TÉCNICO LISBOA

Extracting Maritime Routes for Vessel Route Prediction

Daniela Sofia da Conceição Duarte

Thesis to obtain the Master of Science Degree in

Information Systems and Computer Engineering

Supervisor: Prof. Manuel Fernando Cabido Peres Lopes

Examination Committee

Chairperson: Prof. Name of the Chairperson
Supervisor: Prof. Manuel Fernando Cabido Peres Lopes
Member of the Committee: Prof. Name of First Committee Member

September 2020

Acknowledgments

I would like to acknowledge my dissertation supervisor Prof. Manuel Lopes for his insights and sharing of knowledge that has made this thesis possible.

To INOV, I would like to thank them for the opportunity to work in the MARISA project, which provided the tools for the start of this work.

A special thanks to Stefan Nae, for kindly reviewing the writing.

To my family, for the support in all kinds of forms, and my friends and colleagues, that in a way or another helped me grow as a person and always stand tall, thank you.

To each and every one of you – Thank you.

Abstract

Vessels are often intermittently observed when at sea, either because the monitoring systems are shut down, or the coverage link can not reach the sailed region. These gaps represent a severe threat in terms of safety at sea. The present work proposes to address this problem by predicting ship movement when there is no available information. The movement data of past behaviors is clustered into groups of similar movement patterns, from which is extracted a representative trajectory for each cluster. Ultimately they will represent each main route and be used to identify the route membership of incoming trajectories and predict future movements by querying the route's function over time at a given instant. The evaluation is made with a real-world AIS dataset to demonstrate the viability of this approach.

Keywords

Automatic Identification System; Longest Common Subsequence; Trajectory Clustering; Movement Prediction.

Resumo

Os navios são frequentemente observados de forma intermitente quando navegam no mar, seja porque os sistemas de monitorização a bordo são desligados, ou a cobertura do sinal não alcança a região navegada pelo mesmo. Estas falhas representam uma grave ameaça em termos de segurança no mar. O presente trabalho propõe-se a abordar este problema ao efectuar uma previsão do movimento de barcos, quando a informação disponível do mesmo passa a ser inexistente. Os dados de comportamentos passados são agrupados em conjuntos de padrões de movimento semelhante, dos quais é extraída uma trajetória representativa para cada grupo. Estas, representam cada rota marítima principal e são usadas para identificar a rota a que uma trajetória futura pertence e estimar o seu movimento ao usar a informação da respectiva rota principal, num dado instante. A avaliação é feita num conjunto de dados reais de AIS, de forma a demonstrar a viabilidade desta abordagem.

Palavras Chave

Automatic Identification System; Subsequência Comum Mais Longa; Agrupamento de Trajetórias; Previsão de Rotas.

Contents

Acknowledgments iii

Abstract iv

Resumo vi

List of Figures xi

List of Tables xii

List of Algorithms xii

List of Acronyms xiii

1 Introduction 1

1.1 Project Goal . . . 3

1.2 Document Structure . . . 3

2 Related Work 4

2.1 Trajectory Data Mining . . . 5

2.2 Trajectory Learning . . . 5

2.2.1 Distance-based Approaches . . . 6

2.2.1.A Similarity Measures . . . 6

2.2.1.B Clustering Methods . . . 8

2.2.2 Model-based Approaches . . . 12

2.3 Route Prediction . . . 13

2.3.1 Physical Models . . . 13

2.3.2 Learning Model-Based Approaches . . . 13

2.3.3 Hybrid Models . . . 14

3 Data Analysis and Clustering 15

3.1 Data Sample . . . 16

3.2 Data Treatment . . . 17

3.2.1 Splines . . . 19

3.3 Clustering Method . . . 20
3.3.1 Similarity Measure . . . 21
3.3.1.A Longest Common Subsequence . . . 22
A – Position Data . . . 22
B – Angle Data . . . 23
C – LCSS Joint Distance Measure . . . 23
3.3.1.B Dynamic Time Warping . . . 24
A – DTW Joint Distance Measure . . . 24
3.3.1.C Results . . . 24
3.3.2 Agglomerative Clustering . . . 26
3.3.2.A Model Selection . . . 26
3.3.3 DBSCAN . . . 27
3.3.3.A Model Selection . . . 28
3.3.4 OPTICS . . . 28
3.3.5 Results . . . 29
3.3.5.A Set 1 . . . 29
3.3.5.B Set 2 . . . 33
3.3.5.C Set 3 . . . 37
3.3.6 Re-Classification . . . 41

4 Prediction **45**
4.1 Algorithm Description . . . 46
4.1.1 Softmax . . . 47
4.2 Results . . . 48
4.2.1 Set 1 . . . 49
4.2.2 Set 2 . . . 52
4.2.3 Set 3 . . . 56

5 Conclusions and Future Work **61**
5.1 Conclusions . . . 62
5.2 Future Work . . . 63

A Tables **69**

B Code **71**

List of Figures

2.1 Time-series clustering categorization. . . . 6
2.2 Challenges of measuring similarity between trajectories. . . . 7
2.3 Matching difference between ED, DTW, and LCSS [1]. . . . 8

3.1 Data analysis and clustering approach. . . . 16
3.2 Dataset visualization and statistics. . . . 18
3.3 Interpolation. . . . 19
3.4 Example trajectory resampling. . . . 21
3.5 LCSS representation between trajectories x and y. . . . 22
3.6 Sample ordered by the LCSS joint distance values. . . . 25
3.7 Sample ordered by DTW joint distance values. . . . 25
3.8 Density representation of DBSCAN with k=3 and arbitrary ε. . . . 27
3.9 K-dist plot [2] 28
3.10 Core and reachability distance, for $k = 3$. . . . 29
3.11 Top 15 levels of the hierarchical clustering dendrogram with average link, for set 1. . . . 30
3.12 14 out of 65 HAC clusters with distance_threshold = 70 and average linkage, for set 1. . . . 30
3.13 K-dist plot for set 1. . . . 31
3.14 8 DBSCAN clusters with ε = 13 and k = 3, for set 1. . . . 31
3.15 5 DBSCAN clusters with ε = 20 and k = 6, for set 1. . . . 32
3.16 4 DBSCAN clusters with ε = 22 and k = 9, for set 1. . . . 32
3.17 15 OPTICS clusters with k = 3, for set 1. . . . 32
3.18 6 OPTICS clusters with k = 6, for set 1. . . . 33
3.19 5 OPTICS clusters with k = 9, for set 1. . . . 33
3.20 Top 15 levels of the hierarchical clustering dendrogram with average link, for set 2. . . . 34
3.21 K-dist plot for set 2. . . . 35
3.22 14 out of 39 HAC clusters with distance_threshold = 70 and average linkage, for set 2. . . . 35
3.23 5 DBSCAN clusters with ε=22 and k=3, for set 2. . . . 36

3.24 2 DBSCAN clusters with ε=25 and k=6, for set 2. 36
3.25 2 DBSCAN clusters with ε=30 and k=9, for set 2. 36
3.26 22 OPTICS cluster with k = 3, for set 2. 36
3.27 8 OPTICS cluster with k = 6, for set 2. 37
3.28 5 OPTICS cluster with k = 9, for set 2. 37
3.29 K-dist plot for set 3. 38
3.30 Top 15 levels of the hierarchical clustering dendrogram with average link, for set 3. 39
3.31 22 HAC clusters with distance_threshold = 70 and average linkage, for set 3. 39
3.32 5 DBSCAN clusters with ε = 30 and k = 3, for set 3. 40
3.33 2 DBSCAN clusters with ε = 34 and k = 6, for set 3. 40
3.34 2 DBSCAN clusters with ε = 42 and k = 9, for set 3. 40
3.35 15 OPTICS clusters with k = 3, for set 3. 40
3.36 3 OPTICS clusters with k = 6, for set 3. 41
3.37 2 OPTICS clusters with k = 9, for set 3. 41
3.38 Re-classification changes subset of HAC clusters, for set 1. 43
3.39 Re-classification changes subset of HAC clusters, for set 2. 43
3.40 Re-classification changes of OPTICS clusters with $k = 3$, for set 2. 43
3.41 Re-classification changes subset of HAC clusters, for set 3. 44
3.42 Re-classification changes of OPTICS clusters with $k = 3$, for set 3. 44

4.1 Route prediction approach. 46
4.2 Iterative route prediction. 47
4.3 Route prediction with softmax for $t + 1$. 48
4.4 Comparative analysis of different beta values for each observation size – HAC clusters, on set 1. 50
4.5 Example of movement prediction with HAC clusters, for set 1. 52
4.6 Comparative analysis of different beta values for each observation size – HAC clusters, on set 2. 53
4.7 Example of movement prediction with HAC clusters, for set 2. 54
4.8 Comparative analysis of different beta values for each observation size – OPTICS clusters, on set 2. 55
4.9 Example of movement prediction with OPTICS clusters, for set 2. 56
4.10 Comparative analysis of different beta values for each observation size – HAC clusters, on set 3. 57
4.11 Example of movement prediction with HAC clusters, for set 3. 58

4.12 Comparative analysis of different beta values for each observation size – OPTICS clusters, on set 3. 59
4.13 Example of movement prediction with OPTICS clusters, for set 3. 60

List of Tables

2.1 Overview of similarity measures characteristics, extracted from [3]. 9
2.2 Overview of clustering algorithms, extracted from [4]. 10

3.1 Data record sample. 17
3.2 CFCC values for set 1. 30
3.3 CFCC values for set 2. 35
3.4 CFCC values for set 3. 38

4.1 Outliers for HAC clusters, for set 1. 50
4.2 Outliers for HAC clusters, for set 2. 53
4.3 Outliers for OPTICS clusters, for set 2. 54
4.4 Outliers for HAC clusters, for set 3. 57
4.5 Outliers for OPTICS clusters, for set 3. 59

A.1 Dynamic data description. 70
A.2 Static data description. 70

List of Algorithms

B.1 LCSS Algorithm with Cosine Distance 72
B.2 LCSS Joint distance 72
B.3 DTW Joint distance 72

Acronyms

AIS	Automatic Identification System
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DTW	Dynamic Time Warping
COG	Course Over Ground
CPCC	CoPhenetic Coefficient Correlation
ED	Euclidean Distance
EDR	Edit Distance on Real Sequence
EM	Expected-Maximization
ERP	Edit Distance with Real Penalty
HAC	Hierarchical Agglomerative Clustering
IMO	International Maritime Organization
LCSS	Longest Common Subsequence
LSTM	Long-Short Term Networks
MDL	Minimum Description Length
ML	Machine Learning
MMSI	Maritime Mobile Service Identity
NATO	North Atlantic Treaty Organization
OPTICS	Ordering Points To Identify the Clustering Structure
SOG	Speed Over Ground

1

Introduction

The maritime environment is a great source of economic growth, providing natural resources and access to trade and transport. As outlined by the North Atlantic Treaty Organization (NATO) [5], not only the Earth's surface is covered by 70% water, but 80% of the world's population lives within 100 miles of the coast, and 90% of the world's commerce is made through the sea. This places the world's oceans and seas sustainability and security a high concern for many nations and international organizations, which have the responsibility to control ships crossing territorial and open waters.

The incorporation of the Automatic Identification System (AIS) technology on vessels, an automated on-board tracking device, has supported maritime surveillance authorities by broadcasting among AIS systems (e.g., vessels, on-ground base stations) information stating who they are, where they are and their movement details. Although it was initially conceived for vessel collision avoidance, its use has been extended as a means to achieve a deeper and broader knowledge of maritime situations since its mandatory deployment by the International Maritime Organization (IMO) on a wide range of vessels in 2004 [6]. Large amounts of near-real-time AIS data started to be available, demanding the creation of methods to automatically transform raw data into meaningful information and easily support operational decision makers.

However, it's not always possible to have a continuous observation of vessels at sea: the device may stop working or transmit wrong data due to some malfunction, the crew can turn off the device, and the 20 nautical miles depth of the transmission link may not cover part of the sailed area [7]. These coverage failures clearly represent high risks, such as collision situations, maritime pollution, piracy or unauthorized maritime arrivals [8].

A broad area of trajectory data mining is devoted to addressing this issue, known as trajectory uncertainty [9]. A common approach is to find, amid a set of trajectories, one or more samples that follow the same (apparent) path as the uncertain one and have more information. Consequently, the gaps can be filled with the information retrieved from the sequences of higher extent. This concept is nothing but route prediction, as the movement of trajectories on the same route and observed as a whole serve as basis for the next positions of the (yet) partial ones.

Predicting the destination of a given object is highly tied with the discovery of main paths [10]. The path clustering produces an explicit representation of the intrinsic traffic patterns, which can be further used to retrieve the expected course alterations of vessels on one of the routes, namely to obtain future positions.

In this context, the present work proposes to tackle the problem of long-term vessel position estimation, which comprises the process of predicting ship movements well beyond any available positioning data and based on the movement of past vessels on the same route [11]. More specifically, this covers trajectory pattern mining tasks and methods of Machine Learning (ML), namely clustering, to discover motion patterns – routes — from historical AIS data collections. From the clusters, a compact represen-

tation is extracted to model each of the existing paths, ultimately used to forecast future positions.

Applications of route prediction range from traffic efficiency [12], ship avoidance collision [13] and abnormal movement detection [8] [11] [14].

1.1 Project Goal

The goal of this work is to perform a long-term prediction of the movement of vessel trajectories. In particular, the objectives are to:

1. Approach the clustering of trajectories through a distance-based method.
2. Predict vessel movement through a probabilistic approach given the set of nearest routes.
3. Access the system's performance in forecasting ship future movement.

The main contribution of this thesis is the prediction of vessel trajectories through a probabilistic approach of the set of nearest routes.

1.2 Document Structure

With the exception of the introduction, where the scope of the project is established, this thesis is further organized in the following way:

- Section Chapter 2 provides a review of the existing literature covering the different subjects of this thesis.
- Section Chapter 3 is dedicated to data analysis and knowledge extraction. It includes the description of the data sample used for this study and the pre-processing steps as well as the clustering approach.
- Section Chapter 4 details the prediction methodology and the results obtained.
- Section Chapter 5 contains the work's conclusions and enumerates, as a final remark, the possible directions that could be taken to improve the work presented.

2

Related Work

This chapter presents the state-of-the-art research made to understand several topics regarding the scope of the project. A general review on trajectory data mining is given in section 2.1, introducing trajectories on the time series concept and the main tasks underlying this field. A discussion on trajectory learning is made in section 2.2, dividing the existing approaches into its different categories and detailing the subjacent methods. Ultimately, on chapter 4 are analyzed the different approaches on long-term route prediction, the main focus of this thesis.

2.1 Trajectory Data Mining

A spatial trajectory is a type of time series where the set of recorded variables over spaced timesteps is related to geographical points. With a wide range of devices providing location-based data (e.g., GPS, RFID, GSM beacons, smartphone sensors), trajectories model the movement of different targets, such as people, animals, vehicles, or even natural phenomenons such as hurricanes [10]. These devices offer a valuable source of knowledge for gaining insights about moving objects, such as forecasting, control, understanding features of the data, etc.

Using the taxonomy extracted from [9], the main tasks related to the field of trajectory data mining are divided into the following categories:

- Pattern mining: analysis of motion patterns of one or more trajectories.
  (a) Moving together patterns: similar movements in certain consecutive time windows.
  (b) Group (Clustering): trends over the whole path of a cluster of objects.
  (c) Sequential: common locations in similar timesteps (possibility spaced).
  (d) Periodic: behaviors that repeat themselves over a period of time.
- Classification: distinguishing trajectories with known categories to obtain semantic value. Applications are, for instance, deriving a vessel's activity (fishing or sailing).
- Uncertainty Modelling: trajectories are often available intermittently, and techniques to make them more complete are useful. On the other hand, one may want to make them uncertain for privacy concerns.
- Outlier/Anomaly Detection: definition of models of normal behavior as a means to detect abnormal actions.

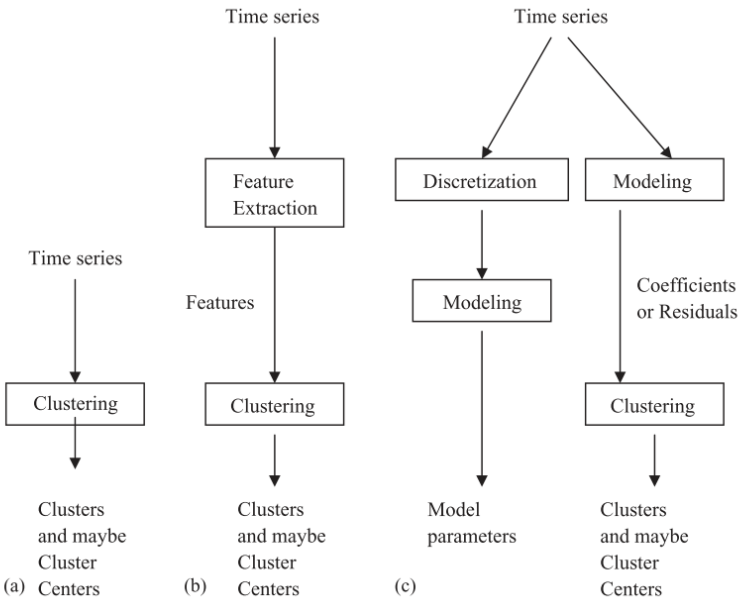
2.2 Trajectory Learning

As a group pattern mining task, the learning of trajectories seeks to capture the existing but unknown motion-patterns from trajectory data, where there are multiple sets of objects navigating through the

same path. General path discovery applications are based on finding the most frequent routes, but this is also extended to the shortest ones, fastest, etc. [10]. In the current context, we're interested in studying the most popular ones, and that is the type covered in this research.

Trajectory learning employs ML unsupervised techniques that are separated in [15] into three main approaches, as shown in Figure 2.1: (a) distance-based, (b) feature-based and (c) model-based. In the present work, we join the feature-based into distance-based approaches when examining the literature, due to employing the same idea of directly comparing trajectories when clustering.

Figure 2.1: Time-series clustering categorization.



2.2.1 Distance-based Approaches

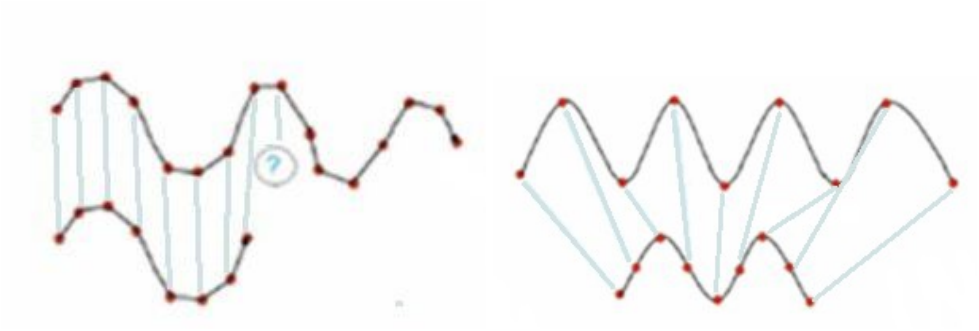
They comprehend the set of methods that groups trajectories by directly comparing how similar they are. In this category, clustering algorithms are employed, as they are able to automatically infer the hidden structure of data according to their distinguishing properties and allocate the data into classes of similar objects [16]. In essence, the methodology utilized in this category is reduced to a) choosing a clustering algorithm that determines how trajectories are gathered together and b) choosing a similarity measure that establishes the candidates to be in the same group [17].

2.2.1.A Similarity Measures

The notion of (dis)similarity is given according to a measure quantifying how close the data points are: the lower the distance the more similar they are and vice-versa. When dealing with simple data types

such as singular points in space, the concept of distance is quite direct, and traditional measures such as the Euclidean Distance (ED) offer a good notion of the proximity between them. However, trajectory data, as a sequence of multi-dimensional points in time, is tied with length, shape and a (often uneven) spatial distribution, so to capture the similarity between these structures is not a direct task [18]. Taking the example of the aforementioned ED, it compares points of common indexes, one by one, with a straight line, i.e. the i^{th} element of one trajectory is compared only with the i^{th} element of the other trajectory [19]. Intuitively it's only able to work with equal-length sequences (see Figure 2.2(a)), and even when there are the same number of points the measure performs poorly if they are unaligned in time, due to different sampling rates/speeds (see Figure 2.2(b)).

Figure 2.2: Challenges of measuring similarity between trajectories.



(a) Trajectories with different length.

(b) Trajectories with different sampling rate.

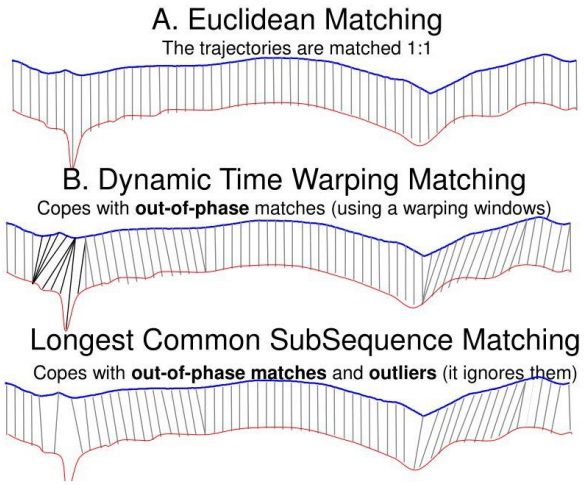
Some of the approaches to work around these issues include re-sampling processes that transform each trajectory to the same length and dimensionality reduction techniques, but this may not always be an easy task to perform with little loss of information [16]. On the other hand, measures allowing to pair up elements that are not in the same index in the data sequence have been introduced in the literature, and are classified into two broad groups [19]: shape-based distances, which only take into account spatial information, and warping based distances, that integrate both the spatial and temporal dimensions.

Regarding warping based distances, Dynamic Time Warping (DTW) [20] is one of the widely known measures that allow some points to be matched several times. Hence, the data sequences can stretch or shrink in order to get the best alignment. One of the problems arising from this method is that it can try to compensate for the difference between two sequences by repeating one point for too many times ("singularity"). The Derivative Dynamic Time Warping [21] was developed to tackle this later issue.

Following DTW, the Longest Common Subsequence (LCSS) [22] relaxes the matching procedure by restraining the calculus to the sequence positions that are met up until an ε threshold, so it leaves some points unmatched and focuses on the similar parts of two sequences. The matching can occur even if

the time indexes are different, regulated through a δ parameter that defines how far it can go.

Figure 2.3: Matching difference between ED, DTW, and LCSS [1].



Similarly to LCSS in applying thresholds for matching, the Edit Distance on Real Sequence (EDR) [23] further includes penalties for those left unmatched, which can be a reason for imprecisions in LCSS. From the same family, the Edit Distance with Real Penalty (ERP) [24] emerges from the need of a metric supporting both the triangle inequality (the aforementioned measures do not) and local time-shifting.

As an alternative to performing time-shifting, shaped-based distances remove the notion of time. In the Hausdorff distance it is measured how close two shapes are by picking the higher distance between the nearest points of the two sequences, so if every data point of both sequences is close to some points of the other, they are considered close [25]. This measure does not allow to differentiate sequences with opposite directions. The Fréchet distance is the maximum ED value among the distances required to connect any two points on trajectories, taking into account the location and sequential relationship between points [25].

2.2.1.B Clustering Methods

A myriad of clustering algorithms are available, but not all are suitable for trajectories. For example, K-means and Birch use Euclidean distance as a metric, thus require fixed-dimensional vectors and do not support time-shifting. Other algorithms offer more flexibility by allowing to represent the data with a matrix (as known as affinity matrix), instead of individual features, where each of its elements are the similarities between samples [15]. This is not only useful for using tailor-made metrics like the ones shown in section 3.3.1, but to overcome the problem of not having a suitable matrix representation with unequal-length data. In this context, some of the methods able to deal with trajectory data are Affinity Propagation, Density-Based Spatial Clustering of Applications with Noise (DBSCAN) and Hierarchical

Table 2.1: Overview of similarity measures characteristics, extracted from [3].

Measure	Meaning	RTN	LTS	DL	IS	CC
ED	Raw representation	✗	✗	✗	✓	O(n)
LCSS	Raw representation	✓	✓	✓	✗	O(n*m)
DTW	Raw representation	✗	✓	✓	✗	O(n*m)
TWED	Raw representation	✗	✓	✓	✓	O(n*m)
ERP	Raw representation	✗	✓	✓	✓	O(n*m)
EDR	Raw representation	✓	✓	✓	✗	O(n*m)
DTW based approach	Movement speed and path	✗	✓	✓	✗	O(n*m)
Trajectory Match	Movement Direction	✗	✗	✗	✗	O(n*m)
EDM	Movement Direction	✓	✗	✓	✗	O(n*m)
Hausdorff	Shape	✗	✗	✓	✓	O(n log n)
Fréchet	Shape	n.a.	✗	✓	✗	O(nm log (nm))
SPADe	Shape	✓	✓	✓	✗	O(n*m)
AMSS	Shape and direction	✓	✓	✓	✗	O(n*m)

Note: 'n.a.' means that the axiom is satisfied, but it is meaningless for that distance, 'RTN' means robust to noise, 'LTS' means local time-shifting, 'DL' mean different lengths, 'IS' means is metric and 'CC' means computation cost.

methods.

Moreover, the choice of the most fitting clustering method is also dependent on the data's composition (i.e., number of samples and clusters) and the clusters' morphology (e.g., shape and size). For instance, density-based approaches (e.g.: DBSCAN) produce maximal, dense clusters with no restrictions on the number, size and shape; center-based clustering methods (e.g.: k-means) produce compact, spherical clusters and are very sensitive to noisy outliers [17]. An analysis of these aspects is presented in Table 2.2.

Li W. *et al.* [26] used DBSCAN clustering to extract the most frequent routes from an AIS dataset. The clustering is coupled with the Longest Common Subsequence, where the Euclidean distance is used between latitude and longitude to obtain the similarity between trajectories. The study was conducted with 237 trajectories traveling in a region of nearly 23 km. Such coverage only includes two types of movement (straight and straight with left turn), making it difficult to assess the system's accuracy in more complex traffic zones. Furthermore, it does not integrate the trajectories course information, which may fail in detecting opposite directions.

A popular approach is to divide the area into a given number of cells, where each one of them is described by the movement properties of the vessels traveling that section [27]. These are named grid-

Table 2.2: Overview of clustering algorithms, extracted from [4].

Method name	Parameters	Scalability	Usecase	Geometry (metric used)
K-Means	number of clusters	Very large number of samples, medium number of clusters with MiniBatch code	General-purpose, even cluster size, flat geometry, not too many clusters	Distances between points
Affinity Propagation	damping, sample preference	Not scalable with number of samples	Many clusters, uneven cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbour graph)
Mean-shift	bandwidth	Not scalable with number of samples	Many clusters, uneven cluster size, non-flat geometry	Distances between points
Spectral clustering	number of clusters	Medium number of samples, small number of clusters	Few clusters, even cluster size, non-flat geometry	Graph distance (e.g. nearest-neighbour graph)
Ward hierarchical clustering	number of clusters or distance threshold	Large number of samples and number of clusters	Many clusters, possibly connectivity constraints	Distances between points
Agglomerative clustering	number of clusters or distance threshold, linkage type, distance	Large number of samples and clusters	Many clusters, possibly connectivity constraints, non Euclidean distances	Any pairwise distance
DBSCAN	neighbourhood size	Very large number of samples, medium number of clusters	Non-flat geometry, uneven cluster sizes	Distances between nearest points
OPTICS	minimum cluster membership	Very large number of samples, large number of clusters	Non-flat geometry, uneven cluster sizes, variable cluster density	Distances between points
Gaussian Mixtures	many	Not scalable	Flat geometry, good for density estimation	Mahalanobis distances to centers
Birch	branching factor, threshold, optional global cluster	Large number of clusters and samples	Large dataset, outlier removal, data reduction	Euclidean distance between points

based approaches and are appropriate for monitoring small areas. Wu l. *et al.* [12] employ this approach for computing shipping density, fast enough to be performed on a global scale (less than 56 hours). In this work, it took 56 hours to produce all the global monthly ship density, traffic density, and AIS receiving frequency maps, from August 2012 to April 2015.

Aiming to reduce computation time and improve model efficiency, Sheng P. and Yin J. [28] use instead characteristic points from vessel trajectories having a wider change on Speed Over Ground (SOG) and Course Over Ground (COG) and with the Minimum Description Length (MDL). The DBSCAN clustering model then integrates not only latitude and longitude data, for which the Hausdorff distance is used for similarity measurement, but also the speed and course information, evaluated with tailor-made measures. Finally, the representative trajectory is calculated with the sweep line approach. The evaluation was made with 1-month AIS data covering approximately 5 km, and the results showed that the method is able to extract patterns in accordance with the ones revealed in traffic schemes of the area.

Also using a compact representation of trajectories, Pallotta G. *et al.* [11] proposed the "Traffic Route Extraction for Anomaly Detection" focused on clustering events instead of trajectories. The underlying idea is to find groups of points called waypoints, associated with key events like entry, exit, and stoppage, in which trajectories are characterized as a set of straight paths connecting the waypoints passed through. The waypoints are incrementally activated upon vessel traffic and clustered with DBSCAN based on the event type and scene features, originating routes when the vessels with matching start and end waypoints reach a certain threshold. Route objects are statistically described by the kinematic features of the vessels that originated them. This approach can easily tackle the problems of unequal-length trajectories, incomplete paths or with gaps. More point-based approaches have been developed in the maritime setting in [29] and [30]. As in [28], this kind of approach allows addressing more efficiently the burden of modeling broad and dense areas.

On the other hand, Yao D. *et al.* [31] apply deep learning techniques in order to work with lower-dimensional data. The trajectories are transformed into a feature sequence with Long-Short Term Networks (LSTM) to which is employed a Seq2seq autoencoder to learn a fixed-length representation. Therefore, K-means was used for clustering. The first experiment was made with synthetic data, encompassing 3000 trajectories and three movement patterns. The second experiment was done with a real dataset of 1-month AIS data and consisted of 4700 trajectories from four different types of vessels. In the former, it was compared with LCSS, DTW, EDR, and Hausdorff distance on K-medoids, and their approach outperformed all the remaining with an 87.5% accuracy. The latter test showed that the clustering was able to form groups of movements where the prevailing instances belonged to the same ship type.

The work mentioned so far deals with trajectories as a whole, meaning that the paths are compared on their totality and cases where only some sections are similar are thus considered distinct. The idea

of clustering sub-trajectories is particularly useful when analyzing specific regions of interest, e.g., behavior of hurricanes near the coast or in open waters, as highlighted by Lee J. and Whang K. [32]. On this matter, the authors developed ”TRACLUS”, a framework that partitions each trajectory into a series of meaningful parts, by finding its characteristic points through the aforementioned MDL principle. In its turn, the parts are grouped with a tailored density-based clustering algorithm to detect similar sub-paths. They delineate a distance measure based on the Hausdorff measure as the weighted sum of 3 properties: the directional difference (angle distance); the physical shift between trajectories (perpendicular distance), calculated with the ED; the length difference between the trajectories (parallel distance), measured with the ED. The tests were made on hurricane and animal movement data.

2.2.2 Model-based Approaches

This type of approach enclose clustering techniques where the data is characterized through a given model or as a mixture of probability distributions, that is used to group instances generated by the same model [15]. Thus, each model represents a cluster. This category is further divided into statistical models and neural networks [15]. In [22], it is highlighted that these types of approaches can suffer from scalability problems; besides that, assuming that the data (trajectories) follows some model, is not a straightforward task to complete and describe real datasets. Also, when the clusters are close between them, the performance may be degraded [33].

A common technique used in neural network approaches is Self-Organizing Maps [34], whose clustering process is based on a projection to a lower-dimensional space. The authors in [35] apply it to cluster simulated vessel data regarding the south region of Sweden and use speed and heading for that purpose. It has the advantage of offering a visual representation of the data’s structure, although it’s not able to work with variable-length time-series [33].

On the other hand, Gaffney S. and Smyth P. [36] presented a probabilistic model-based clustering by representing trajectory data with a mixture of regression models, where the Expected-Maximization (EM) is used to estimate the cluster’s parameters. Experiments were made with synthetic and video streams data, and compared with k-means and Gaussian Mixture Models, achieved better results. The authors later considered discrete-valued shifts along the time axis and real-valued additive offset on the independent variables of trajectories within clusters [37].

Alon *et al.* [38] also proposed a statistical model for clustering motion time-series data, where the clusters are described with Hidden Markov Models. Again, the EM algorithm is used to estimate the model parameters, and a soft cluster assignment is determined by the posterior probability, which depends via Bayes rule on the cluster a priori probability and the data likelihood. The evaluation was made with synthetic and real data, where the ground truth was available for both, and the method was compared with k-means. The results showed similar results for both approaches, although the EM performed

best in the presence of overlapping densities.

Ramoni M *et al.* [39] introduced the ”Bayesian Clustering by Dynamics” algorithm that models univariate time series with discrete values as Markov Chains and finds the most probable set of clusters given the observed time series through a Bayesian model. The method was applied to robot sensor data and compared with mixture model EM-Clustering, for which the algorithm obtained the best results. The authors later applied this approach to the multivariate case [40].

Aiming to simplify the choice of an appropriate model to fit the data, Zhong S. and Ghosh J. [41] proposed a unified framework for model-based clustering.

2.3 Route Prediction

The approaches regarding this topic can be categorized into three types [42]: physical model-based methods, learning model-based methods and hybrid methods.

2.3.1 Physical Models

Physical Models are based on physical laws that consider the various aspects affecting the motion (e.g., force, mass, inertia). Although useful for building simulation systems, they require strict environmental and state conditions that are hard to obtain in practice.

Kalman Filters are a widely used technique in this category [43], initially created to solve the problem of minimizing the mean least square error for linear systems under Gaussian noise [44].

2.3.2 Learning Model-Based Approaches

Learning models consider that the aspects that physical models include can be estimated automatically by creating movement models of past behaviors, as they reflect what is considered normal upon the existing conditions.

Ristic B. *et al.* [8] apply a Kernel Density Estimator (KDE) to previously extracted motion patterns with a particle filter to predict the most probable future state (position and velocity).

In [26], an LSTM was used for path modeling, after conducting a distance-based clustering. The LSTMs are applied to each unraveled route, where an iterative forecasting process is made to get the future values of both position, speed, and course, given the current ones. The prediction was made on a 10-minute window and the method was compared to Kalman Filters and BP neural networks, LSTMs achieving higher accuracy.

Given the problems of vanishing or exploding gradients with LSTMs, Gao *et al.* [45] use Bidirectional-Long Short Term Memory Networks instead. The information regarding position, heading, and time was

chosen to perform the model's training from a 10-month data sample. Compared to an LSTM and a BI-RNN, the BI-LSTM achieved the best results with a 10 m error.

For predicting tracks in [11], the Bayes Rule is used to get the posterior probability of following one of the existing routes, given the current position and velocity, from which the future positions of the route with the highest probability are extracted with kinematic equations.

Rong H. *et al.* [46] provide a probabilistic ship prediction method that incorporates the notion of uncertainty. The uncertainty is defined by the ship position probability density functions on lateral and longitudinal directions at discrete timestamps, where a Gaussian Process model is used to describe the former and acceleration distributions for the latter. The dataset used was from the coast of Portugal, with AIS data from 3 months containing 1396 trips from cargo and tanker vessels. The results showed a 900 m prediction error for the lateral position and low errors for 60-minute prediction for the longitudinal position.

2.3.3 Hybrid Models

These models either combine the hybrid and learning models or use multiple learning methods.

Perera L. *et al.* [44] exploit a hybrid approach with Extended Kalman Filters (EKF), a variation able to handle nonlinearities, to estimate position, velocity, and acceleration. The method is incorporated in a Curvilinear Motion Model that is used to describe the movement. The results showed small errors in estimating vessel speed and acceleration from noised.

3

Data Analysis and Clustering

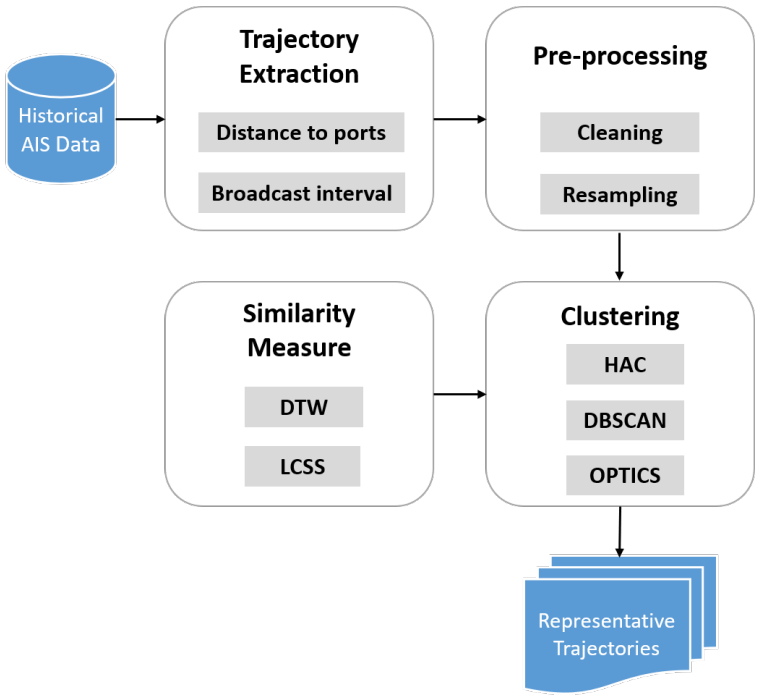
The usual steps taken for the creation of prediction methods require the extraction of the main routes of the region [13]. Each route is a description of a movement pattern over time and can be used to have an estimation of the vessel movement.

The present chapter covers the clustering approach that is used to extract these routes patterns. We start by describing the composition and characteristics of the data set that will be used in section 3.1, followed by the pre-processing taken to uniformly resample trajectories in section 3.2. Then, the most fitted clustering algorithms are proposed and evaluated along with a primary assessment of the similarity measure in section 3.3.

The development of this component is divided into the following main tasks:

1. Designate a distance measure to accurately quantify how similar two trajectories are.
2. Cluster historical AIS data to extract route patterns, using the similarity measure from task 1.
3. Extract representative trajectories for each obtained motion pattern on task 2.

Figure 3.1: Data analysis and clustering approach.



3.1 Data Sample

The dataset used in this project is composed of a real-world AIS dataset[1] from the region of Brittany, in France, collected by the French Naval Academy receiver. It covers a period of 6 months from 01-

[1] http://datacron-project.eu/

10-2015 to 31-03-2016 in an area delimited by latitude between 45º and 51º and longitude between -10º and 0º. The dataset contains two main tables, differing on the type of information: dynamic data, which contemplates AIS messages Table A.1; static data, containing information regarding the vessel's characteristics and voyage related data Table A.2. Using the Maritime Mobile Service Identity (MMSI) to link each dynamic message to the static ones, we chose to use the following subset of attributes:

Table 3.1: Data record sample.

MMSI	Lon	Lat	SOG	COG	T	Ship_Type
228854000	-4.3472633	48.118046	10.4	257	1443650401	Cargo
245257000	-4.4657183	48.38249	0.1	36	1443650402	Fishing
227705102	-4.4965715	48.38242	0	259	1443650403	Fishing
228131600	-4.644325	48.092247	8.5	258	1443650404	Tanker

Regarding the vessel's trajectories, it was assumed that a new trip was made when the delta time between 2 broadcasted messages by a given vessel was greater than 24 hours or when the vessel's position was less than 5 km from a port.

All records with duplicated values for both longitude, latitude, and date from a given vessel were removed, as well as all vessels with the ship type field not filled. Unfeasible speed or infeasible position messages were removed from the set. Tracks whose speed is lower than 0.1 knots for more than 80% of the time (at anchor or moored vessels) were also removed. All trips were required to have at least 3 data points. After the data cleaning, the data sample contains a total of 1 461 000 records, 2 045 vessels, and 5 157 trips.

In this thesis, we're interested in studying long and repetitive movements that characterize the main global routes, thus we will only work with the cargo and tanker AIS messages. This subset contains a total of 3187 trajectories.

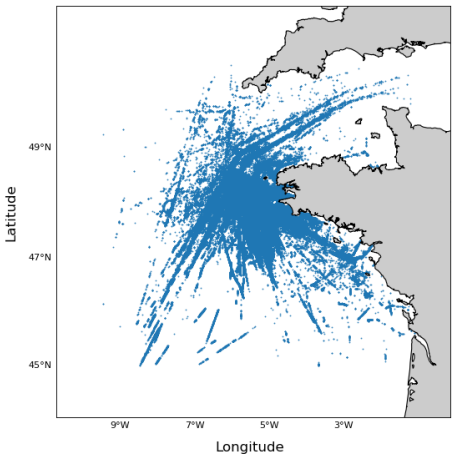
As can be seen in Figure 3.2(e), the samples are composed of trajectories of different lengths. It makes sense to always include trajectories of a greater extent since we're interested in finding part of main global routes. On the other hand, most of the trajectories have less than 200 km sailed, which is a small distance for a $\approx 500 \times 800$ km area regarding long-term paths, so we excluded those with less 100 km because they were not a meaningful portion of the data. To sum up, the data sample was further divided into three sets according to the sailed distance: the 1^{st} contains trajectories with more than 100 km (total of 1381 trajectories); the 2^{nd} contains trajectories with more than 200 km (total of 754 trajectories); and the 3^{rd} contains trajectories with more than 300 km (total of 183 trajectories).

3.2 Data Treatment

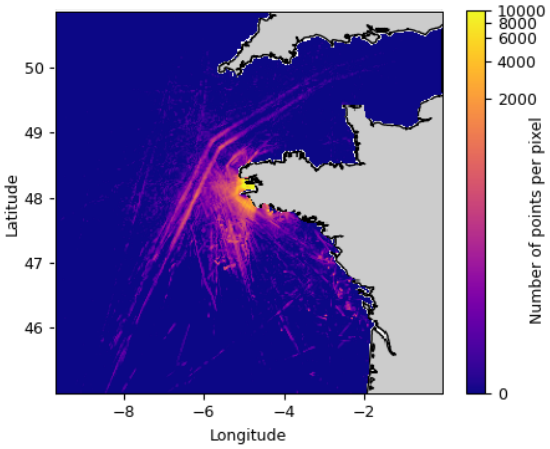
For the purposes of forecasting movement in regular time-intervals, it is required for trajectories to have observations of fixed and uniform frequency. Moreover, the gaps between two consecutive sampling

Figure 3.2: Dataset visualization and statistics.

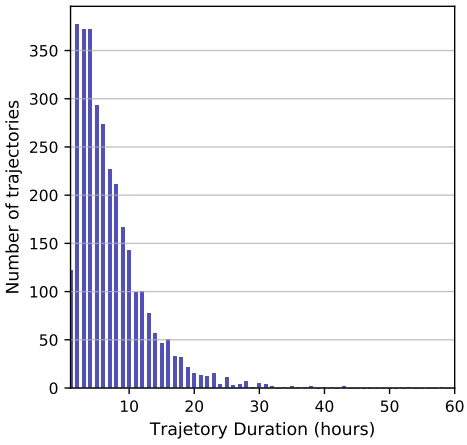
(a) $\approx 500 \times 800$ km area.



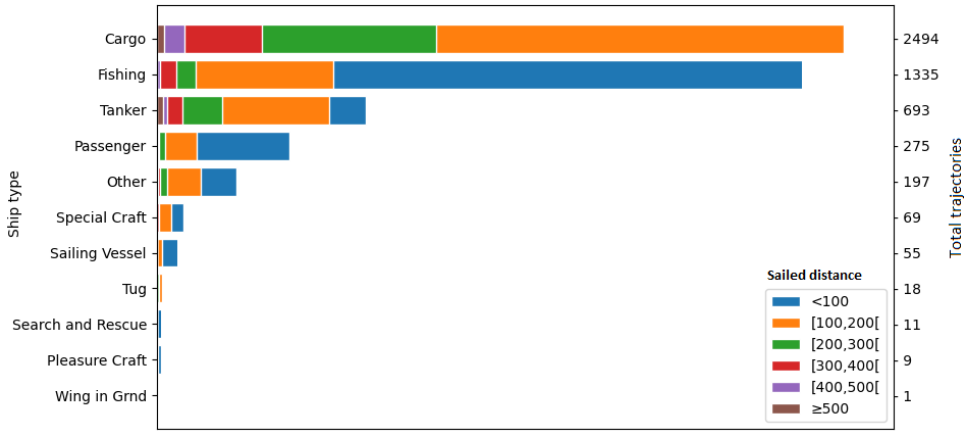
(b) AIS positions in the covered area.



(c) Positions heatmap.



(d) Trajectory duration distribution.

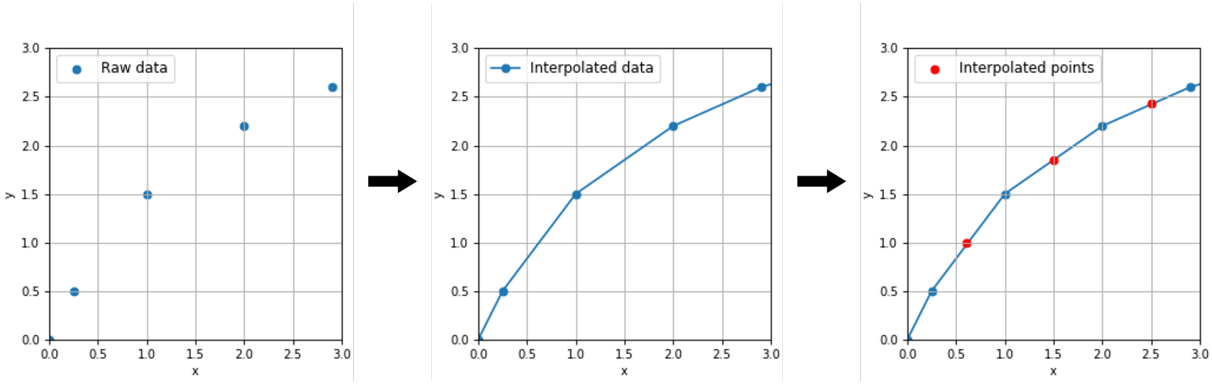


(e) Trajectories length distribution per ship type.

points can be wide enough to fail in capturing the trajectory movement, which may introduce errors when measuring distances.

The dataset contains irregularly sampled messages, so a pre-processing step is thus needed to transform the trajectories to the same broadcasting time interval, namely resampling. A natural way of resampling data is to represent it through a mathematical function that intersects the set of discrete points, from which new points are easy to retrieve in any range of the representation [47], as demonstrated in Figure 3.3.

Figure 3.3: Interpolation.



The most common methods of interpolation use polynomials or spline functions [47]. Both are easy to differentiate, integrate and evaluate, but splines can keep the function order low by dividing the representation into several pieces [48], and exhibit flexibility for both smoothing and interpolation [47]. Accordingly, splines were used to perform resampling in this work.

3.2.1 Splines

Spline functions are piecewise polynomials, which can be linear, quadratic, cubic, etc., according to the polynomial's degree. For means of simplicity, linear splines were used, and that is the type covered in the remainder of this section.

For a given set of strictly increasing points $\{x_i, y_i\}_{i=1}^{n}$, the spline function is defined as $y_i = f(x_i)$, where $f(x)$ takes the form [49]:

$$f(x) = \begin{cases} p_1(x), & \text{if } x_1 \leq x < x_2 \\ p_2(x), & \text{if } x_2 \leq x < x_3 \\ \vdots & \\ p_{n-1}(x), & \text{if } x_{n-1} \leq x < x_n \end{cases} \tag{3.1}$$

with each $p_i(x)$ being the polynomials of 1^{st} degree, defined as

$$
\begin{aligned}
p_i(x) &= a_i + b_i x \\
&= \underbrace{y_i}_{p_i(x_i)} + \underbrace{\frac{y_{i+1} - y_i}{x_{i+1} - x_i}}_{p_i'(x_i)}(x - x_i)
\end{aligned} \tag{3.2}
$$

The problem is reduced to calculate the $2(n-1)$ coefficients of the $n-1$ linear polynomials. The piecewise function f(x) will interpolate all data points, and it is assumed that it must be continuous at each one of them, yielding:

$$
p_i(x_i) = p_{i+1}(x_i) = y_i \tag{3.3}
$$

Up to now, splines fit 1-dimensional data, but trajectory data can have multiple dimensions. In fact, each dimension is an individual sequence of values to which splines can be directly applied. By joining the resulting vectors, the interpolated trajectory is formed. Naturally, this is only possible if the independent variable is the same for all the fitted dimensions so that each new value belongs to the same measurement. In this case, time was used.

The interpolation module from the Python-based SciPy library[2] provides the resources to carry out the spline fitting and resampling. The method *UnivariateSpline* was used to create linear splines representations and evaluate new points with a step size of 300 (5-minute interval) regarding the latitude and longitude dimensions.

The final step involves re-calculating the heading (*bearing* function from geo-py library[3]) and speed ($\frac{1}{1.85200} \times \frac{\Delta x}{\Delta t}$) given the new set of points.

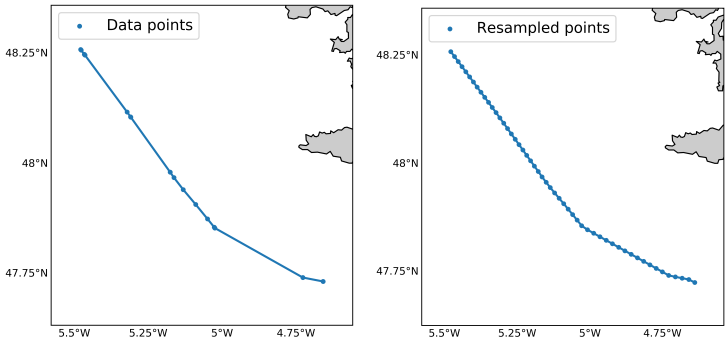
3.3 Clustering Method

Choosing a clustering method is highly dependent on the data's characteristics, so the technique must satisfy the following requirements:

1. No size uniformization needs to be made.
2. The clusters can have different densities.
3. The clusters can have an irregular shape.

[2] https://www.scipy.org/
[3] https://pypi.org/project/geo-py/

Figure 3.4: Example trajectory resampling.



(a) Original trajectory. (b) Resampled trajectory.

On this note, from the methods that are able to work with trajectories, three were chosen: Agglomerative Clustering, DBSCAN, and OPTICS. Under the Python's machine learning framework scikit-learn[4], which provides implementations of all methods in the clustering module, the algorithms were further compared in section 3.3.5.

From each of the obtained clusters, a representative trajectory was retrieved that provides the underlying baseline movement of the cluster and materializes one of the main routes. In technical terms, this trajectory is the cluster centroid and in the present work is the one with the lowest sum of distances to all its respective cluster members.

At a later point, a re-classification was made in section 3.3.6 to evaluate the routes assignment.

3.3.1 Similarity Measure

The first step towards applying any clustering method is the definition of the similarity measure. To accurately measure the distance between the trajectories at hand, the method must be able to assess the affinity between each pair of trajectories even if:

1. The number of points differs.
2. The sampling rates are different.
3. The starting and ending positions are not the same.

Given that there is no agreement on what method is the best, we tested and compared the two most used measures in the literature, namely Dynamic Time Warping and the Longest Common Subsequence. Both can work with the conditions pointed above.

[4]https://scikit-learn.org/

3.3.1.A Longest Common Subsequence

Let $dist(x_n, y_n)$ represent the function measuring the distance between two data points, the Longest Common Subsequence $LCSS(X, Y)$ between two trajectories X and Y is recursively defined as follows [22]:

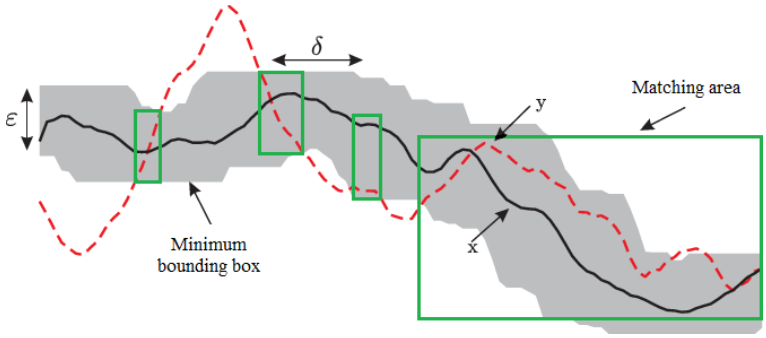
$$\begin{cases} 0, & \text{if } X \text{or } Y \text{ is empty} \\ 1 + LCSS_{\delta,\epsilon}(Rest(X), Rest(Y)), & \text{if } dist(x_i, y_j) \leq \epsilon \text{ and } |j - i| \leq \delta \\ \max \begin{cases} LCSS_{\delta,\epsilon}(Rest(X), Y) \\ LCSS_{\delta,\epsilon}(X, Rest(Y)) \end{cases} & \text{otherwise} \end{cases} \tag{3.4}$$

To retrieve a meaningful dissimilarity value from the length of the longest common subsequence, the percentage of that value regarding the length of the shortest trajectory is returned. Let min_length represent the number of points of the shortest trajectory, the distance $lcss\_dist(X, Y)$ between two trajectories X and Y is therefore obtained by:

$$lcss\_dist(X, Y) = \frac{LCSS(X, Y)}{min\_length(X, Y)} \times 100 \tag{3.5}$$

According to this definition, the values returned by $lcss\_dist$ range from 0 to 100, with the lowest value when two trajectories fully match, and vice-versa.

Figure 3.5: LCSS representation between trajectories x and y.



A – Position Data Initially, LCSS was tested with positional information, namely the longitude and latitude attributes. Since we are dealing with points on the surface of a sphere, the Euclidean distance does not apply to measure distances. To this end, the haversine formula was adopted, which determines the distance between geographic coordinates on the globe. Let R represent the average radius of the earth, the distance between points p_1 and p_2 along its two dimensions (longitude and latitude) is as follows:

$$haversine\_distance(p_1, p_2) = 2Rarcsin(\sqrt{sin^2(\frac{p_2^1 - p_1^1}{2}) + cos(p_1^1)cos(p_2^1)sin^2(\frac{p_2^2 - p_1^2}{2})}) \quad (3.6)$$

The implementation of LCSS from the traj-dist library[5] (*lcss*) developed in Python was used with the spherical option, which employs the haversine distance between 2D-coordinates. Several tests were made to choose the best distance threshold that can only be evaluated by visually checking the values for each pair of trajectories. Based on data visualizations of the existing routes, the value was based on the average width of the routes, and it was chosen a value of 19 km. The results showed that working only with this type of data does not allow us to identify trajectories in different directions.

B – Angle Data For the directional component, expressed by the COG attribute, the distance was measured by the cosine similarity, which is a widely used measure for assessing the affinity between angles. Taking the difference between two angles, when they are opposite (180º), the cosine takes the minimum value of -1, and when they are equal (0º or 360º), it takes the maximum value of 1. The cosine distance between two angles a_1 and a_2 is thus represented as:

$$cosine\_distance(a_1, a_2) = 1 - cosine(a_1 - a_2) \quad (3.7)$$

The code from Python's traj-dist library regarding LCSS was adapted to incorporate the cosine distance instead and is depicted in algorithm B.1 at section B. Regarding the distance threshold, a value of 10º was set for both datasets, estimated once more by visually checking different values until they were acceptable according to the trajectories at hand. In this case, only angle data is not able to determine that trajectories are far apart.

C – LCSS Joint Distance Measure A natural conception of the final measure is the average values of both position and angle. Taking the example where we have two trajectories close together (e.g., position distance to 0%) but with opposite directions (e.g., angle distance to 100%), the resulting distance is 50%, which is too optimistic for such difference. So, for the cases where one of these values is too low, this should be a straight indication that the trajectories differ greatly. For this reason, it was assumed that when either the distance with position or direction is greater than 60% then that was the value returned by the joint measure, while the remaining cases took the average. The pseudo-code for the joint distance is described in algorithm B.2 at section B in the annexes.

[5] https://github.com/bguillouet/traj-dist

3.3.1.B Dynamic Time Warping

Let $dist(x_n, y_n)$ the function measuring the distance between two data points, the Dynamic Time Warping DTW(X,Y) between two trajectories X and Y is defined as follows [3]:

$$\begin{cases} 0, & \text{if } X \text{ and } Y \text{ is empty;} \\ \infty, & \text{if } X \text{ or } Y \text{ is empty;} \\ dist(x_1, y_1) + min \begin{cases} DTW(Rest(X), Rest(Y)) \\ DTW(Rest(X), Y) \\ DTW(X, Rest(Y)) \end{cases} & \text{, otherwise.} \end{cases} \tag{3.8}$$

DTW requires all points from two trajectories to be matched. Hence when measuring two trajectories with different lengths, an overflow is introduced from the remaining points of the longest sequence, and the measure would perform poorly. That being so, a subsequence from the longest trajectory corresponding to the nearest points of the beginning and end of the shorter trajectory was used.

A – DTW Joint Distance Measure Due to the same issues as with LCSS, both position and direction data were used in an averaged joint distance. The implementation of DTW from tslearn[6] (function *dtw_from_metric*) was used once more using the haversine distance, for positions, and the cosine distance, for directions. The pseudo-code is shown in algorithm B.3 on attachments.

3.3.1.C Results

Comparing the results of LCSS and DTW for the same subset of samples, it is possible to conclude that both measures were able to identify when the routes matched entirely (or almost) (figs. 3.7(a) to 3.7(c) and figs. 3.6(a) to 3.6(c)), and when they were far apart (Figures 3.6(g) and 3.7(g)).

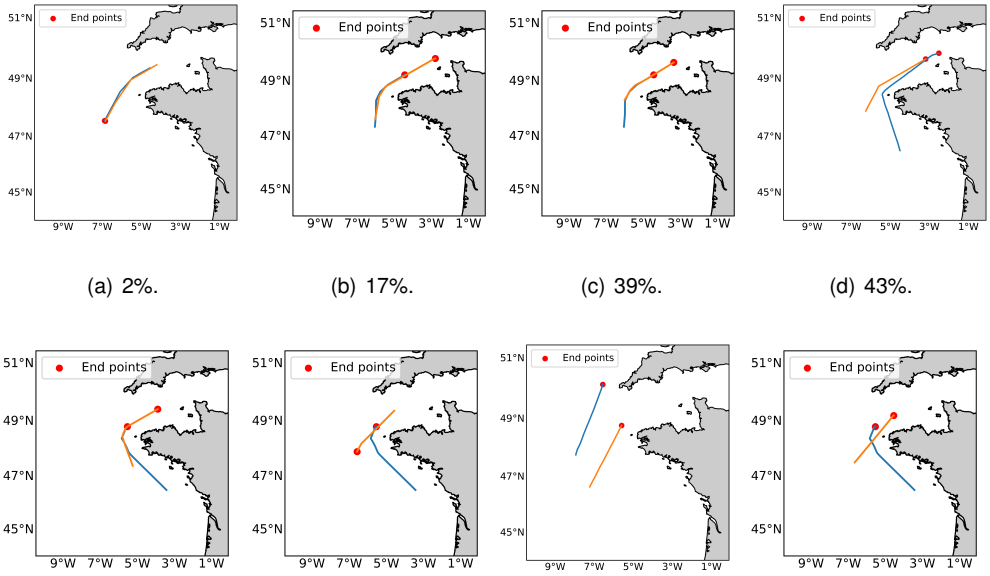
On the one hand, Dynamic Time Warping did not give fair values when the routes were close but with different shape or direction, to the extent that lower values were associated to trajectories with different directions in Figure 3.7(d) and higher to partially identical trajectories in Figures 3.7(e) and 3.7(g). This can be explained by the fact that DTW simply calculates the distances between aligned points. So, when trajectories are too close, this makes it enough to dictate that the routes might be similar, even though they have opposite directions. In spite of the fact that using a subset from the longest trajectory helps to tackle the sum excesses from unequal-length sequences, this distance may become even shorter because there are even less points to add.

On the other hand, Longest Common Subsequence provided reasonable values for all examples, and they are easier to understand than those from DTW. By using an envelope wrapping the acceptable area it focuses on the similar parts. So, in Figure 3.6(e) the similarity value is exactly the amount of space where they intersect. Furthermore, this allowed to also attribute automatically higher values for opposite

[6] https://tslearn.readthedocs.io/

courses, as in Figure 3.6(f), which DTW does not. In conclusion, the Longest Common Subsequence had the best results.

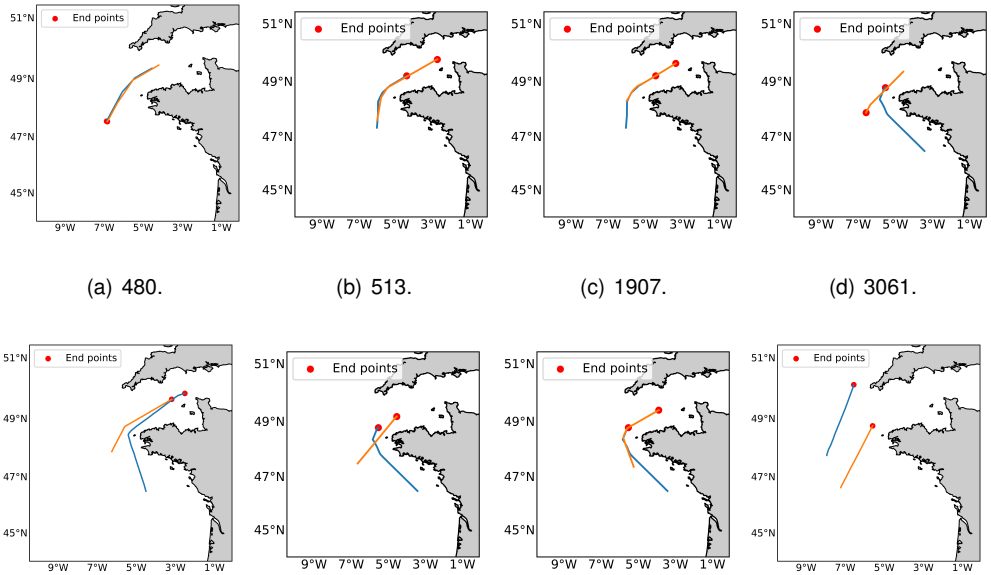
Figure 3.6: Sample ordered by the LCSS joint distance values.



(a) 2%. (b) 17%. (c) 39%. (d) 43%.

(e) 67%. (f) 100%. (g) 100%. (h) 100% .

Figure 3.7: Sample ordered by DTW joint distance values.



(a) 480. (b) 513. (c) 1907. (d) 3061.

(e) 3416. (f) 3970. (g) 5073. (h) 10810.

3.3.2 Agglomerative Clustering

Hierarchical Agglomerative Clustering (HAC) falls into the hierarchical methods category, which builds a tree, also known as a dendrogram, where each node contains a set of samples that are merged or split repeatedly [16]. The root node contains all the samples, and the leaf nodes group each sample individually. In a bottom-up approach, as in HAC, it starts with all samples in individual clusters, merging nodes that minimize one of the following linkage criteria [50]:

- Average or connectedness: the distance between two clusters is the mean of the distances between each pair of observations from both sets.
- Complete or diameter: uses the maximum distance between the observations of both nodes.
- Single or minimum variance: considers the minimum of the distances between observations of both sets.

It has been observed that the single and complete-link may suffer from the chaining and crowding effects, respectively: in the first case, some samples form a junction point connecting clusters and, in some scenarios, this leads to undesirable merges simply due to one bridging sample that verifies the minimum distance to some other sample; secondly, nodes are merged continuously if the maximum distance is not reached where maybe splits should be made in the meantime. The average linkage on the other hand does not, but may cause elongated clusters to split and for portions of neighboring elongated clusters to merge.

It was used the function *AgglomerativeClustering* from scikit-learn. Because the trajectories do not have the same number of points and there is no suitable data representation to feed the clustering method, the input to the algorithm was a pre-computed distance matrix with the measure defined in section 3.3.1.A. The distance matrix is an $N \times N$ matrix containing the LCSS joint distance between each pair of the N samples. Additionally, the linkage criteria needs to be set, and, when the number of clusters is unknown, a distance threshold is required to establish up until when clusters are merged.

3.3.2.A Model Selection

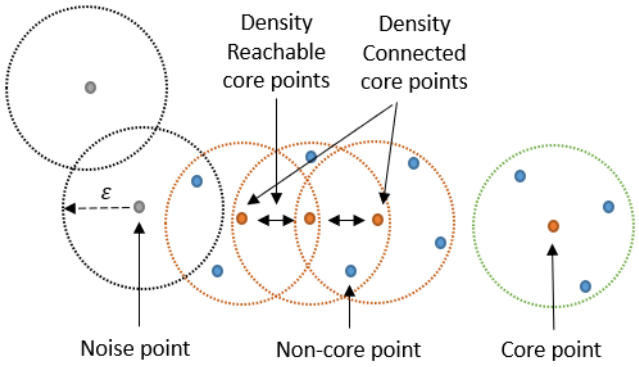
For clusters obtained through hierarchical approaches, a standard evaluation measure for finding the best linkage criteria is the CoPhenetic Coefficient Correlation (CPCC) [51]. The cophenetic distance is equivalent to the minimum distances between any of the samples that are to be included in the same node, (i.e., it measures how similar two objects have to be in order to be grouped in the same cluster). A matrix is created with this distance and is further correlated with the data's distance matrix to quantify how accurately the pairwise distances were preserved.

The common approach to find an ideal distance threshold is to analyze the resulting dendrogram for a given link and look for groups that combine at a higher dendrogram level.

3.3.3 DBSCAN

This technique incorporates the notion of density, where clusters are high-density areas followed by low-density areas [52]. These higher density areas are obtained by retrieving the data points that have at least k neighbors in an ε radius — core points —, each one of them formalizing a cluster. Subsequently, these samples are further grouped with the neighbors that are core as well — density reachable core points —, along with their neighborhood. This verification is made for every sample in the ε-neighborhood of every core sample, allowing to iteratively create a network of clusters even if two clusters are not directly reachable — density connected core points. A cluster emerges as a set of core points and the points that are close to one of the cores but are not themselves core — non-core points. The low-density areas — noise/outliers — are then the non-core samples that are not close to any core [4]. The density concept is illustrated in Figure 3.8.

Figure 3.8: Density representation of DBSCAN with k=3 and arbitrary ε.



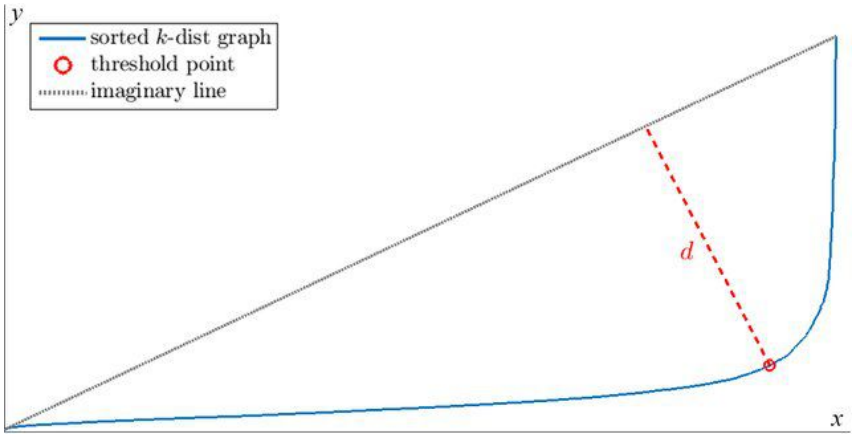
The algorithm requires the minimum number of neighbors and the maximum distance between samples to be set, along with a distance metric. While the former indicates how tolerant the algorithm is towards noise (lower values means more tolerance and vice versa), the latter defines the proportions of which the neighborhood can increase (too small means most data will not be clustered at all and labeled as noise, too large causes close clusters to be merged and eventually resulting in one big cluster [4]). Higher values of the number of neighbors or lower distance thresholds between samples suggest a higher density to find a cluster.

The function *DBSCAN* from scikit-learn was used.

3.3.3.A Model Selection

For a given k, the respective ε can be automatically extracted by taking the distance from the k^{th}-nearest neighbors of every point, sort them increasingly, and plot the points in the so-called k-dist plot [51]. The ideal ε is evaluated at the point where the distance has a sharper rise. In others words, it is equivalent to the furthest point between the k-dist plot and the line joining the first data point to the last one. Such a point is illustrated in fig. 3.9.

Figure 3.9: K-dist plot [2]



There is no technique for finding the k, but values between 3 and 24 have been used with most datasets in the literature, for density-based methods [53]. In this work, values in [3,6,9] were tested.

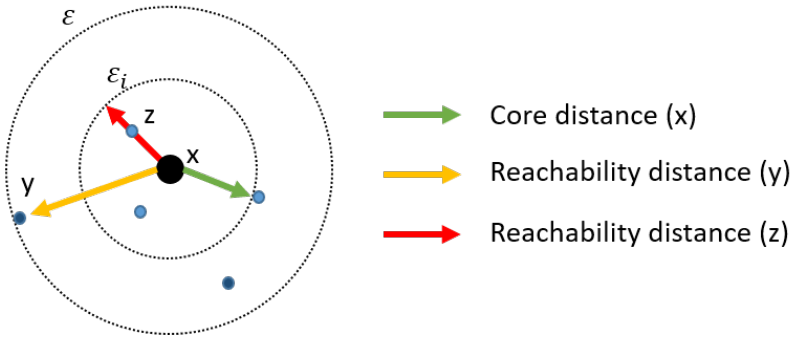
3.3.4 OPTICS

Ordering Points To Identify the Clustering Structure (OPTICS) [54] closely follows DBSCAN but extends its strategy in order to extract more effectively clusters of great varying density, where sub-clusters can be taken from a single cluster. In short, the algorithm takes the ε parameter as a value range ($0 \leq \varepsilon_i \leq \varepsilon$) and explores clustering with multiple ε_i to find different densities. The following attributes are defined for each data sample to accomplish this:

- Core distance: the minimal ε_i from which a given point is a core point for a given k. This distance is thus undefined for non-core and noise points.
- Reachability distance: the distance between two directly reachable samples, which cannot be smaller than the core distance. It is undefined if the point we are comparing to is not a core point.

A reachability graph is then created by ordering the data points with respect to their lowest reachability distance, such that spatially near points are neighbors in the ordering. Furthermore, the core distance

Figure 3.10: Core and reachability distance, for $k = 3$.



is used to define the density that must be accepted for a cluster so that both points belong to the same cluster.

In practice, OPTICS requires only the minimum number of neighbors; the parameter ε can be fixed to its maximum value to find clusters across all scales [4], and is mainly used to reduce computation time.

The function *OPTICS* was used from scikit-learn's clustering module, and the same values for ε in DBSCAN were tested. The default setting of ε to ∞ was adopted, to explore all possible ranges.

3.3.5 Results

Considering the three data samples described in section 3.1, the results for each clustering method are presented in this section. To create the distance matrix that is fed to the algorithms, the Longest Common Subsequence was used, because it provided better results and allows to easily interpret the distances.

To accomplish this, 70% of each data sample was used. The representative trajectories are shown in red, together with the trajectories' end point.

3.3.5.A Set 1

For HAC, the CFCC values in Table 3.2 gave to the average link the highest CFCC. Thus that was the linkage criteria used in the remaining experiments. The respective dendrogram was plotted in Figure 3.11 and showed a tree cut at a distance of approximately 70%. With this setting, the agglomerative technique produced 65 clusters (see Figure 3.12), where 32% was formed by only one trajectory (e.g., Figure 3.12(f)). The method experienced the introduction of trajectories that distort the overall shape of the cluster (e.g., Figures 3.12(a) and 3.12(d)), which can be caused by a high distance threshold, but lowering this value would result in more clusters with just one trajectory. In any case, the representative path of these clusters was according to the global shape. On this subject, working with trajectories of lower distance made the baseline movements to be smaller (e.g., Figure 3.12(d)).

Table 3.2: CFCC values for set 1.

Link	CFCC
Single	0.3602648533473505
Complete	0.46262985297177095
Average	0.8737476085525828

Figure 3.11: Top 15 levels of the hierarchical clustering dendrogram with average link, for set 1.

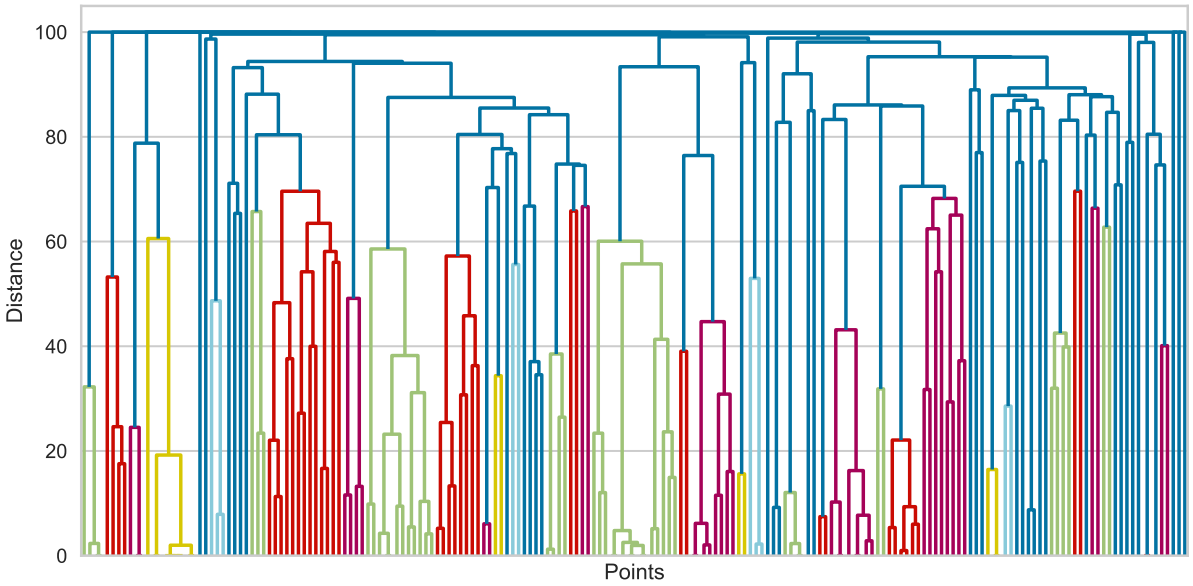
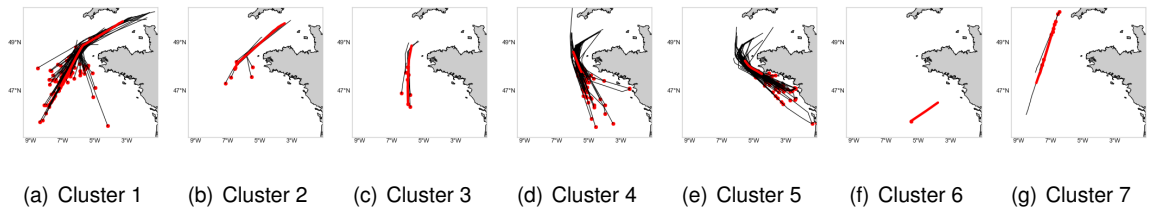
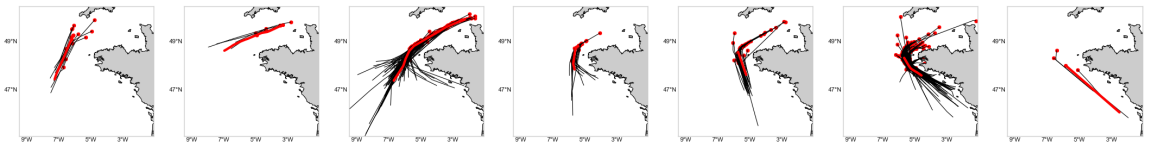


Figure 3.12: 14 out of 65 HAC clusters with distance_threshold = 70 and average linkage, for set 1.



(a) Cluster 1 (b) Cluster 2 (c) Cluster 3 (d) Cluster 4 (e) Cluster 5 (f) Cluster 6 (g) Cluster 7



(h) Cluster 8 (i) Cluster 9 (j) Cluster 10 (k) Cluster 11 (l) Cluster 12 (m) Cluster 13 (n) Cluster 14

The model selection for DBSCAN, shown in Figure 3.13, yielded 13, 20, and 22 as the ε for k as 3, 6, and 9, respectively. The lowest k had more clusters extracted, but grouped most of the data into

two big clusters (e.g., Figures 3.14(a) and 3.14(e)) that represent the overall downwards and upwards movements. While increasing k the trajectories were consecutively merged to these two big clusters. In general, the parameters obtained by the model selection gave poor results and showed that the neighborhood size was too big for the present data, which was aggravated with even higher values of ε for an increasing k. In fact, the existing clusters are not well separated and, as a natural response to density-connected points, this led in practice to undesirable consecutive merges of close clusters.

Figure 3.13: K-dist plot for set 1.

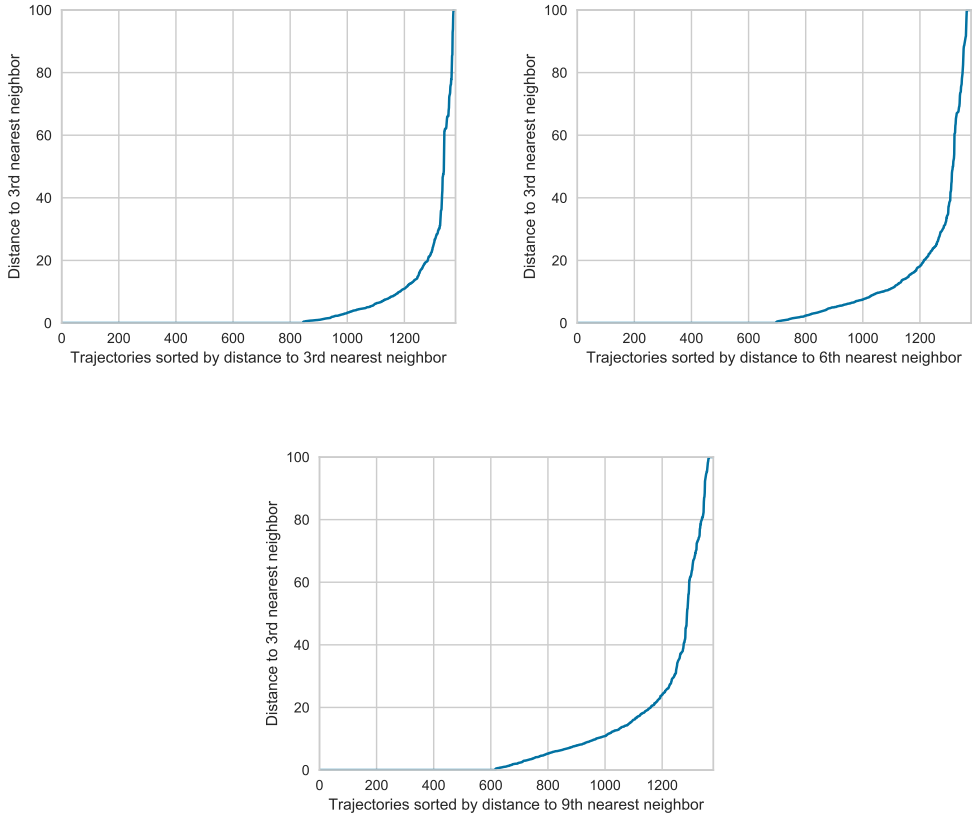
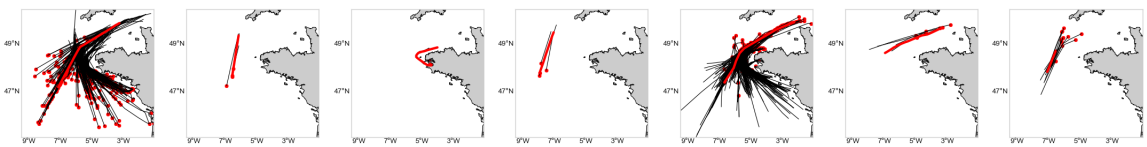
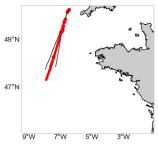


Figure 3.14: 8 DBSCAN clusters with ε = 13 and k = 3, for set 1.

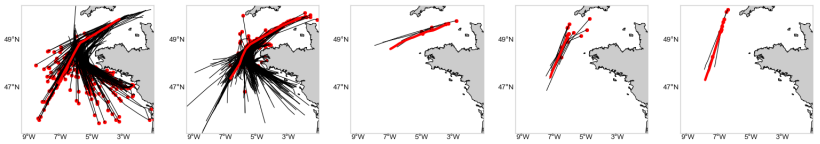


(a) Cluster 1 (b) Cluster 2 (c) Cluster 3 (d) Cluster 4 (e) Cluster 5 (f) Cluster 6 (g) Cluster 7



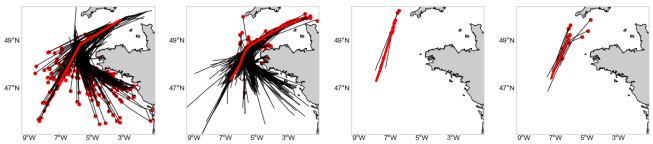
(h) Cluster 8

Figure 3.15: 5 DBSCAN clusters with ε = 20 and k = 6, for set 1.



(a) Cluster 1 (b) Cluster 2 (c) Cluster 3 (d) Cluster 4 (e) Cluster 5

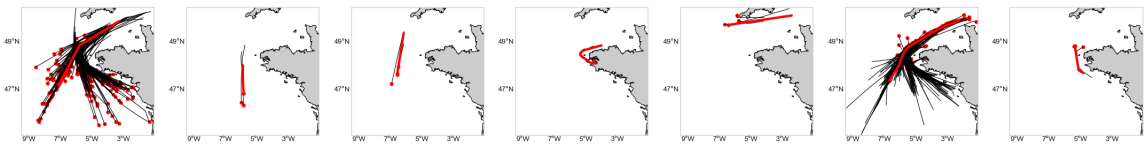
Figure 3.16: 4 DBSCAN clusters with ε = 22 and k = 9, for set 1.



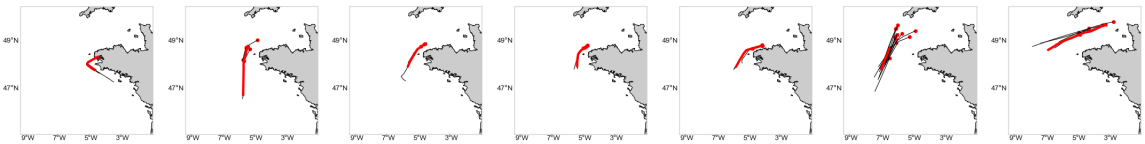
(a) Cluster 1 (b) Cluster 2 (c) Cluster 3 (d) Cluster 4

OPTICS also grouped a significant portion of the data into two big clusters distinguished by their directions, as seen in Figures 3.17(a) and 3.17(f), and , with increasing k, the data was again successively added to these clusters. Nevertheless, it could unravel more patterns than DBSCAN, which is already known for having problems when densities vary widely (the neighborhood size also meant excluding smaller clusters), whilst OPTICS experiments different densities by varying ε.

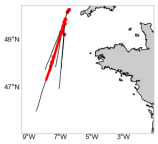
Figure 3.17: 15 OPTICS clusters with k = 3, for set 1.



(a) Cluster 1 (b) Cluster 2 (c) Cluster 3 (d) Cluster 4 (e) Cluster 5 (f) Cluster 6 (g) Cluster 7

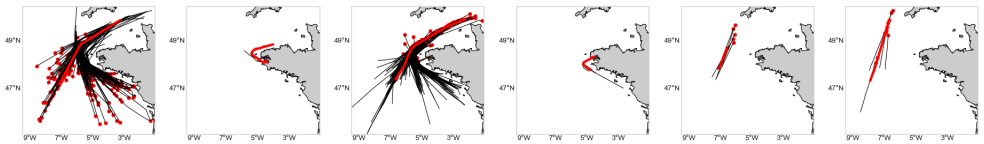


(h) Cluster 8 (i) Cluster 9 (j) Cluster 10 (k) Cluster 11 (l) Cluster 12 (m) Cluster 13 (n) Cluster 14



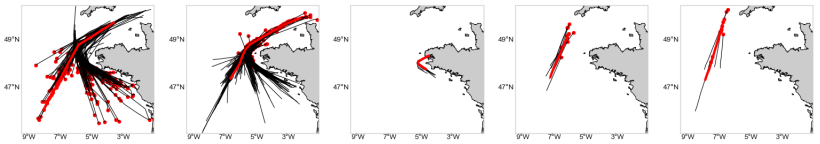
(o) Cluster 15

Figure 3.18: 6 OPTICS clusters with k = 6, for set 1.



(a) Cluster 1 (b) Cluster 2 (c) Cluster 3 (d) Cluster 4 (e) Cluster 5 (f) Cluster 6

Figure 3.19: 5 OPTICS clusters with k = 9, for set 1.



(a) Cluster 1 (b) Cluster 2 (c) Cluster 3 (d) Cluster 4 (e) Cluster 5

3.3.5.B Set 2

This set contained approximately 50% of the trajectories from set 1, and more useful clusters were obtained. The changes were as follows:

- As Table 3.3 revealed, the average link obtained the best CPCC outcome, whose corresponding dendrogram, displayed in Figure 3.20, exhibited a value of 70 for the distance threshold. Fewer clusters were deduced than from set 1, specifically 39 clusters (see Figure 3.22), from which 17 were single-trajectory clusters. The consequences of having misplaced trajectories are visible in Figure 3.22(c), where the baseline movement does not fit the trajectories. Regardless of this fact, the results improved in terms of allocating the noisy trajectories to better fitted clusters. For

example, the cluster in Figure 3.12(d) was divided into the hidden movements of clusters in Figures 3.22(d) and 3.22(e). Likewise, from the existing paths, it was possible to fulfill them with more information. For instance, the representative trajectory from the cluster in Figure 3.12(e) was better represented in the cluster of Figure 3.22(g).

- The model selection for DBSCAN in Figure 3.21 gave 22, 25, and 30 for ε with k as 3, 6, and 9. The neighborhood size was slightly increased than from the 1^{st} set, yielding overall less clusters and with worse quality. If this number was already high, in the present one revealed to cluster most data into two clusters sooner. Even though the trajectories had higher lengths than from set 1, and it could be expected that the bridging effect would be less, having fewer trajectories and an even bigger neighborhood worsen the results, being able to extract only the two big clusters from $k = 6$.

- OPTICS improved its results significantly from the ones in the previous set for any k, and the best results were obtained with $k = 3$, where the paths from the aforementioned two big clusters were unraveled (e.g., Figures 3.26(b), 3.26(c) and 3.26(n)) and more complete paths were provided (e.g., Figures 3.26(p) and 3.26(r)). There were misplacements in Figures 3.26(a) and 3.26(l), but not as significant as in set 1, indicating that some trajectories of those clusters shared similarities with others having differences with respect to the overall cluster and created thus a bridge to introduce ”noise”. This is one of the disadvantages of density-based approaches, which, in comparison with HAC, do not limit points that would distort the global cluster similarity. Despite the fact that this set provided better results, erasing small trajectories also meant erasing groups of routes that could be important, such as in Figure 3.17(b). This method outperformed all others.

Figure 3.20: Top 15 levels of the hierarchical clustering dendrogram with average link, for set 2.

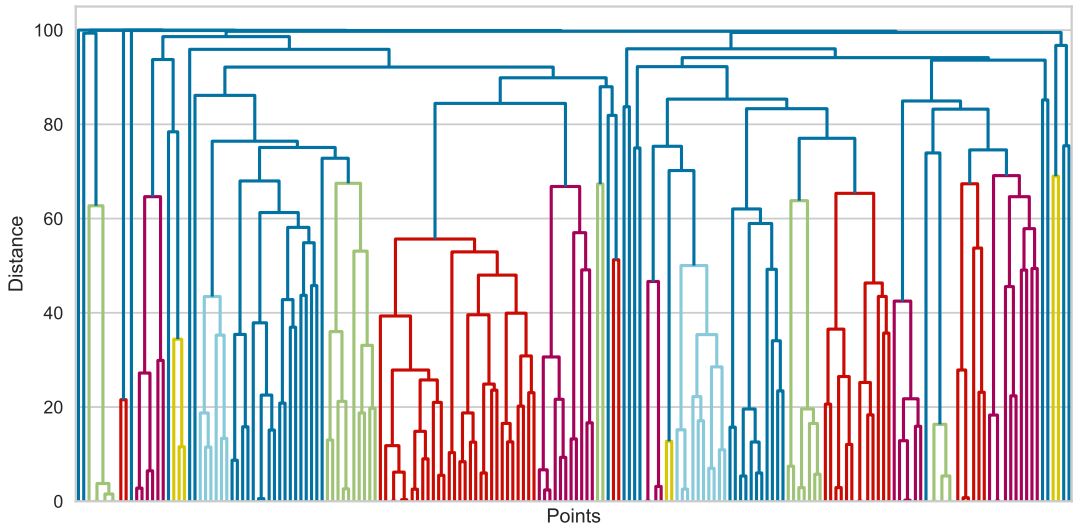
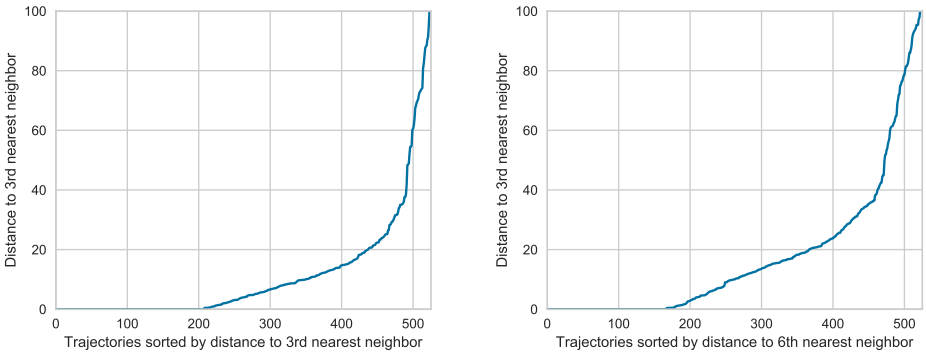


Table 3.3: CFCC values for set 2.

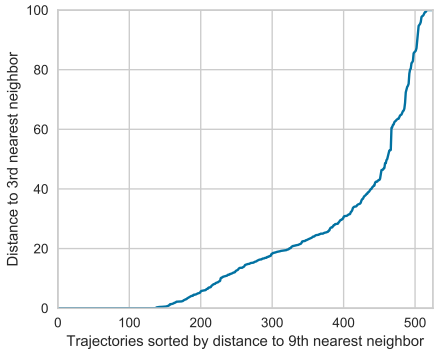
Link	CFCC
Single	0.6243815774286966
Complete	0.596917863849527
Average	0.8859866106125401

Figure 3.21: K-dist plot for set 2.



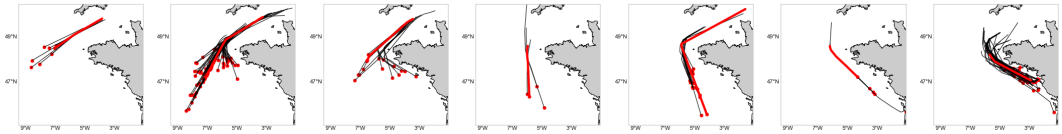
(a) 3-nearest neighbors.

(b) 6-nearest neighbors.



(c) 9-nearest neighbors.

Figure 3.22: 14 out of 39 HAC clusters with distance_threshold = 70 and average linkage, for set 2.



(a) Cluster 1

(b) Cluster 2

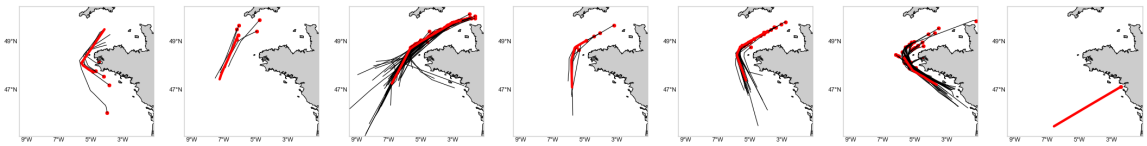
(c) Cluster 3

(d) Cluster 4

(e) Cluster 5

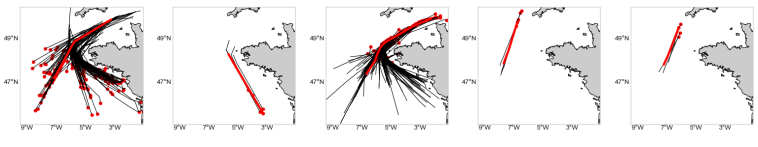
(f) Cluster 6

(g) Cluster 7



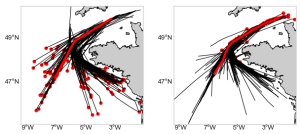
(h) Cluster 8 (i) Cluster 9 (j) Cluster 10 (k) Cluster 11 (l) Cluster 12 (m) Cluster 13 (n) Cluster 14

Figure 3.23: 5 DBSCAN clusters with ε=22 and k=3, for set 2.



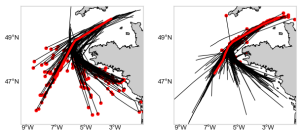
(a) Cluster 1 (b) Cluster 2 (c) Cluster 3 (d) Cluster 4 (e) Cluster 5

Figure 3.24: 2 DBSCAN clusters with ε=25 and k=6, for set 2.



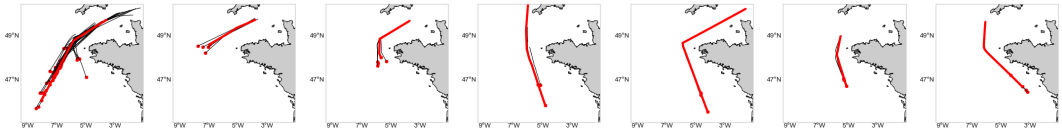
(a) Cluster 1 (b) Cluster 2

Figure 3.25: 2 DBSCAN clusters with ε=30 and k=9, for set 2.

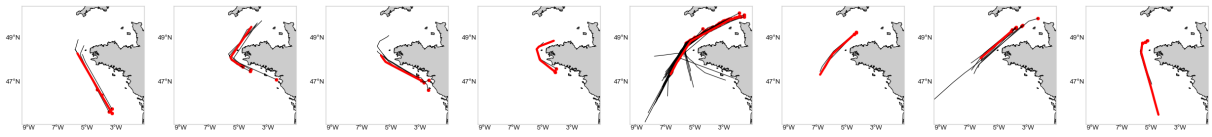


(a) Cluster 1 (b) Cluster 2

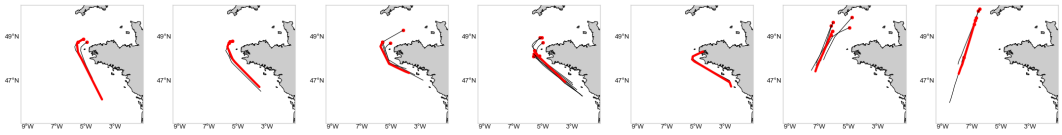
Figure 3.26: 22 OPTICS cluster with k = 3, for set 2.



(a) Cluster 1 (b) Cluster 2 (c) Cluster 3 (d) Cluster 4 (e) Cluster 5 (f) Cluster 6 (g) Cluster 7

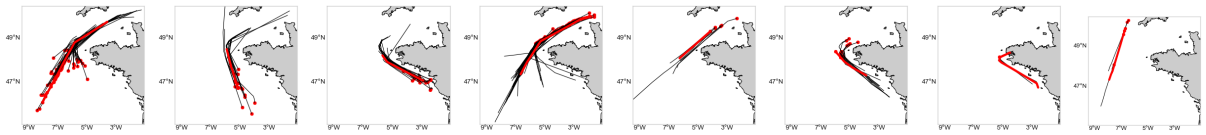


(h) Cluster 8 (i) Cluster 9 (j) Cluster 10 (k) Cluster 11 (l) Cluster 12 (m) Cluster 13 (n) Cluster 14 (o) Cluster 15



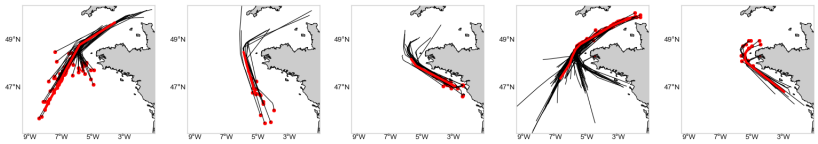
(p) Cluster 16 (q) Cluster 17 (r) Cluster 18 (s) Cluster 19 (t) Cluster 20 (u) Cluster 21 (v) Cluster 22

Figure 3.27: 8 OPTICS cluster with k = 6, for set 2.



(a) Cluster 1 (b) Cluster 2 (c) Cluster 3 (d) Cluster 4 (e) Cluster 5 (f) Cluster 6 (g) Cluster 7 (h) Cluster 8

Figure 3.28: 5 OPTICS cluster with k = 9, for set 2.



(a) Cluster 1 (b) Cluster 2 (c) Cluster 3 (d) Cluster 4 (e) Cluster 5

3.3.5.C Set 3

Containing fewer but longer trajectories, the remarks for the last set are enumerated:

- Again, agglomerative clustering was composed mainly of single-trajectory clusters (50%). The number of clusters was reduced, but on the other hand, it was possible to get more information regarding some of the paths (e.g., Figures 3.31(c) and 3.31(e)).

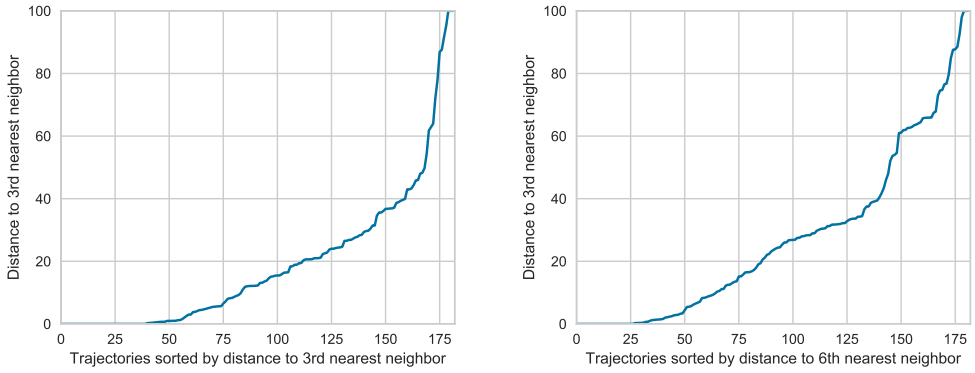
- In Figure 3.29, DBSCAN's ε was 30, 34, and 42 for the pre-defined k range. These values yielded similar results from those of the 2^{nd} set and proved that not even longer trajectories could get better results.

- Overall, OPTICS collected fewer clusters than from the previous set, and for $k > 6$ only less than three clusters were extracted. Yet, 15 clusters were achieved with $k = 3$, which offered a good comprise between the size of the data sample and the number of clusters. The compactness of those clusters was enhanced than from set 2 because we dealt with more complete trajectories. As in the 2^{nd} set, it was the outperforming algorithm.

Table 3.4: CFCC values for set 3.

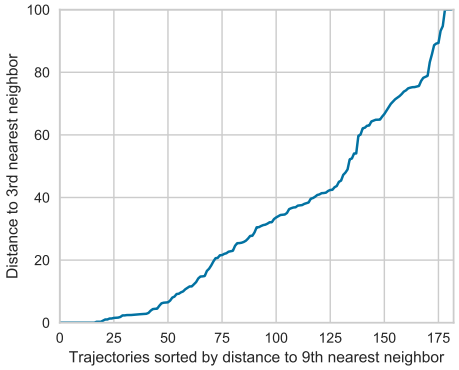
Link	CFCC
Single	0.739636152870022
Complete	0.7072190004481498
Average	0.9202937660976129

Figure 3.29: K-dist plot for set 3.



(a) 3-nearest neighbors.

(b) 6-nearest neighbors.



(c) 9-nearest neighbors.

Figure 3.30: Top 15 levels of the hierarchical clustering dendrogram with average link, for set 3.

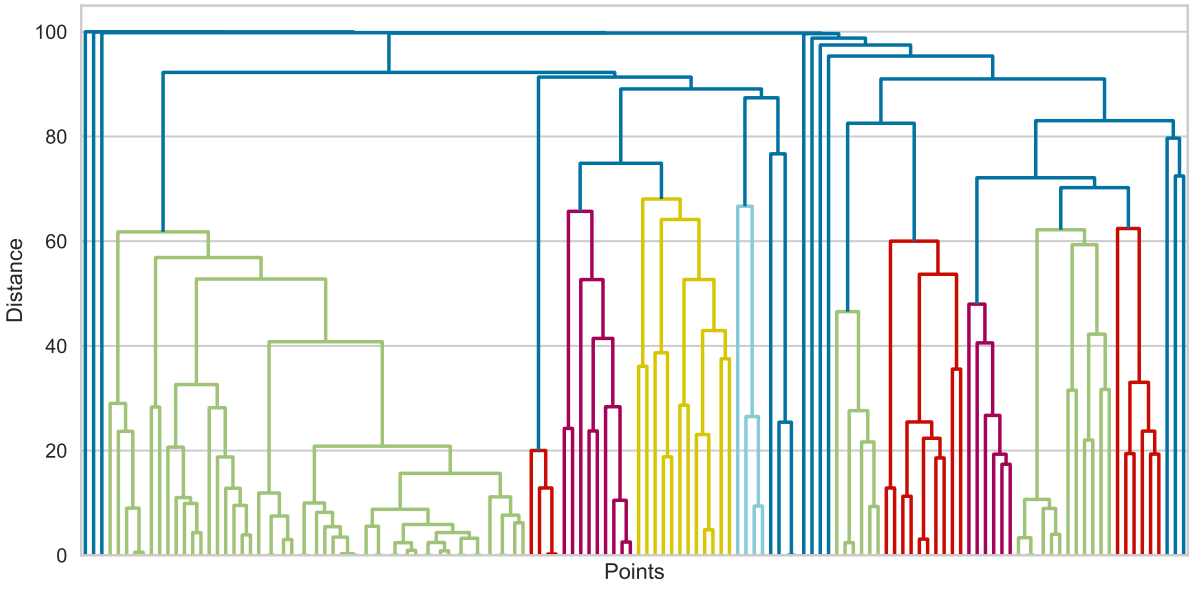
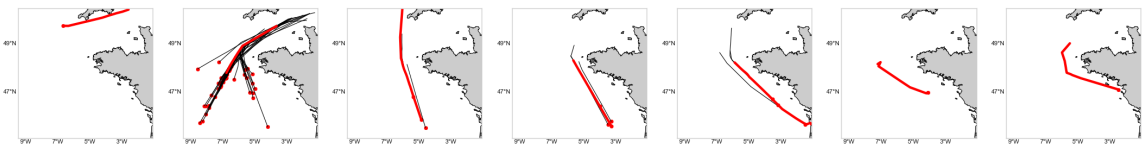
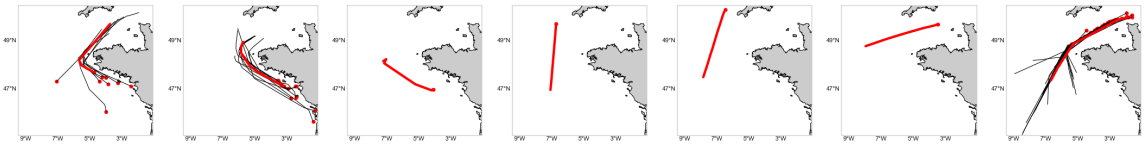


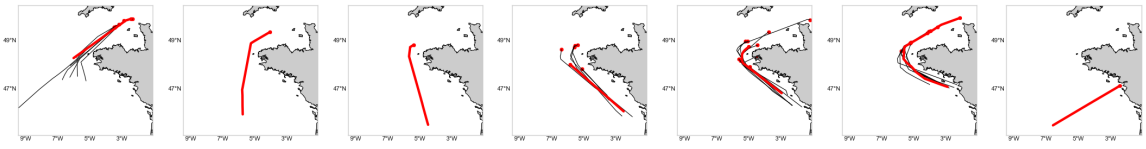
Figure 3.31: 22 HAC clusters with distance_threshold = 70 and average linkage, for set 3.



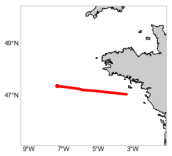
(a) Cluster 1 (b) Cluster 2 (c) Cluster 3 (d) Cluster 4 (e) Cluster 5 (f) Cluster 6 (g) Cluster 7



(h) Cluster 8 (i) Cluster 9 (j) Cluster 10 (k) Cluster 11 (l) Cluster 12 (m) Cluster 13 (n) Cluster 14

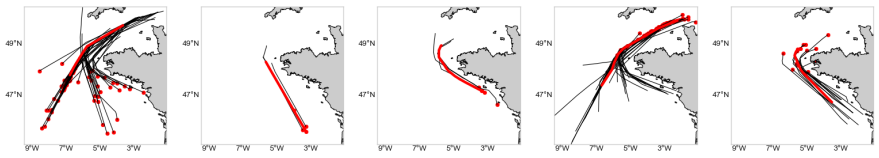


(o) Cluster 15 (p) Cluster 16 (q) Cluster 17 (r) Cluster 18 (s) Cluster 19 (t) Cluster 20 (u) Cluster 21



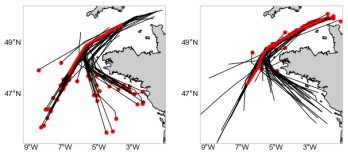
(v) Cluster 22

Figure 3.32: 5 DBSCAN clusters with ε = 30 and k = 3, for set 3.



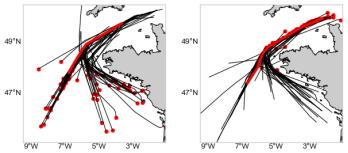
(a) Cluster 1 (b) Cluster 2 (c) Cluster 3 (d) Cluster 4 (e) Cluster 5

Figure 3.33: 2 DBSCAN clusters with ε = 34 and k = 6, for set 3.



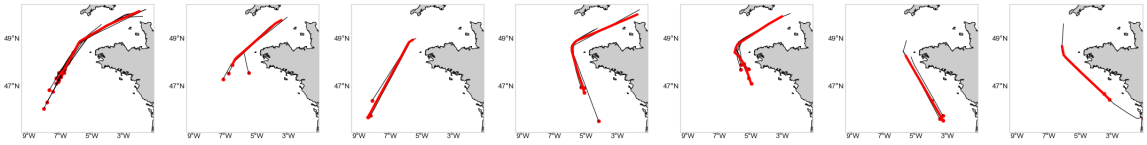
(a) Cluster 1 (b) Cluster 2

Figure 3.34: 2 DBSCAN clusters with ε = 42 and k = 9, for set 3.

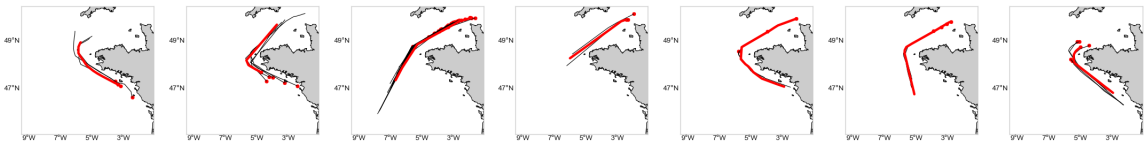


(a) Cluster 1 (b) Cluster 2

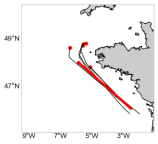
Figure 3.35: 15 OPTICS clusters with k = 3, for set 3.



(a) Cluster 1 (b) Cluster 2 (c) Cluster 3 (d) Cluster 4 (e) Cluster 5 (f) Cluster 6 (g) Cluster 7

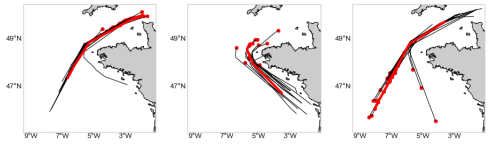


(h) Cluster 8 (i) Cluster 9 (j) Cluster 10 (k) Cluster 11 (l) Cluster 12 (m) Cluster 13 (n) Cluster 14



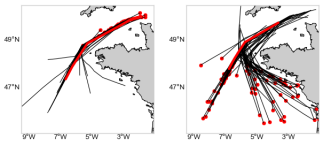
(o) Cluster 15

Figure 3.36: 3 OPTICS clusters with k = 6, for set 3.



(a) Cluster 1 (b) Cluster 2 (c) Cluster 3

Figure 3.37: 2 OPTICS clusters with k = 9, for set 3.



(a) Cluster 1 (b) Cluster 2

3.3.6 Re-Classification

A re-classification process was made to evaluate the quality of the assignment of trajectories with respect to the main routes. By computing the distance between each trajectory and the representative paths of the clusters, the routes in the training set were allocated to the representative route with the lowest distance.

A natural choice for measuring distances is the LCSS joint measure, which was used so far. In reality, if a trajectory belongs to a cluster, it does not mean that it meets the distance threshold set in LCSS, due to density-connected points. So, if some trajectory belonging to a given cluster is not inside its acceptable region, the output value would be 100 for that cluster. Thereby, this method revealed

itself inadequate for measuring the distance between trajectories and main routes. As an alternative, we chose to work with DTW joint distance, described in section 3.3.1, whenever the LCSS joint measure failed. Unlike LCSS, DTW is able to produce a hierarchy of distances.

DBSCAN was excluded from this evaluation because it found no reasonable clusters for any of the data samples. OPTICS clusters from set 1 were also excluded due to the same reasons, as well as its remaining clusters from other sets with $k > 3$. From the re-classification results, the following remarks were made:

- For set 1, HAC had correct re-classifications for clusters in Figures 3.38(a), 3.38(b), 3.38(e) and 3.38(h). In particular, the trajectories from cluster 1 that were slightly straight were precisely placed in cluster 2. However, having many short representative routes led to a series of wrong assignments, as seen in the cluster of Figure 3.38(d), whose destinations differed. The distance measure also did not provide good estimates in Figure 3.38(c), given that there were better options such as the cluster in Figure 3.38(d).

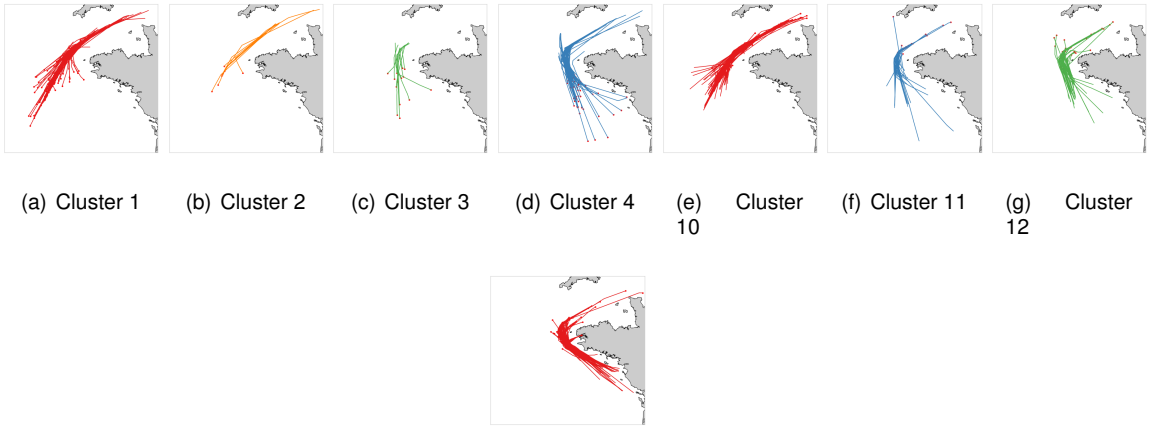
- In Figure 3.39, the results for HAC in the 2^{nd} set show that clusters in Figures 3.39(b), 3.39(c) and 3.39(g), which had many misplaced trajectories, are now significantly more compact: trajectories from cluster 2 were correctly distributed to cluster 1 and 5; those from cluster 3 were put in cluster 7; from cluster 10, they were assigned to cluster 13. The same scenario was not confirmed in Figure 3.39(f), and the noisy trajectories remained. Given that there was a better placement for them, specifically in cluster 5, the conclusion is that the measure could not provide the best results. The measure also provided wrong assignments in Figure 3.39(h).

- For OPTICS, with the 2^{nd} set, it can be seen in Figure 3.40 that the measurements were made successfully. Some of the trajectories of cluster 1 and 12 were positioned in cluster 3 and cluster 14, respectively, where they were more fitted. Still, cluster 12 continued with trajectories coming from a different origin, because although cluster 5 had the same origin it ended sooner, and there was a greater extension where it matched with cluster 12. The noisy route in cluster 14 was placed in the same way.

- For the 3^{rd} set, HAC got overall correct assignments. Nevertheless, again, it is shown that more clusters should have been eventually extracted, as seen in Figure 3.41(c), where the best fitted location is a partially similar path.

- OPTICS, for set 3, already had trajectories assigned to fitted clusters, but the re-classification showed that there were better placements, specifically in terms of starting at the same point (changes from the cluster in Figure 3.42(a) to the one in Figure 3.42(b)), and having the same length (changes from the cluster in Figure 3.42(c) to the one in Figure 3.42(d)).

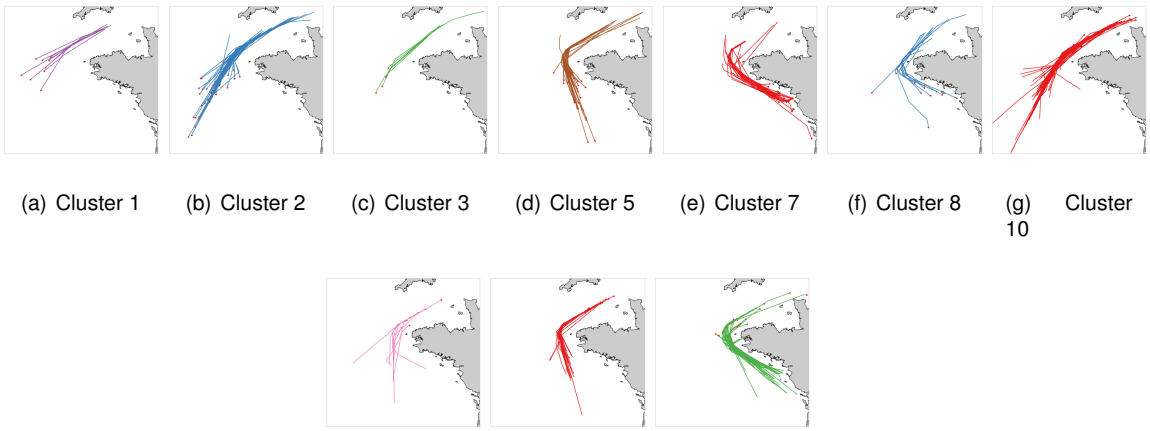
Figure 3.38: Re-classification changes subset of HAC clusters, for set 1.



(a) Cluster 1 (b) Cluster 2 (c) Cluster 3 (d) Cluster 4 (e) Cluster 10 (f) Cluster 11 (g) Cluster 12

(h) Cluster 13

Figure 3.39: Re-classification changes subset of HAC clusters, for set 2.



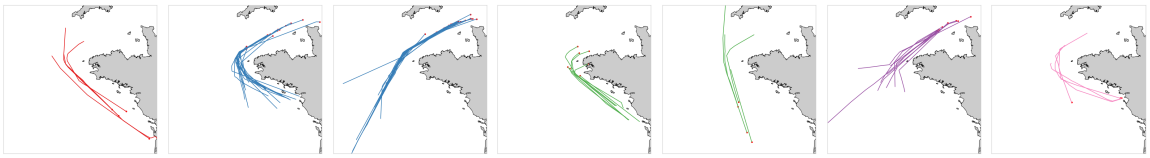
(a) Cluster 1 (b) Cluster 2 (c) Cluster 3 (d) Cluster 5 (e) Cluster 7 (f) Cluster 8 (g) Cluster 10

(h) Cluster 11 (i) Cluster 12 (j) Cluster 13

Figure 3.40: Re-classification changes of OPTICS clusters with $k = 3$, for set 2.

(a) Cluster 1 (b) Cluster 2 (c) Cluster 3 (d) Cluster 12 (e) Cluster 13 (f) Cluster 14 (g) Cluster 18

Figure 3.41: Re-classification changes subset of HAC clusters, for set 3.



(a) Cluster 1 (b) Cluster 2 (c) Cluster 10 (d) Cluster 11 (e) Cluster 12 (f) Cluster 14 (g) Cluster 14

Figure 3.42: Re-classification changes of OPTICS clusters with $k = 3$, for set 3.

(a) Cluster 1 (b) Cluster 2 (c) Cluster 10 (d) Cluster 11 (e) Cluster 12 (f) Cluster 14

4

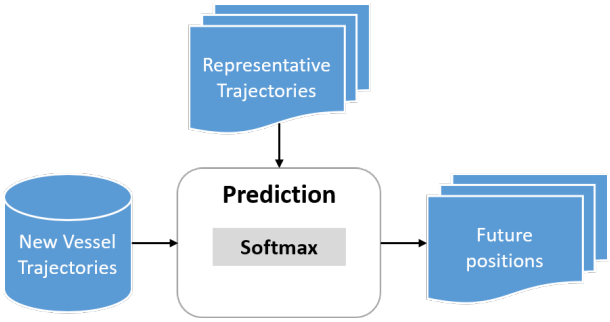
Prediction

The route prediction was approached through a probabilistic measure over the nearest routes' positions. In short, the estimation for a given Δt is the weighted sum of the positions with respect to t of the three most probable paths. The softmax function was used to acquire the weights distribution for each route. This procedure is detailed in section 4.1.1. The steps underlying this module are:

1. Identify the set of the three most similar representative routes of incoming vessel trajectories, which were obtained in chapter 3.

2. Predict future positions of a given test trajectory according to the movement performed by the identified set.

Figure 4.1: Route prediction approach.



4.1 Algorithm Description

For an incoming trajectory, it is necessary to judge whether there is available information regarding the movement being held up until the last moment. For this, a radius is centered in the observation's end point to trace the area where the nearest routes should be. Evidently, if no routes are inside that radius, then the route is considered as an "outlier", at least up until that point. Likewise, if none of the routes found have a similar direction, which is measured with LCSS coupled with the cosine distance, for a threshold of 30. The same distance used in LCSS joint distance, to define the acceptable area, was set to represent the radius (19 km).

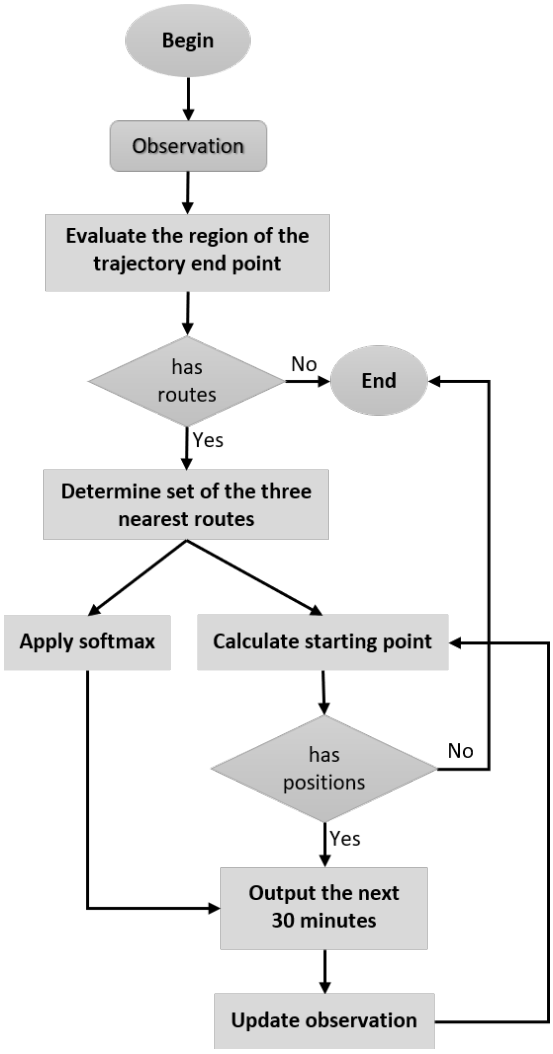
Having intersecting routes, the hierarchy of the three closest routes is calculated to feed the softmax function accordingly, whose details are presented in section 4.1.1. As mentioned in section 3.3.6, the LCSS joint distance fails in computing this hierarchy, so the same approach was taken. Finally, the nearest points between the current observations' end point and the closest routes are calculated, aiming to define the location from where the prediction will start in each route.

The movement obtained from each route must be made with an approximate speed to the incoming trajectory. So, the same approach used for resampling, in section 3.2, was employed to interpolate the

data points at the same speed. To accomplish this, it is calculated the average speed of the incoming trajectory for the last 6 points (equivalent to 30 minutes), assuming that it provides a good estimate of future speed. With both the speed and the five-minute interval between each data point, it is extracted the average distance sailed per point, which is used as the new resampling interval. Here, the independent variable was the accumulative sailed distance.

For a given prediction time, the collected positions are averaged according to the respective weights from softmax, and the predicted position is thus returned.

Figure 4.2: Iterative route prediction.



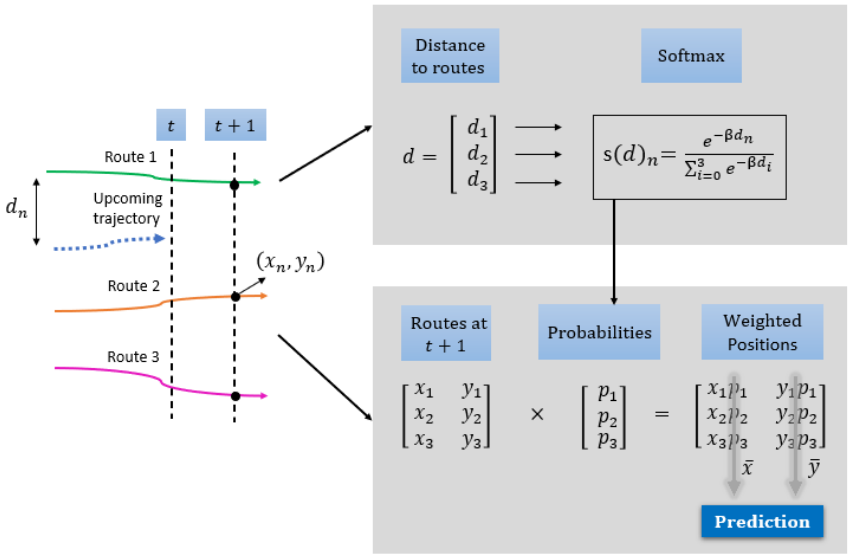
4.1.1 Softmax

Softmax is a commonly used function in the ML field as the output of a classifier, to express the probability distribution of a discrete variable with n different values (classes) [55]. In the present work, it

was chosen due to its simplicity and usefulness to convert the vector of distance scores to a vector of probabilities, that serves as the mixture of weights for the path prediction.

The function takes a set of n real numbers and transforms it into n probabilities proportional to the exponentials of the input numbers. It creates a sort of competition between the numbers: the outputs always sum to 1, so an increase in the value of one unit necessarily corresponds to a decrease in the value of the others [55]. This behavior is regulated with the β parameter determining how the probabilities converge according to the input distances. That is, higher β values mean the exponential of higher values grows quicker and a tighter adjustment to these values is seen, whereas lower values produce probability distributions consistent with the input values. Since we are working with distance inputs and lower values should get a higher probability because the route is closest, negative beta values are used instead.

Figure 4.3: Route prediction with softmax for $t + 1$.



4.2 Results

The clusters from section 3.3.5 that did not render reasonable clusters were again removed from the evaluation. Thus, the prediction was performed with the following settings: for the 1^{st}, only HAC was used; for the 2^{nd} and 3^{rd} set, both HAC and OPTICS (with $k = 3$) were used. The remaining 30% of each data sample was used as the testing set.

Each trip was divided into two sets in order to simulate a moving vessel: one representing the current observation and the other serving as the ground truth of the future movement, to which the system's estimate will be compared. This split was done according to 25%, 50% and 75% of the beginning of the trajectory, and it corresponds to the current observation.

The average lateral deviation per km is used to evaluate the prediction accuracy. Let L be the prediction distance in km and N the number of predicted points, for the set of points p_i and its estimates p_i' the error is defined as follows:

$$error = \frac{1}{L}\sum_{i=1}^{N} haversine(p_i^{'}, p_i) \tag{4.1}$$

The haversine equation was already defined in eq. (3.6).

To have a deeper understanding of what situations the system is able to achieve better results, the following evaluations were assessed:

- A parametric test to evaluate the effect of the β parameter, in softmax's transformation, on the resulting prediction. This allows to select the best setting for β, for which the remaining considerations were based.
- For a fixed observation size, test different prediction temporal window sizes.

4.2.1 Set 1

The evaluation for HAC clusters is exhibited in Figure 4.4 and Table 4.1, and the illustration of the prediction for an incoming vessel in Figure 4.5. The analysis exposed the following:

- Looking at Table 4.1, it is possible to observe that the number of outliers increases directly with the amount of observation that is being used. In practice, this means that while we get more positional information, the main routes reach a point where they cannot offer more predictions. Looking at the clusters in Figure 3.12, most of them where regarding one trajectory only, and the underlying baseline movement was small. So, the area is not well represented. It appears that this resulted in a big number of outliers, reaching more than a half with a 75% of observation.
- Between the observation sizes, there was no overall agreement on what is the best setting for β, in order to achieve lower average deviation per trip. Either way, the behavior of β between 5 and 11 was most of the times identical.
- Overall, the predictions' average deviations were not stable, e.g., on a 25% observation, for predictions up to 15 hours, the errors took an unexpected fall. Looking back at the trajectories duration, in Figure 3.2(d), we conclude that more than a half has less than 15 hours, so, from this point, the error might be averaged with fewer trajectories and the results can suffer great variances. On the other hand, it also shows that for trajectories on that range of prediction, the average deviation balanced itself with the distance sailed so far.

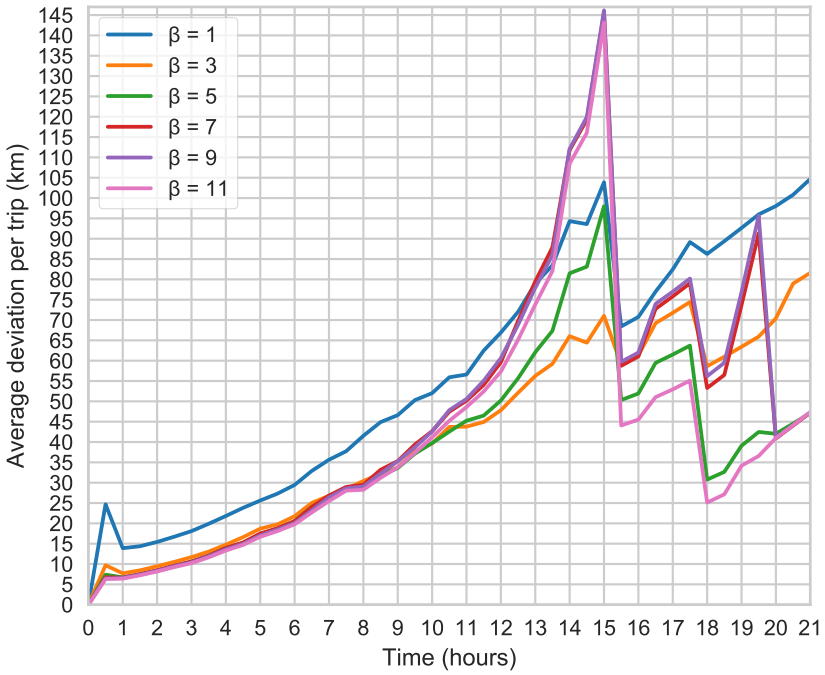
- In section 3.3.1, it was established that the width of a maritime corridor had 19 km, which means that if a vessel has approximately 10 km of lateral deviations, then it would still belong to that corridor. Bearing this in mind, for 25% of observation this was achieved up until a 3-hour prediction on a total of 21 hours; for 50% of observation, also below 3 hours; and, for 75%, for only 30 minutes.

- It would be expected that with increasing observations sizes the overall deviations per trip decreased, because there was more certainty about the route being pursued. This did not happen, showing that the global prediction performance with HAC clusters was low.

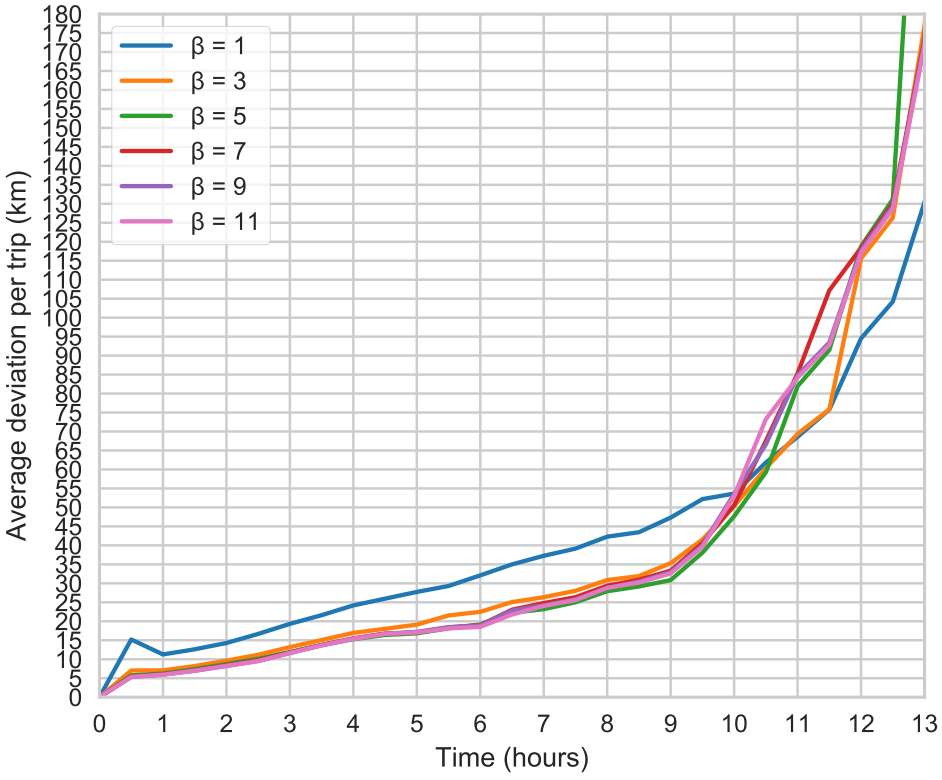
Table 4.1: Outliers for HAC clusters, for set 1.

Observation	Outliers
25%	36%
50%	41%
75%	52%

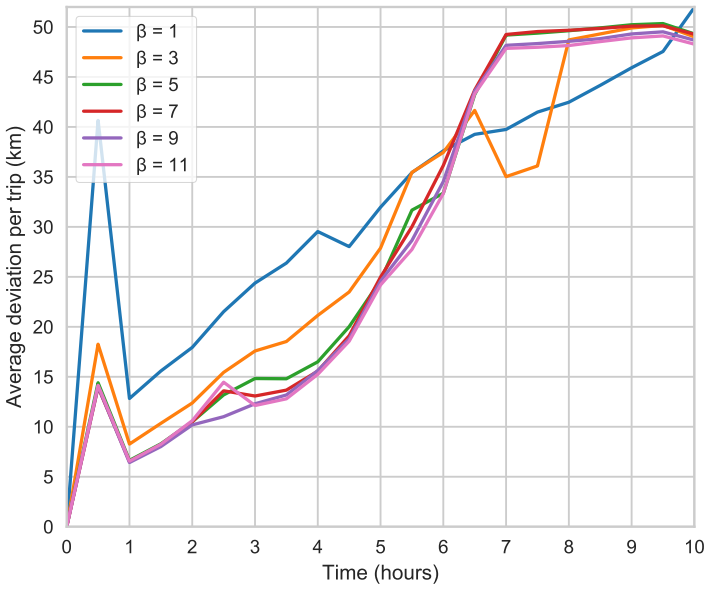
Figure 4.4: Comparative analysis of different beta values for each observation size – HAC clusters, on set 1.



(a) 25% of observation.

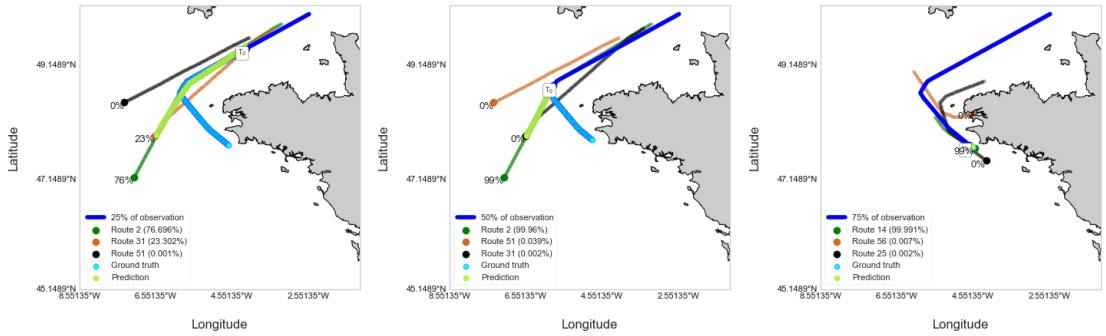


(b) 50% of observation.



(c) 75% of observation.

Figure 4.5: Example of movement prediction with HAC clusters, for set 1.



(a) 25% of observation – average deviation of 35 km.

(b) 50% of observation – average deviation of 50 km.

(c) 75% of observation – average deviation of 0.85 km.

4.2.2 Set 2

The prediction evaluation made with the representative routes from HAC clusters is demonstrated in Figure 4.6 and table 4.2, and a practical example is shown in Figure 4.7. It was possible to verify that:

- For 75%, half of the data did not find information to perform the prediction. This suggests that the main routes had insufficient information.
- The higher the β the lower the average deviations, The worst results are seen with $\beta = 1$ and the best results for $\beta = 11$. Higher β's means that the prediction closely follows the information of the closest route, showing that adding more routes to the equation leads to worse results.
- For every observation size, a large error reduction is seen from $\beta = 1$ to $\beta = 3$, for the whole prediction. These differences start to stabilize from $\beta = 5$.
- Through all observations, the behavior of the the average deviation error with time is highly irregular, and an unexpected decrease is seen for later predictions. This set still contains small/medium-sized trajectories, so the same reason for the previous set could apply. Yet, this demonstrates that the deviation could keep up with an increasing distance for those trajectories.

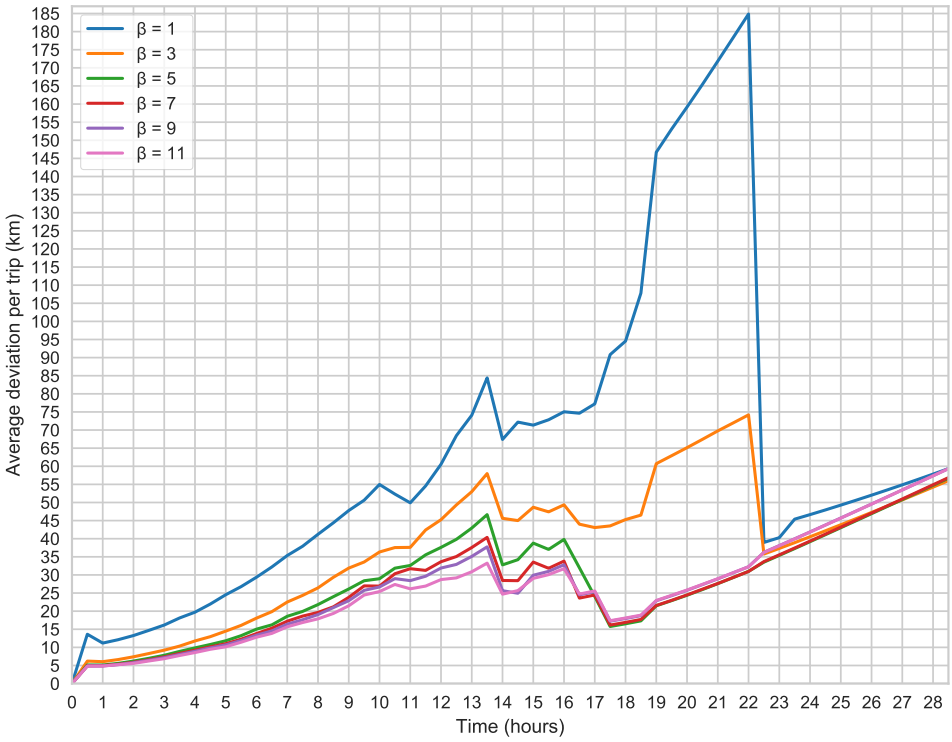
With β=11,

- According to the prediction range that each observation size can offer, it was verified that increasing the observation size could keep low average deviations longer: with 25%, the values were less than 10 km for the prediction of the following 5 hours in a total of 28,5 hours; with observations of 50%, the 10-km threshold is attained until the first 4 hours on an approximate 12-hour window; the last set of observations got reasonable values of 10 km half of the prediction window. This means that more information is required to obtain good performance.

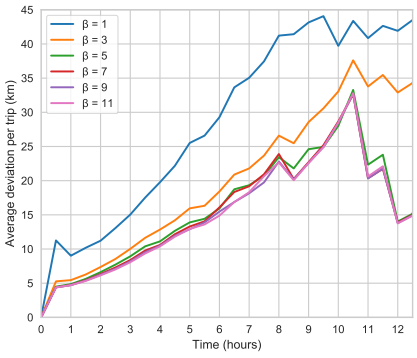
Table 4.2: Outliers for HAC clusters, for set 2.

Observation	Outliers
25%	22%
50%	34%
75%	50%

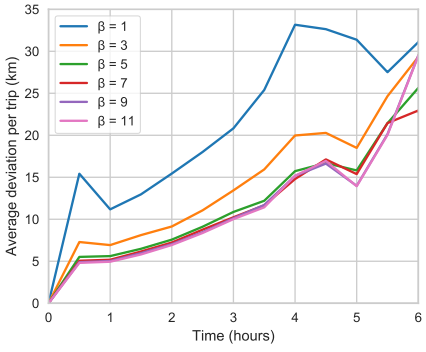
Figure 4.6: Comparative analysis of different beta values for each observation size – HAC clusters, on set 2.



(a) 25% of observation.

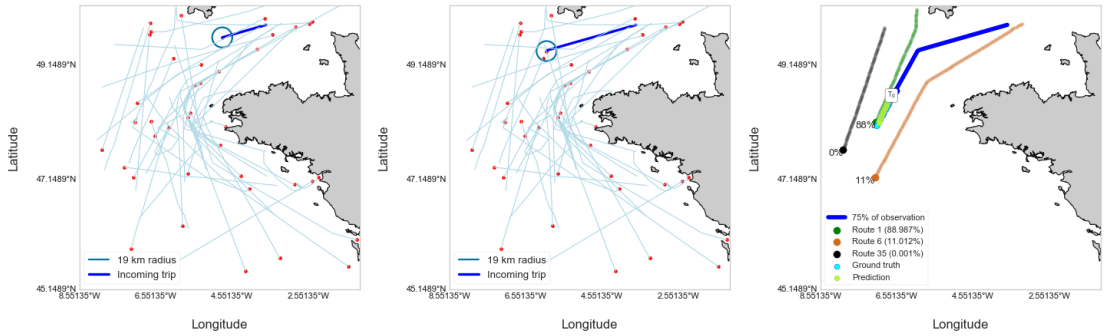


(b) 50% of observation.



(c) 75% of observation

Figure 4.7: Example of movement prediction with HAC clusters, for set 2.



(a) 25% of observation – no acceptable routes in a 19 km radius.

(b) 50% of observation – no acceptable routes in a 19 km radius.

(c) 75% of observation – average deviation of 3 km.

In relation to OPTICS main routes, the results are depicted in Figure 4.8 and Table 4.3, an in Figure 4.9, a prediction sample. The observations are:

- Once more, the number of outliers increased for bigger sizes of observation, but less than HAC. Effectively, the prediction window was extended up to 19 hours, while HAC provided 12 hours, for observations of 50%.

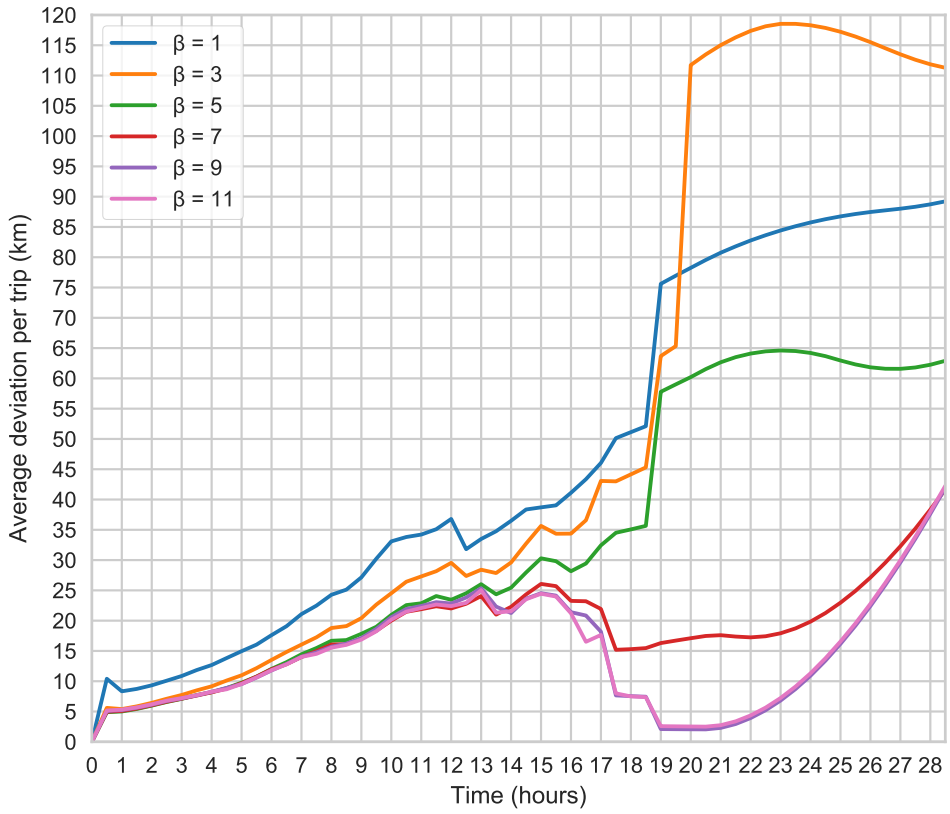
- The outperforming β did not reach a general agreement: for a 25% observation, β=1 is the best setting, and β=2 the worst; for a 50% observation, β=11 is the best setting and β=1 the worst; finally, for the last observation size, β=3 reveals to have the best performance, and overall, β=11, the worst.

- In comparison with HAC, it had, in general, a better performance for observations of 25%, and higher average deviations for a 75% size. Even tough it provided a bigger time window with observations of 50%, the additional hours had high average deviations per trip.

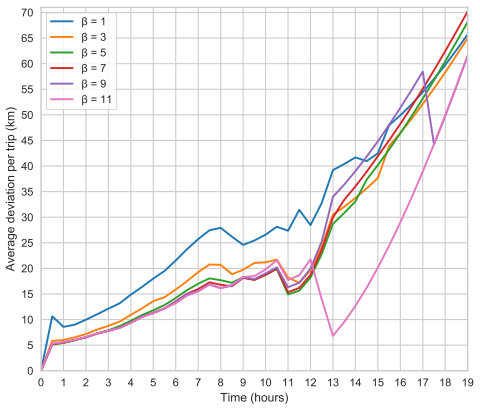
Table 4.3: Outliers for OPTICS clusters, for set 2.

Observation	Outliers
25%	25%
50%	32%
75%	46%

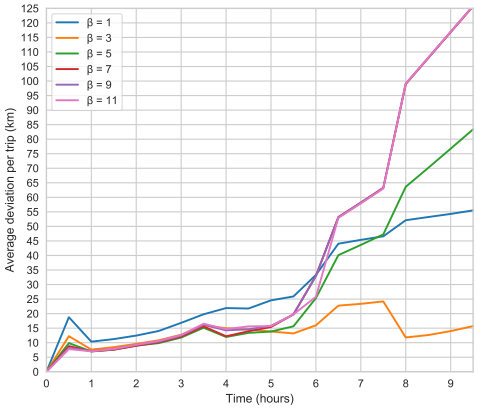
Figure 4.8: Comparative analysis of different beta values for each observation size – OPTICS clusters, on set 2.



(a) 25% of observation.

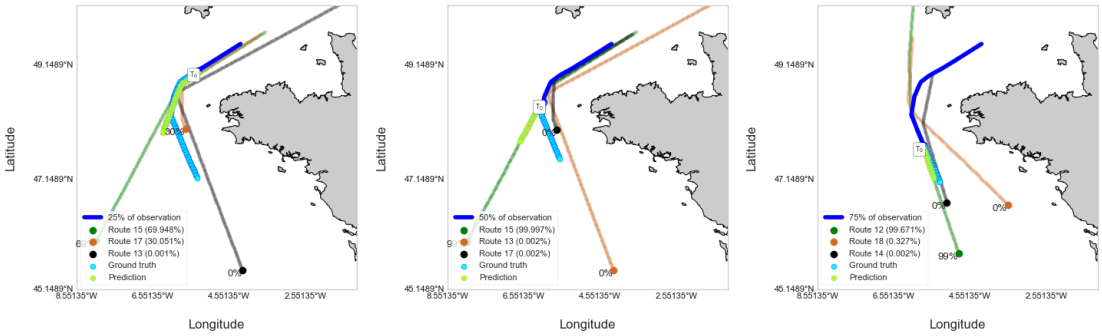


(b) 50% of observation.



(c) 75% of observation.

Figure 4.9: Example of movement prediction with OPTICS clusters, for set 2.



(a) 25% of observation – average deviation of 13 km.

(b) 50% of observation – average deviation of 24 km.

(c) 75% of observation – average deviation of 4 km.

4.2.3 Set 3

Concerning HAC clusters, the results are found in Figure 4.10 and Table 4.4, and an example in Figure 4.11. The changes are as follows:

- In this set, the number of the outliers increased with each observation size. For a 75% observation, close to half of the trajectories could not be predicted, meaning that the main routes were either insufficient or small.

- For any size of the broadcasting information, the worst results belonged to β=1 and the best results to β=11. The differences on the average deviations start to minimize in β=3.

As regards to β=11,

- When only 25% of the trajectories was ”broadcasted”, the prediction could provide reasonable results for 6 hours in a total of 28,5 hours.

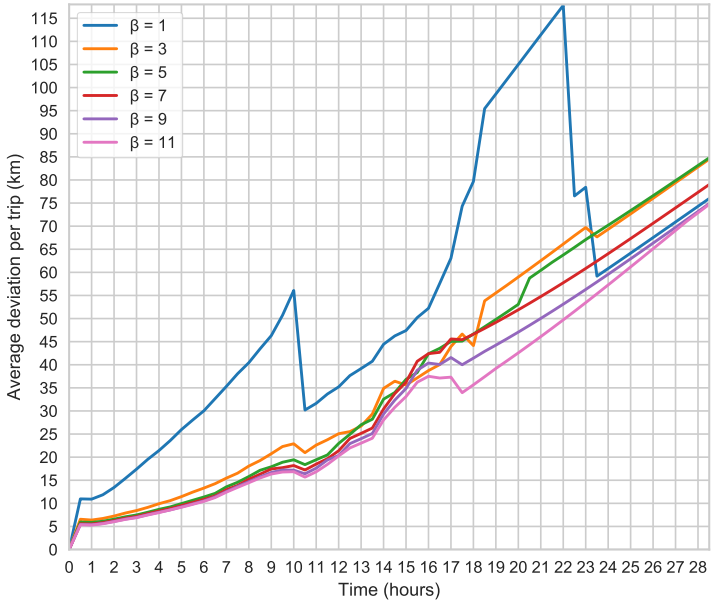
- Having half of trips' information meant reducing the overall average deviation to a lower range, but it could not maintain errors lower than 10 km for more than 3 hours out of 12,5 hours.

- The best results were achieved when we had 75% of the trajectories' information, keeping low average deviations for most of the prediction.

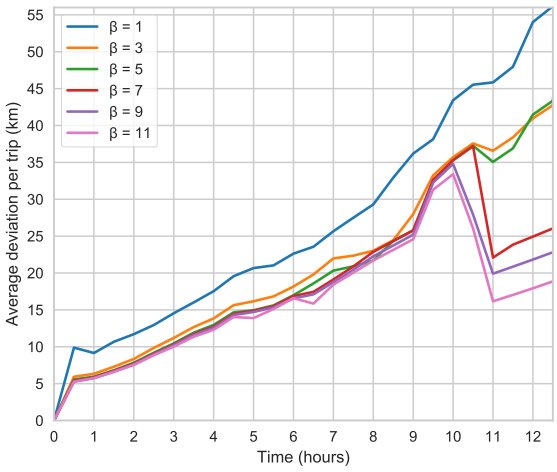
Table 4.4: Outliers for HAC clusters, for set 3.

Observation	Outliers
25%	23%
50%	37%
75%	46%

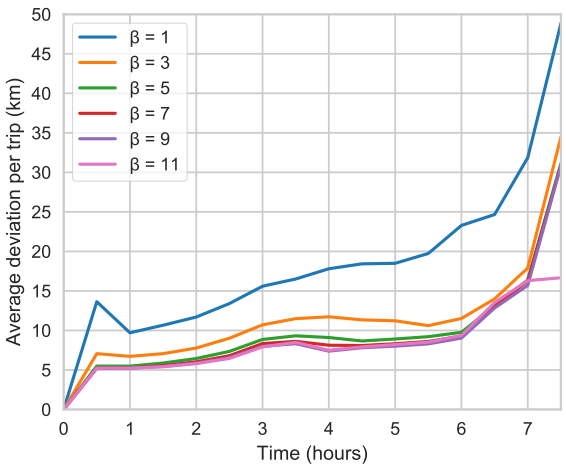
Figure 4.10: Comparative analysis of different beta values for each observation size – HAC clusters, on set 3.



(a) 25% of observation – average deviation of km.

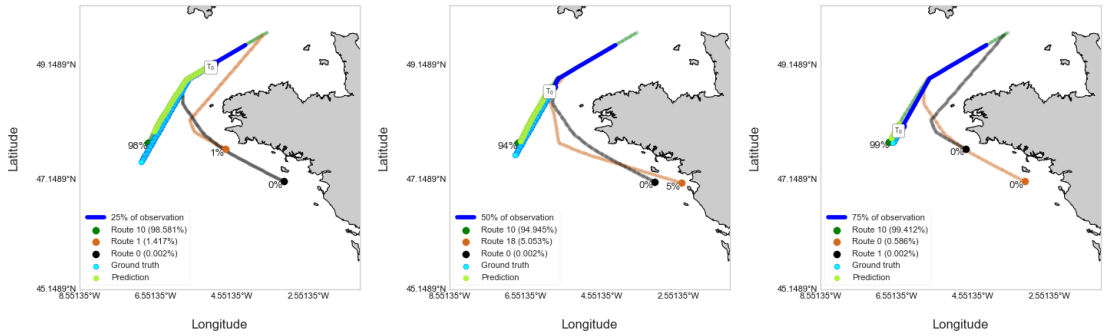


(b) 50% of observation.



(c) 75% of observation.

Figure 4.11: Example of movement prediction with HAC clusters, for set 3.



(a) 25% of observation – average deviation of 19 km.

(b) 50% of observation – average deviation of 9 km.

(c) 75% of observation – average deviation of 5 km.

As to OPTICS clusters, the results are shown in Figure 4.12 and Table 4.5, and a prediction example in Figure 4.13. Some of the remarks are:

- Once more, the number of outliers increased with the size of observation, but were lower than HAC, in the present set, proving that the routes had more information. It is possible to verify that OPTICS could get more 3 hours than HAC.
- For any of the observation sizes, β=1 had the worst average deviation and β=11 was the outperforming value. An exception was found between 13 and 14 hours, in a 50% observation, showing eventually the turning point where the deviation error could keep up with the sailed distance. it starts to diminish.

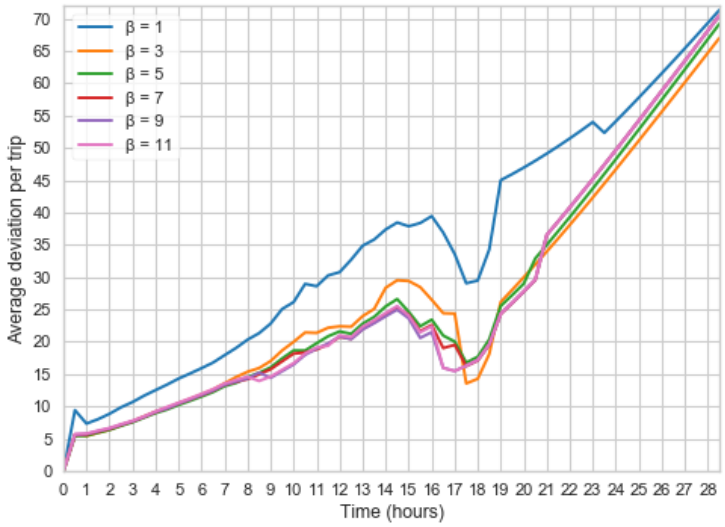
With respect to β=11,

- For OPTICS's representative routes, observations of 25% did not obtain a good performance, but offered overall lower deviations errors than HAC.
- With observations of 50%, the average deviations remained lower for longer than the previous observation size, and it also provided a bigger prediction window than HAC due to better represented main routes.
- As expected, a 75% observation got lower average deviations per trip for the prediction totality. The performance was equivalent to HAC.

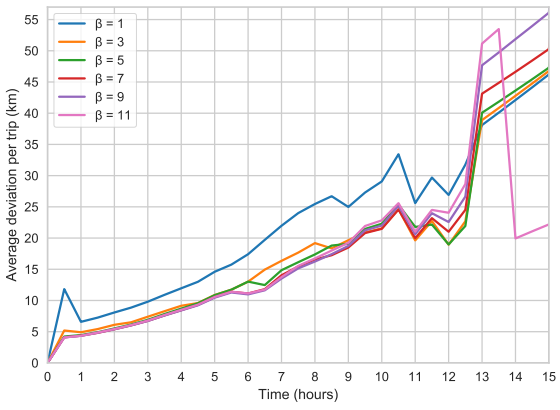
Table 4.5: Outliers for OPTICS clusters, for set 3.

Observation	Outliers
25%	21%
50%	28%
75%	42%

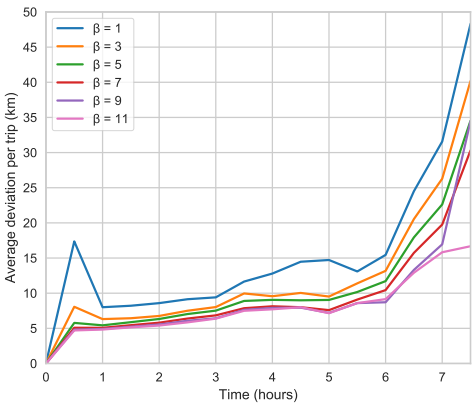
Figure 4.12: Comparative analysis of different beta values for each observation size – OPTICS clusters, on set 3.



(a) 25% of observation.

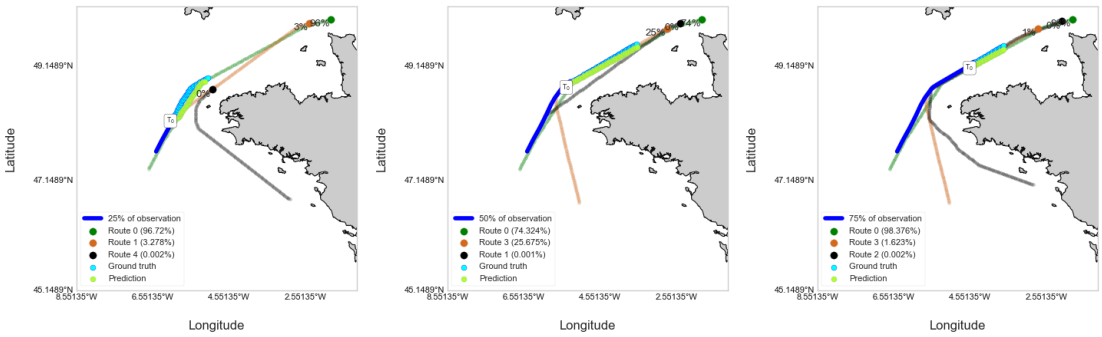


(b) 50% of observation.



(c) 75% of observation.

Figure 4.13: Example of movement prediction with OPTICS clusters, for set 3.



(a) 25% of observation – average deviation of 5 km.

(b) 50% of observation – average deviation of 2 km.

(c) 75% of observation – average deviation of 3 km.

5

Conclusions and Future Work

5.1 Conclusions

This thesis focused in providing a study for the problem of route prediction, exploring Machine Learning techniques to automatically extract route patterns from historical AIS data, from which main routes are obtained to provide guidelines of future movement. First, a distance-based clustering was made and three methods were tested: Agglomerative Clustering, DBSCAN and OPTICS. On the same topic, two similarity measures, DTW and LCSS, were evaluated to be used in the clustering algorithms. Ultimately, a probabilistic route prediction approach was proposed based on the weighted movement of the three nearest routes.

The presented solution was tested in a 6-month real AIS dataset, regarding the region of Brittany, in France. The dataset was further divided in three sub-sets, containing trajectories with more than 100 km, 200 km and 300 km, respectively, to evaluate what kind of trajectories would characterize the region best.

The evaluation showed that LCSS offered better results than DTW, providing a more comprehensive view on the distances values. This was the measure used in the clustering testing.

HAC was the only clustering method to obtain reasonable clusters for the 1^{st} set, even though many of the clusters had misplaced trajectories. Unlike DBSCAN and OPTICS, it could unravel groups of movements within the aforementioned big cluster. The fact that HAC applies a cutoff to nodes that exceed the distance defined by the linkage criteria certifies that the clusters had some overall similarity. Density-based approaches do not necessarily guarantee that a cluster's shape is maintained. This cutoff was also responsible for creating 21 single-trajectory clusters out of 65.

For set 2, while the number of trajectories was reduced, working with longer trajectories proved to achieve better results with OPTICS; HAC had few clusters, but their quality was improved in terms of finding local behaviors from the clusters for set 1, and providing paths with more information; the parameters yielded by the model selection of DBSCAN did not provide reasonable clusters, grouping most of the data into two big clusters (upwards and downwards movements).

A decrease in the number of clusters was seen in all clustering algorithms for the 3^{rd}, with the exception of DBSCAN that had the same and rendered useless clusters again. Indeed this set is much smaller than the previous ones; nonetheless the quality of the clusters for HAC and OPTICS was better than of set 1, and a good compromise between the data size and the clusters representation was achieved.

The prediction was then evaluated with the clusters obtained from HAC, for set 1, and, HAC and OPTICS, for set 2 and 3. The remaining clusters rendered no useful clusters. Concerning the 1^{st} set, the predictions with HAC did not obtain a good performance overall; having more information about the vessel's position did not provide lower average deviation; a deviation of less than 10 km was, at most, achieved for 3 hours, with 50% of observation. It showed that the clusters did not provide a good

representation of the main paths, mainly due to having more small trajectories that do not bring useful information about a main path, and are an easy gateway to unwanted merges between clusters. This was already reflected in the re-classification process. For the remaining sets, it was proved that using the positional information from one route only could provide better estimates, and as the information regarding one ship is accumulated, the movement forecasting becomes more accurate. In order to construct the map of main routes, it is not required for the trajectories to have the highest sailed distance. In fact, a settlement between medium to long-sized trajectories can be established and still provide fair results in forecasting.

To sum, this thesis provided a straightforward method to approach long-term vessel route prediction, where, upon the knowledge extraction of the main maritime routes from a given region, they contain the characteristic movement of the ships sailing those routes.

5.2 Future Work

As a future reference for improvements of the present work, the following are highlighted:

- Although this work focused on the prediction of long and stable movement of transportation vessels, a more complete approach would be to include fishing vessels and characterize their movement. In fact, a significant share of the ships in the dataset were from fishing vessels.
- Test and compare more robust distance measures.
- Enrich the clustering of trajectories with other types of information, such as the origin and destination.
- It would also be interesting to compare the existing prediction approach to other methods and study the integration of other sources of information, such as weather and sea conditions.

Bibliography

[1] D. G. D. Zeinalipour-Yazti, S. Lin, “Distributed Spatio-Temporal Similarity Search,” *ACM 15th Conference on Information and Knowledge Management, (ACM CIKM 2006)*, Novemeber 2006.

[2] F. D. Salim, E. Rosalina, and F. D. Salim, “Automated density-based clustering of spatial urban data for interactive data exploration,” *The IEEE International Conference on Computer Communications (INFOCOM) Workshops*, no. May, 2017.

[3] N. Magdy, M. A. Sakr, T. Mostafa, and K. El-Bahnasy, “Review on trajectory similarity measures,” *2015 IEEE 7th International Conference on Intelligent Computing and Information Systems, ICICIS 2015*, no. December, pp. 613–619, 2016.

[4] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, “Scikit-learn: Machine learning in Python,” *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[5] NATO. (2011, May) Official Texts: Alliance Maritime Strategy. [Online]. Available: https://www.nato.int/cps/en/natohq/official_texts_75615.htm

[6] L. Cazzanti, L. M. Millefiori, and G. Arcieri, “A document-based data model for large scale computational maritime situational awareness,” *Proceedings - 2015 IEEE International Conference on Big Data, IEEE Big Data 2015*, pp. 1350–1356, 2015.

[7] R. Kerbiriou, A. Rajabi, and A. Serry, “The Automatic Identification System (AIS) as a Data Source for Studying Maritime Traffic,” 2018.

[8] B. Ristic, B. La Scala, M. Morelande, and N. Gordon, “Statistical analysis of motion patterns in AIS data: Anomaly detection and motion prediction,” *Proceedings of the 11th International Conference on Information Fusion, FUSION 2008*, pp. 40–46, 2008.

[9] Y. U. Zheng, “Trajectory Data Mining : An Overview,” vol. 6, no. 3, pp. 1–41, 2015.

[10] Z. Feng and Y. Zhu, "A Survey on Trajectory Data Mining: Techniques and Applications," *IEEE Access*, vol. 4, pp. 2056–2067, 2016.

[11] P. Giuliana, V. Michele, and B. Karna, "Vessel pattern knowledge discovery from AIS data: A framework for anomaly detection and route prediction," *Entropy*, vol. 15, no. 6, pp. 2218–2245, 2013.

[12] S. Gan, S. Liang, K. Li, J. Deng, and T. Cheng, "Ship trajectory prediction for intelligent traffic management using clustering and ANN," *2016 UKACC International Conference on Control, UKACC Control 2016*, 2016.

[13] S. Hexeberg, A. L. Flaten, B. O. H. Eriksen, and E. F. Brekke, "AIS-based Vessel Trajectory Prediction," *20th International Conference on Information Fusion, Fusion 2017 - Proceedings*, 2017.

[14] R. Laxhammar, "Anomaly Detection for Sea Surveillance," *Proceedings of the 11th International Conference on Information Fusion (FUSION)*, pp. 55–62, 2008.

[15] T. W. Liao, "Clustering of time series data — a survey," vol. 38, pp. 1857–1874, 2005.

[16] J. Bian, D. Tian, Y. Tang, and D. Tao, "A survey on trajectory clustering analysis," 2018. [Online]. Available: http://arxiv.org/abs/1802.06971

[17] M. Nanni, S. Kisilevich, F. Mansmann, M. Nanni, and S. Rinzivillo, "Spatio-Temporal Clustering : a Survey Spatio-Temporal Clustering : a Survey," no. September, 2015.

[18] H. Wang, H. Su, K. Zheng, S. Sadiq, and X. Zhou, "An effectiveness study on trajectory similarity measures," *Conferences in Research and Practice in Information Technology Series*, vol. 137, no. February, pp. 13–22, 2013.

[19] P. Besse, B. Guillouet, J.-M. Loubes, and R. François, "Review and Perspective for Distance Based Trajectory Clustering," pp. 1–10, 2015. [Online]. Available: http://arxiv.org/abs/1508.04904

[20] D. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series," *Workshop on Knowledge Knowledge Discovery in Databases*, vol. 398, pp. 359–370, 1994. [Online]. Available: http://www.aaai.org/Papers/Workshops/1994/WS-94-03/WS94-03-031.pdf

[21] E. J. Keogh and M. J. Pazzani, "Derivative Dynamic Time Warping," pp. 1–11, 2000.

[22] G. Kollios, M. Vlachos, and D. Gunopulos, "Discovering Similar Multidimensional Trajectories," *Encyclopedia of GIS*, pp. 1–8, 2016.

[23] L. Chen, M. T. Özsu, and V. Oria, "Robust and fast similarity search for moving object trajectories," *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pp. 491–502, 2005.

[24] L. Chen and R. NG, “On The Marriage of Lp-norms and Edit Distance,” *Proceedings 2004 VLDB Conference*, pp. 792–803, 2004.

[25] G. Yuan, P. Sun, J. Zhao, D. Li, and C. Wang, “A review of moving object trajectory clustering algorithms,” *Artificial Intelligence Review*, vol. 47, no. 1, pp. 123–144, 2017.

[26] W. Li, C. Zhang, J. Ma, and C. Jia, “Long-term vessel motion predication by modeling trajectory patterns with AIS data,” *ICTIS 2019 - 5th International Conference on Transportation Information and Safety*, pp. 1389–1394, 2019.

[27] G. Spiliopoulos, D. Zissis, and K. Chatzikokolakis, “A big data driven approach to extracting global trade patterns,” *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 10731 LNCS, pp. 109–121, 2018.

[28] P. Sheng and J. Yin, “Extracting Shipping Route Patterns by Trajectory Clustering Model Based on Automatic Identification System Data,” 2018.

[29] N. Le Guillarme and X. Lerouvreur, “Unsupervised extraction of knowledge from S-AIS data for maritime situational awareness,” *Proceedings of the 16th International Conference on Information Fusion, FUSION 2013*, no. December 2014, pp. 2025–2032, 2013.

[30] M. Vespe, I. Visentini, K. Bryan, and P. Braca, “Unsupervised learning of maritime traffic patterns for anomaly detection,” pp. 14–14, 2012.

[31] D. Yao, C. Zhang, Z. Zhu, J. Huang, and J. Bi, “Trajectory Clustering via Deep Representation Learning,” *2017 International Joint Conference on Neural Networks (IJCNN)*, pp. 3880–3887, 2017.

[32] J.-g. Lee and J. Han, “Trajectory Clustering: A Partition-and-Group Framework.”

[33] S. Aghabozorgi, A. Seyed Shirkhorshidi, and T. Ying Wah, “Time-series clustering - A decade review,” *Information Systems*, vol. 53, pp. 16–38, 2015. [Online]. Available: http://dx.doi.org/10.1016/j.is.2015.04.007

[34] K. T., “The self-organizing maps,” *Proceedings IEEE 78*, 1990.

[35] M. Riveiro, F. Johansson, G. Falkman, and T. Ziemke, “Supporting maritime situation awareness using Self Organizing Maps and Gaussian Mixture Models,” *Frontiers in Artificial Intelligence and Applications*, vol. 173, no. January, pp. 84–91, 2008.

[36] S. Gaffney and P. Smyth, “Trajectory clustering with mixtures of regression models,” in *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM Press, 1999, pp. 63–72.

[37] D. Chudova, S. Gaffney, E. Mjolsness, and P. Smyth, “Translation-invariant mixture models for curve clustering,” *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, no. 03, pp. 79–88, 2003.

[38] J. Alon, S. Sclaroff, and G. Kollios, “Discovering Clusters in Motion Time-Series Data,” 2003.

[39] M. Ramoni, P. Sebastiani, and P. Cohen, “Bayesian clustering by dynamics,” *Machine Learning*, vol. 47, no. 1, pp. 91–121, 2002.

[40] E. N. F. Nota, G. Benadering, R. E. N. Herintegratie, and S. E. N. Risicobeheersing, “Multivariate Clustering by Dynamics Marco,” *Drugs*, no. September, pp. 1–68, 2001.

[41] S. Zhong and J. Ghosh, “A Unified Framework for Model-Based Clustering,” *Journal of Machine Learning Research*, vol. 4, pp. 1001–1037, 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=945365.964287

[42] E. Tu, G. Zhang, L. Rachmawati, E. Rajabally, and G. B. Huang, “Exploiting AIS Data for Intelligent Maritime Navigation: A Comprehensive Survey from Data to Methodology,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 5, pp. 1559–1582, 2018.

[43] Y. Huang, L. Chen, P. Chen, R. R. Negenborn, and P. H. van Gelder, “Ship collision avoidance methods: State-of-the-art,” *Safety Science*, vol. 121, no. March, pp. 451–473, 2020.

[44] L. P. Perera, P. Oliveira, and C. Guedes Soares, “Maritime Traffic Monitoring Based on Vessel Detection, Tracking, State Estimation, and Trajectory Prediction,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 13, no. 3, pp. 1188–1200, 2012.

[45] M. Gao, G. Shi, and S. Li, “Online Prediction of Ship Behavior with Automatic Identification System Sensor Data Using Bidirectional Long Short-Term Memory Recurrent Neural Network,” *Sensors (Switzerland)*, vol. 18, no. 12, 2018.

[46] H. Rong, A. P. Teixeira, and C. Guedes Soares, “Ship trajectory uncertainty prediction based on a Gaussian Process model,” *Ocean Engineering*, vol. 182, no. April, pp. 499–511, 2019. [Online]. Available: https://doi.org/10.1016/j.oceaneng.2019.04.024

[47] P. Dierckx, *Curve and Surface Fitting with Splines*. Oxford University Press, 1993.

[48] T. J. Pires and M. A. Figueiredo, “Shape-based trajectory clustering,” *ICPRAM 2017 - Proceedings of the 6th International Conference on Pattern Recognition Applications and Methods*, vol. 2017-January, no. October, pp. 71–81, 2017.

[49] H. Späth, *One Dimensional Spline Interpolation Algorithms*. A K Peters/CRC Press, 1995.

[50] F. Camastra and A. Vinciarelli, “Clustering Methods,” *Machine Learning for Audio, Image and Video Analysis*, pp. 117–148, 2008.

[51] V. K. P. Tan, M. Steinbach, *Data Mining Cluster Analysis: Basic Concepts and Algorithms*, 2013.

[52] M. Daszykowski and B. Walczak, “Density-Based Clustering Methods,” *Comprehensive Chemometrics*, vol. 2, pp. 635–654, 2009.

[53] M. Pourrajabi, “Model selection for semi-supervised clustering,” *Proceedings of the 17th International Conference on Extending Database Technology (EDBT)*, vol. 28, no. 2, pp. 49–60, 2014.

[54] M. Ankerst, M. M. Breunig, H.-p. Kriegel, and J. Sander, “Optics: Ordering points to identify the clustering structure,” *ACM SIGMOD Record*, vol. 28, no. 2, pp. 49–60, 1999.

[55] Y. B. I. Goodfellow and A. Courville, “Deep-learning,” *MIT Press*, 2016.

A

Tables

Table A.1: Dynamic data description.

Column	Data Type	Description
Source MMSI	integer	Vessel MMSI identifier
Navigational Status	integer	Navigational status (e.g.: 1- at anchor, 7- engaged in fishing, 8- under way sailing)
Rate of Turn	double precision	Rate of turn, right or left, 0 to 720 degrees per minute
SOG	double precision	Speed over ground in knots (allowed values: 0-102.2 knots)
COG	double precision	Course over ground in degrees
True Heading	integer	True heading in degrees, relative to true north
Lon	double precision	Longitude (georeference: WGS 1984)
Lat	double precision	Latitude (georeference: WGS 1984)
T	bigint	Timestamp in UNIX epochs

Table A.2: Static data description.

Column	Data Type	Description
Source MMSI	integer	Vessel MMSI identifier
IMO Status	integer	IMO ship identification number
callsign	text	International radio call sign
shipname	text	Name of the vessel
shiptype	integer	Code for the type of the vessel (e.g.: 30- fishing, 60-69: passenger, 36- sailing vessel)
to-bow	integer	Distance (meters) to Bow
mothershipmmsi	double precision	Longitude (georeference: WGS 1984)
to-stern	integer	Distance (meters) to Stern= to-bow + to-stern = length of the vessel
to-starboard	integer	Distance (meters) to Starboard, i.e., right side of the vessel = to-port + to- starboard = beam at the vessel's nominal waterline
to-port	integer	Distance (meters) to Port, i.e., left side of the vessel (meters)
eta	text	ETA (estimated time of arrival) – UTC time zone
draught	double precision	Allowed values: 0.1-25.5 meters
destination	text	Destination of this trip (manually entered)
T	bigint	Timestamp in UNIX epochs

B

Code

Algorithm B.1: LCSS Algorithm with Cosine Distance

```python
def lcss_cosine(t1, t2, eps):
    n0 = len(t1)
    n1 = len(t2)
    C = [[0] * (n1 + 1) for _ in range(n0 + 1)]
    for i in range(1, n0 + 1):
        for j in range(1, n1 + 1):
            if cosine_distance(t1[i - 1], t2[j - 1]) < eps:
                C[i][j] = C[i - 1][j - 1] + 1
            else:
                C[i][j] = max(C[i][j - 1], C[i - 1][j])
    return C[n0][n1])
```

Algorithm B.2: LCSS Joint distance

Input: Trajectory1, Trajectory2
Output: Percentage representing how much the trajectories match
begin
  $position\_sim \longleftarrow lcss\_dist(trajectory1.positions, trajectory2.positions)$
  $direction\_sim \longleftarrow lcss\_cosine(trajectory1.angles, trajectory2.angles)$
  $max\_value \longleftarrow max(position\_sim, direction\_sim)$
  if $max\_value > 60$ **then**
    return max_value
  else
    return $(position\_sim + direction\_sim)/2$

Algorithm B.3: DTW Joint distance

Input: Trajectory1, Trajectory2
Output: Distance in km between trajectories
begin
  $shortest\_traj, longest\_traj \longleftarrow find\_lengths(trajectory1, trajectory2)$
  $longest\_subsequence \longleftarrow get\_longest\_subsequence(shortest\_traj, longest\_traj)$
  $position\_sim \longleftarrow$
   $dtw\_from\_metric(shortest\_traj.positions, longest\_subsequence.positions, haversine)$
  $direction\_sim \longleftarrow$
   $dtw\_from\_metric(shortest\_traj.angles, longest\_subsequence.angles, cosine\_distance)$
  return $(position\_sim + direction\_sim)/2$