledPoint<-data[sample(nrow(data),1),2:3];
  count=1;
  while(count<sample_num) {
    sampledPoint<-data[sample(nrow(data),1),2:3];
    end=nrow(result);
    nearFlag=FALSE;
    for(i in 1:end) {
      if(gpsDistance(result[i,],sampledPoint)<radius) {
        nearFlag=TRUE;
        break;
      }
    }
    if(nearFlag) {
      next;
    } else {
        count=count+1;
        result<-rbind(result,sampledPoint);
      }
    }
    return(result);
}
## Estimate the number of points to be sampled
sampleNumberEstimate<-function(clusterPoints, radius) {
    min_x<-min(clusterPoints[,"Latitude"]);    #downmost point
    max_x<-max(clusterPoints[,"Latitude"]);    #upmost point
    min_y<-min(clusterPoints[,"Longitude"]);   #rightmost point
    max_y<-max(clusterPoints[,"Longitude"]);   #leftmost point
    areaEstimate<-(max_x-min_x)*(max_y-min_y);
    numberEstimate<-floor(areaEstimate/(pi*radius^2))+1;
    if(areaEstimate==0) {
      numberEstimate=max((max_x-min_x),(max_y-min_y))/radius+1;
    }
      return(numberEstimate);
}
```

Listing A.7: Extract SSPs from stopping clusters

A.2 Source Code of Anomaly Detection Model

This section is focused on the anomaly detection labeling process. The algorithm uses as input the clusters described in previous sections, and uses them to generate a label that informs what type of anomaly was found. The algorithm associates labels to each AIS data point according to the following:

- **0** stands for distance abnormal
- **1** is normal
- **-1** is direction or speed abnormal
- **9** is data incomplete

The R function code, called labelAnomalyPointsWithStopMoveclusters, contains the function that is responsible for the labeling detection. Observe that this function uses other helper functions that are later described. The function inputs are the following:

- The inputData parameter is a data frame of the following format: ("MMSI", "SOG", "Longitude", "Latitude", "COG");
- The normalPointsMove parameter is a data frame of the following format: ("clusterindex", "Longitude", "Latitude", "SOG", "COG", "QuartileDistance")
- The normalPointsStop is a data frame with the following format: ("Longitude","Latitude")

The algorithm first executes a check to guarantee that data is in correct format, by verifying if all required fields in inputData are complete. If they are not, the algorithm adds a label 9, indicating that data is incomplete. This step guarantees that all AIS coordinates that have missing values are going to be labeled as incomplete.

Next, the algorithm executes a function called generatefeaturesRelativeDistance, which generates a set of extra distance descriptors for the inputData parameter. This function is presented in next section. These distances will be used to rank which AIS points are more abnormal than others. The function getIndexOfFarPointsRELATIVE calculates which points are abnormal in relation to distance from the clusters. This function uses two extra parameters as threshold to indicate the anomaly. These values were empirically found during clustering steps.

If the points are still normal in relation to distance from the clusters, it does not mean that they are normal in relation to speed and direction. Then a second set of tests is required to check abnormality. This is started in function getIndexOfNearPointsRELATIVE, which simple gets a list of points that are close to the clusters. After getting the list of close points, the algorithm generates another set of distances that are used to estimate direction and speed anomalies. This is done in function generatefeatures4 that calculates the cosine distance, and speed similarities in relation to the cluster. Next section presents the other sub-routines.

```
## label the input data with normal moving clusters(GVs) and normal
    stopping clusters(SSPs)
labelAnomalyPointsWithStopMoveclusters<-function(inputData,
    normalPointsMove, normalPointsStop) {
  labeledData<-inputData;
  labeledData[,"label"]<-labeledData[,1];
  labeledData[,"label"]<-1;
  #calulate features for the 1st step
  #extract those incomplete data
```

```
end=nrow(labeledData);
for(i in 1:end) {
  if(!complete.cases(labeledData[i,])) {
    labeledData[i,"label"]<-9;
  }
}
print("Incomplete records labeled finished")
featuresOne <- generatefeaturesRelativeDistance(inputData,
    normalpointsMove = normalPointsMove, normalpointsStop =
    normalPointsStop);
#below thresholds are for Juan De Fuca strait in paper of
    "Knowledge-based clustering of ship trajectories using
    density-based approach"
rel_dis_threshold = 5.765;
abs_dis_threshold = 94.096
#get indices for far points
index_far_relative <- getIndexOfFarPointsRELATIVE(featuresOne,
    rel_dis_threshold,abs_dis_threshold = abs_dis_threshold);
if(nrow(index_far_relative)>0) {
  for(m in 1:nrow(index_far_relative)) {
    if(labeledData[index_far_relative[m,1],"label"]!=9) {
      labeledData[index_far_relative[m,1],"label"] <-0;
    }
  }
}
print("First step finished.")
#2nd step
index_near_relative <- getIndexOfNearPointsRELATIVE(featuresOne,
    rel_dis_threshold, abs_dis_threshold);
if(nrow(index_near_relative)>0) {
  featureSecond<-generatefeatures4(inputData [index_near_relative
      [,1],], normalPoints, rel_dis_threshold);
  cosineThreshold=0.588
  end2 = nrow(featureSecond);
  for(j in 1:end2) {
    if(featureSecond[j,]$cosSimilarity<cosineThreshold) {
      labeledData[index_near_relative[featureSecond[j,]$INDEX,1],
          "label"] <- (-1);
    }
    }
  }
  print("Data label process finished!")
  return(labeledData);
}
## generate features considering relative distance
## realpoints: the point to be labeled
## normalpointsMove: moving clusters (GVs)
## normalpointsStop: stopping clusters (SSPs)
generatefeaturesRelativeDistance <- function(realpoints,
    normalpointsMove, normalpointsStop) {
    end = nrow(realpoints);
    features = data.frame (distance=numeric(),
        relative_distance=numeric(), SOGRatio=numeric(), COG=numeric(),
        isStopPoint=integer());
    for(i in 1:end) {
        features =
            rbind(features,calculateDistancesRelative(realpoints[i,],
            normalpointsMove, normalpointsStop));
    }
    colnames(features) <- c("Absolute_Distance", "Relative_Distance",
        "SOGratio", "COG", "isStopPoint");
    return(features);
}
## calculate the relative distances between the point and the clusters
## realpoints: the point to be labeled
## normalpointsMove: moving clusters (GVs)
## normalpointsStop: stopping clusters (SSPs)
calculateDistancesRelative<- function(realpoint, normalpoints,
    normalpointsStop) {
    index = -1;
    min_d = 1000000000000000;
    relative_distance = 1000000000000000;
    end = nrow(normalpoints);
    end2=nrow(normalpointsStop);
    isStopPoint = 0;
    #use 0.5 as stopping point condition, we can change it to another
        speed threshold
    if(realpoint[1,"SOG"]<=0.5 | !complete.cases(realpoint[1,])) {
        isStopPoint = 1;
        for( i in 1:end2) {
            #actually here I should use absolute distance instead of
                relative_distance
            #I am too lazy to do this and this will not influence the
                result , just a name
            distance_tmp = gpsdist(realpoint[1,"Latitude"],
                realpoint[1,"Longitude"],
                normalpointsStop[i,"Latitude"],
                normalpointsStop[i,"Longitude"]);
            if(distance_tmp < min_d) {
                relative_distance<-999999999;#stands for NA
                min_d <-distance_tmp;
                index <- i;
            }
        }
    } else {
        for(i in 1:end) {
            distance = gpsdist(realpoint[1, "Latitude"], realpoint[1,
                "Longitude"], normalpoints[i, "Latitude"],
                normalpoints[i, "Longitude"]);
            quartileDistance=normalpoints[i, "QuartileDistance"];
            if(quartileDistance==0) {
                quartileDistance=1; #to avoid the x/0 error
            }
            relative_dis_tmp = distance/quartileDistance;
            if(relative_dis_tmp < relative_distance) {
                relative_distance<-relative_dis_tmp
                min_d <-distance;
                index <- i;
            }
        }
    }
    speedratio <- (abs(realpoint[1,"SOG"] -
        normalpoints[index,"SOG"]))/ normalpoints[index,"SOG"];
    direction_dif<- abs(realpoint[1,"COG"] - normalpoints[index,"COG"]);

    return (c(min_d, relative_distance,speedratio, direction_dif,
        isStopPoint ));
```

```
}
## used for calculating the geographical distance of two positions
## lat1: latitude of point 1
## lon1: longitude of point 1
## lat2: latitude of point 2
## lon2: longitude of point 2
gpsdist<-function(lat1,lon1,lat2,lon2) {
    R = 6367000;
    dlat = (lat2-lat1)*pi/180;
    dlon = (lon2-lon1)*pi/180;
    radlat1 = lat1*pi/180;
    radlat2 = lat2*pi/180;
    a = sin(dlat/2) * sin(dlat/2) + (sin(dlon/2) * sin(dlon/2) *
        cos(radlat1) * cos(radlat2));
    c = 2 * atan2(sqrt(a), sqrt(1-a));
    d = R * c;
    d;
}
## get the indices of those points which are too far away
## test_set: the data set to be labeled
## dis is the threshold for the distance
getIndexOfFarPoints <- function(test_set, dis) {
    end = nrow(test_set)
    index = data.frame(index=numeric());
    for(i in 1:end) {
        if(test_set[i,]$distance>dis) {
            index <- rbind(index,i);
        }
    }
    return(index);
}
## get the indices of the points that are near the clusters
## they can be used for training or predicting the speed&direction
   abnormal thing
getIndexOfNearPoints <- function(test_set, dis) {
    end = nrow(test_set)
    index = data.frame(index=numeric());
    for(i in 1:end) {
        if(test_set[i,]$distance<=dis) {
            index <- rbind(index,i);
        }
    }
    return(index);
}
## get the indices of the points that are near the clusters relatively
## they can be used for training or predicting the speed&direction
    abnormal thing
#this is based on relative distance
getIndexOfNearPointsRELATIVE <- function(test_set,
    relative_dis_threshold, abs_dis_threshold) {
    end = nrow(test_set)
    index = data.frame(index=numeric());
    for(i in 1:end) {
        if(test_set[i,"isStopPoint"]==1) {
        } else {
            if(test_set[i,]$Relative_Distance <=
                relative_dis_threshold) {
                index <- rbind(index,i);
            }
        }
    }
    return(index);
}
## points far away from normal points
## this is based on relative distance
## relative_dis_threshold: the threshold for moving points
## abs_distance: the threshold for stopping points
getIndexOfFarPointsRELATIVE <- function(test_set,
    relative_dis_threshold, abs_dis_threshold) {
    end = nrow(test_set)
    index = data.frame(index=numeric());
    for(i in 1:end) {
        if(test_set[i,"isStopPoint"]==1) {
            if(test_set[i,]$Absolute_Distance > abs_dis_threshold) {
                index<-rbind(index,i);
            }
        } else {
            if(test_set[i,]$Relative_Distance > relative_dis_threshold)
               {
                index <- rbind(index,i);
            }
        }
    }
    return(index);
}
## generate the features for the second labelling step: cosine division
    distance
generatefeatures4 <- function(realpoints, normalpoints,
    threshold_relative) {
    end = nrow(realpoints);
    features = data.frame(relativeDistance=numeric(),
        distanceSD=numeric(),
        distance=numeric(),cosSimilarity=numeric(),
        distanceSimilarity=numeric(), SOG=numeric(), COG=numeric(),
        INDEX=numeric());
    print("Start to label cosine division distances")
    for(i in 1:end) {
        index = i;
        if(complete.cases(realpoints[i,])) {
            tmp<-calculateDistances4(realpoints[i,], index,
                normalpoints, threshold_relative);
            features <- rbind(features,tmp);
        }
    }
    colnames(features) <- c("relativeDistance", "distanceSD",
        "distance", "cosSimilarity", "distanceSimilarity", "SOGdif",
        "COGdif", "INDEX");
    #distanceSD minimum distance considering speed and direction
    return(features);
}
## subroutine in function of "generatefeatures4"
calculateDistances4<- function(realpoint, index, normalpoints,
    threshold) {
    min_relativeDistance = 1000000000; # used for storing the relative
        distance considering speed&direction
    min_d = 100000000000;       #used for storing the distance to the
       nearest points considering speed&direction
    cosSimilarity = -1000000;   # cosineSimilarity
    direction_dif<-100000000;  # direction difference
    speed <- 100000000;        # speed difference
    min_distance <- 10000000000;#used for storing the nearest points
       without considering speed&direction
    distanceSimilarity <- 0;
    end = nrow(normalpoints);
    for(i in 1:end) {
        distance = gpsdist(realpoint[1, "Latitude"], realpoint[1,
           "Longitude"], normalpoints[i, "Latitude"], normalpoints[i,
           "Longitude"]);
        relative_distance = distance/(normalpoints[i,
           "QuartileDistance"] + 0.00000001);
        # below is to record the nearest point in the space without
           considering speed & direction
        if(distance<min_distance) {
            min_distance = distance;
        }
        if(relative_distance < threshold) {
            min_relativeDistance=relative_distance;
            # consider those points with small speed in the stop area
               as normal, that is, no need to consider direction
               parameter(set its difference to 0)
            if(realpoint[1,"SOG"]<=0.3& normalpoints[i,"SOG"]<=0.3) {
                if(distance<min_d) {
                    min_d <- distance;
                }
                speed <- abs(realpoint[1,"SOG"]-normalpoints[i,"SOG"]);
                direction_dif <- 0;

                cosSimilarity = 1;
            }
            else {
                if(realpoint[1,"SOG"]>normalpoints[i,"SOG"]) {
                    speedRatio <-
                       normalpoints[i,"SOG"]/realpoint[1,"SOG"];
```

```
                        cosSimilarityTmp <- cos(((abs(realpoint[1,"COG"] -
                            normalpoints[i,"COG"]))/180)*pi) * speedRatio;
                    } else {
                        speedRatio <-
                            realpoint[1,"SOG"]/normalpoints[i,"SOG"];
                        cosSimilarityTmp <- cos(((abs(realpoint[1,"COG"] -
                            normalpoints[i,"COG"]))/180)*pi) * speedRatio;
                    }
                    if(cosSimilarityTmp>cosSimilarity) {
                        min_d <-distance;
                        direction_dif<-
                            abs(realpoint[1,"COG"]-normalpoints[i,"COG"]);
                        speed<-abs(normalpoints[i,"SOG"]-realpoint[1,"SOG"]);
                        cosSimilarity<- cosSimilarityTmp;
                    }
                }
            }
            distanceSimilarity <- (min_distance+1)/(min_d+1)
        }
        return (c(min_relativeDistance,min_d, min_distance, cosSimilarity,
            distanceSimilarity, speed, direction_dif,index));
}
```

Listing A.8: Anomaly Detection