HAROKOPIO UNIVERSITY

SCHOOL OF DIGITAL TECHNOLOGY

DEPARTMENT OF INFORMATICS AND TELEMATICS

Real-time Mobility Data Analytics: Methodologies for Efficient Anomaly Detection and Trajectory Classification

Doctoral Thesis

Ioannis Kontopoulos

Athens, 2022

ΧΑΡΟΚΟΠΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

ΣΧΟΛΗ ΨΗΦΙΑΚΗΣ ΤΕΧΝΟΛΟΓΙΑΣ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΜΑΤΙΚΗΣ

Ανάλυση σε πραγματικό χρόνο δεδομένων κινητικότητας και ειδικά η ανάπτυξη μεθοδολογιών για την αποτελεσματική ανίχνευση μη αναμενόμενων συμπεριφορών και την κατηγοριοποίηση τροχιών

Διδακτορική Διατριβή

Ιωάννης Κοντόπουλος

Αθήνα, 2022

HAROKOPIO UNIVERSITY

SCHOOL OF DIGITAL TECHNOLOGY

DEPARTMENT OF INFORMATICS AND TELEMATICS

Declaration of Committee

Konstantinos Tserpes (Supervisor)
Associate Professor, Informatics & Telematics, Harokopio University of Athens

Dimosthenis Anagnostopoulos (Examiner)
Professor, Informatics and Telematics, Harokopio University of Athens

Mara Nikolaidou (Examiner)
Professor, Informatics and Telematics, Harokopio University of Athens

Iraklis Varlamis (Examiner)
Associate Professor, Informatics and Telematics, Harokopio University of Athens

Dimitrios Michail (Examiner)
Assistant Professor, Informatics and Telematics, Harokopio University of Athens

Dimitrios Zissis (Examiner)
Associate Professor, Product and Systems Design Engineering, University of the Aegean

Yiannis Theodoridis (Examiner)
Professor, Informatics, University of Piraeus

The acceptance of the PhD Thesis from the Department of Informatics and Telematics of Harokopio University does not imply the acceptance of the author's point of view.

Ethics and Copyright Statement

I, Ioannis Kontopoulos, hereby declare that:

1) I am the owner of the intellectual rights of this original work and to the best of my knowledge, my work does not insult persons, nor does it offend the intellectual rights of third parties.

2) I accept that Library and Information Centre of Harokopio University may, without changing the content of my work, make it available in electronic form through its Digital Library, copy it in any medium and / or any format and hold more than one copy for maintenance and safety purposes.

3) I have obtained, where necessary, permission from the copyright owners to use any third-party copyright material reproduced in the master thesis while the corresponding material is visible in the submitted work.

Acknowledgements

I would like to thank all the people and my family especially for supporting me during the course of this PhD. Furthermore, I would also like to thank my supervisor Dr. Konstantinos Tserpes for giving me the opportunity to start my PhD thesis and for his help throughout my years as a PhD student. Moreover, I would like to thank Dr. Dimitrios Zissis for his support and the opportunity he gave me to work for the research team of MarineTraffic, thus allowing me to acquire hands-on experience in the industry and in the maritime domain. Finally, special thanks goes to Konstantinos Chatzikokolakis and Giannis Spiliopoulos, members of the MarineTraffic research team, for helping me write my first papers, giving me ideas and challenge me to find solutions.

Contents

List of Figures

List of Tables

Περίληψη

Στην εποχή μας, ο αυξανόμενος αριθμός των αισθητήρων των κινητών αντικειμένων έχει ως αποτέλεσμα τη συνεχόμενη παραγωγή ροών δεδομένων υψηλής συχνότητας και μεγάλου όγκου. Αυτό το φαινόμενο παρατηρείται πολύ στον τομέα της ναυτιλίας όπου τα περισσότερα πλοία παγκοσμίως μεταδίδουν την τοποθεσία τους περιοδικά. Επομένως, υπάρχει μεγάλη ανάγκη για εξαγωγή χρήσιμης πληροφορίας και αναγνώριση μοτίβων κίνησης από αυτά τα δεδομένα με έναν αυτόματο τρόπο. Επιπλέον, η αύξηση αυτών των δεδομένων θέτει νέες προκλήσεις στην κοινότητα της εξαγωγής δεδομένων όσον αφορά την αποδοτική ανάλυση και εξαγωγή γνώσης.

Η υποχρεωτική χρήση του αυτόματου συστήματος αναγνώρισης (Automatic Identification System - AIS) - ένα σύστημα παρακολούθησης πλοίων - σε πολλά πλοία, που έχει επιβληθεί από τους κανονισμούς ναυτιλίας, έχει ανοίξει νέες ευκαιρίες για τη ναυτιλιακή παρακολούθηση. Οι μεταδότες AIS είναι πλούσια πηγή πληροφοριών που ο καθένας μπορεί να συλλέξει με τη χρήση ενός δέκτη RF και παρέχουν πληροφορίες σε πραγματικό χρόνο για τις θέσεις των πλοίων. Η εκμετάλλευση δεδομένων AIS μπορεί να αποκαλύψει παράνομη συμπεριφορά, να προσφέρει ειδοποιήσεις σε πραγματικό χρόνο και να ενημερώσει τις αρχές για κάθε είδος παράξενης συμπεριφοράς. Μια μεγάλη πρόκληση στην παρακολούθηση πλοίων είναι η αναγνώριση γεγονότων ενδιαφέροντος μέσα από ογκώδεις και υψηλής συχνότητας ροές δεδομένων σε πραγματικό χρόνο. Ακόμη μεγαλύτερη πρόκληση είναι η ανάπτυξη εφαρμογών στους επίγειους δέκτες που έχουν περιορισμένη ικανότητα επεξεργασίας. Επιπροσθέτως, οι γεμάτες λάθη και θόρυβο ροές δεδομένων παρακολούθησης πλοίων κάνει την ακριβή ανίχνευση γεγονότων ενδιαφέροντος ακόμη πιο δύσκολη. Σε αυτήν τη διατριβή προσπαθούμε να αντιμετωπίσου-με τις παραπάνω προκλήσεις.

Συνεπώς, προτείνουμε μια κατανεμημένη αρχιτεκτονική ικανή να αναγνωρίζει γεγονότα μέσα από λανθασμένες και θορυβώδεις ροές δεδομένων παρακολούθησης πλοίων όπως το spoofing και το κλείσιμο των δεκτών μετάδοσης σε πραγματικό χρόνο. Επιπλέον, παρουσιάζουμε μια επέκταση ενός υπάρχοντος δικτύου ναυτιλιακής κίνησης που βασίζεται σε κόμβους που αντιστοιχούν σε ναυτικές περιοχές παρατεταμένης παραμονής πλοίων ή μεγάλων στροφών (παραδείγματος χάριν, λιμάνια, ακρωτήρια, πλατφόρμες) και ακμές που αντιστοιχούν σε διαδρομές πλοίων μεταξύ δύο διαδοχικών κόμβων. Η εστίαση του προβλήματος είναι στις συνδέσεις του δικτύου και στον εμπλουτισμό με σημασιολογική πληροφορία σχετικά με τον τρόπο διάσχισης μιας ακμής. Επομένως, αραιά ιστορικά δεδομένα παρακολούθησης πλοίων και πολυωνυμική παρεμβολή χρησιμοποιούνται για την εξαγωγή διαδρομών πλοίων. Προτείνεται μια παραλλαγή του αλγορίθμου ομαδοποίησης DBSCAN πάνω από μια κατανεμημένη αρχιτεκτονική, όπου οι παράμετροι εγγύτητας του αλγορίθμου αλλάζουν. Η

παραλλαγή του αλγορίθμου εκμεταλλεύεται τη διαφορά στην ταχύτητα, πορεία και θέση για να οριστεί η απόσταση μεταξύ δύο διαδοχικών θέσεων πλοίων.

Επιπλέον, καινοτόμες προσεγγίσεις παρουσιάζονται για την κατηγοριοποίηση δραστηριότητας πλοίων από ροές δεδομένων σε πραγματικό χρόνο. Παρουσιάζεται μια λύση που τμηματοποιεί τροχιές πλοίων σε πολλές μικρότερες τροχιές και ξεχωρίζει τα τμήματα στα οποία τα πλοία ψαρεύουν από άλλα τμήματα στα οποία τα πλοία απλά έχουν χαράξει πορεία προς τον προορισμό τους. Επίσης, παρουσιάζεται μια συγχώνευση των ερευνητικών τομέων του computer vision και της κατηγοριοποίησης τροχιών (trajectory classification). Ο στόχος αυτής της συγχώνευσης είναι να αυξήσει την ακρίβεια αναγνώρισης των μοτίβων κίνησης των πλοίων μέσα από τεχνικές deep learning σε πραγματικό χρόνο, υπερνικώντας τις προκλήσεις των μεγάλων δεδομένων όπως ο όγκος και η ταχύτητα. Τέλος, προς επίλυση των ίδιων προκλήσεων, διάφοροι αλγόριθμοι συμπίεσης τροχιών παρουσιάζονται και αξιολογούνται σε δεδομένα προερχόμενα από τροχιές πλοίων. Οι αλγόριθμοι συμπίεσης τροχιών που παρουσιάζονται σε αυτήν την έρευνα είναι κατάλληλοι είτε για ιστορικά δεδομένα είτε για δεδομένα πραγματικού χρόνου. Οι αλγόριθμοι αξιολογούνται ως προς το βαθμό συμπίεσης, την ταχύτητα εκτέλεσης και την απώλεια πληροφορίας. Παρουσιάζουμε τα ευρήματα αυτής της έρευνας που προορίζονται σε ερευνητές στον τομέα της έξυπνης παρακολούθησης πλοίων.

Λέξεις Κλειδιά: Κατηγοριοποίηση τροχιών; Ανίχνευση μη αναμενόμενης συμπεριφοράς; Ομαδοποίηση τροχιών; Ανάλυση σε πραγματικό χρόνο; Νευρωνικά δίκτυα.

Abstract

Nowadays, the increasing number of moving objects tracking sensors, results in the continuous flow of high-frequency and high-volume data streams. This phenomenon can especially be observed in the maritime domain since most of the vessels worldwide are now transmitting their positions periodically. Therefore, there is a strong necessity to extract meaningful information and identify mobility patterns from such tracking data in an automated fashion, eliminating the need for experts' input. Furthermore, this increase of vessel tracking data has posed new challenges in the data mining community in terms of efficient analytics and knowledge extraction out of such data.

The compulsory use of Automatic Identification System (AIS) -- a vessel tracking system -- for many vessel types, which has been enforced by naval regulations, has opened new opportunities for maritime surveillance. AIS transponders are rich sources of information that everyone can collect using an RF receiver and provide real-time information about vessels' positions. Properly taking advantage of AIS data, can uncover potential illegal behavior, offer real-time alerts and notify the authorities for any kind of anomalous vessel behavior. One major challenge in vessel tracking and surveillance is the ability to detect events of interest out of the voluminous and high-velocity streams of data in real-time. What is even more challenging is the deployment of applications on-site and on the terrestrial receivers that have a limited processing capacity. On top of that, the noisy and erroneous streams of vessel tracking data, makes the accurate detection of events of interest even more challenging. In this thesis, we aim to address the above challenges.

Towards this direction, we propose a distributed architecture able to identify erroneous or noisy events in streams of vessel tracking data such as spoofing and the switch-off of the vessel tracking transponder in real-time. Moreover, we introduce an extension of an existing network abstraction of maritime traffic, that is based on nodes (called way-points) that correspond to naval areas of long stays or major turns for vessels (e.g. ports, capes, offshore platforms etc.) and edges (called traversals) that correspond to the routes followed by vessels between two consecutive way-points. The focus is the connections of the network abstraction and the enrichment with semantic information about the different ways that vessels employ when traversing an edge. For achieving this, sparse historic vessel tracking data and polynomial interpolation are employed in order to extract shipping lanes; an alternative of the popular density based clustering algorithm DBSCAN is proposed over a distributed architecture, which modifies the proximity parameter of the algorithm. The proposed alternative employs in tandem the difference in i) speed, ii) course and iii) position for defining the distance between two consecutive vessel positions (two consecutive AIS signals received from the same vessel).

Furthermore, novel approaches are presented for the classification of vessel activity from real-time data streams. A solution is presented that splits vessel trajectories into multiple overlapping segments and distinguishes the ones in which a vessel is engaged in trawling or longlining operation (e.g. fishing activity) from other segments that a vessel is simply underway from its departure towards its destination. Moreover, a fusion of the research fields of computer vision and trajectory classification is introduced. The aim of this fusion is the delivery of a high-precision classification of mobility patterns through deep learning schemes in near real-time, tackling the Big Data challenges of volume and velocity. Finally, towards the solution of the same challenges a wide range of several well-known trajectory compression algorithms is presented and evaluated on data originating from vessel trajectories. Trajectory compression algorithms included in this research are suitable for either historical data (offline compression) or real-time data streams (online compression). The performance evaluation is three-fold and each algorithm is evaluated in terms of compression ratio, execution speed and information loss. We present our results and findings intended for both researchers and practitioners in the field of intelligent ship tracking and surveillance.

Keywords: Trajectory Classification; Anomaly Detection; Trajectory Clustering; Real-time Analysis; Neural Networks.

Part I
Setting the scene

1 Introduction

The recent years, the increase of vessel positional data through emerging vessel tracking systems has led to the production of vast amounts of mobility data. This increase of vessel tracking data has posed new challenges in the data mining community in terms of efficient analytics and knowledge extraction out of such data. Therefore, in this chapter, we introduce the problem at hand, provide the motivation and present some interesting application scenarios. Furthermore, the newly emerged challenges and the contributions of this thesis are presented. Finally, we present the structure of this thesis.

1.1 Motivation

Nowadays vessels produce unprecedented amounts of information contrary to the past where the field of vessel monitoring suffered from a lack of data. Systems that utilize this data, such as the Vessel Traffic Management System (VTMS[1]) and the Vessel Traffic Management Information System (VTMIS), have been available for many years. These systems have made a significant contribution to increasing the efficiency and safety of operations and operations at sea. Nevertheless, environmental (eg., the opening of the northwestern passage) and socio-economic (eg., influx of refugees and migrants crossing the Mediterranean) factors again increase the risks at sea, thus requiring the development of VTMS and VTMIS systems, in order for the human-operator to adequately understand the complex reality and to enhance decision-making. A critical requirement of such systems is the ability to anticipate emerging and potentially hazardous situations in order to propose risk avoidance measures.

The recent approval of the European Union Maritime Security Strategy (EUMSS) - Action Plan, published on December 16^{th} 2014, has stated that discovering and characterizing the activities of vessels at sea are key tasks to Maritime Situational Awareness (MSA) and constitute the indispensable basis for a fully capable Maritime Security (MS) plan. The achieved knowledge requires the classification and prediction of vessel activities, as well as the detection of anomalous behaviors, enabling an effective and quick response to maritime threats and risks. Maritime scenarios target the analysis of the evolution of traffic in different time scales (weeks, seasons, years) in order to:

- provide support to the maritime spatial planning operations when new traffic separation schemes need to be assessed, new infrastructures at sea need to be built (e.g., wind off-shore platform, oil platforms);
- inform the traffic modeling and simulation techniques;
- facilitate search and rescue operations;
- contribute to the reduction of green-house gas (GHG) emission intensity for each vessel by providing data to increase and optimise operational efficiency;
- accomplish a decrease in port congestion and seaways by monitoring and improving the forecasting of vessel arrivals (ship sizes, cargoes, ETAs, loading/discharge times) to enable better planning and execution of port operations (virtual arrivals).

Maritime navigation technology can automatically provide real-time information from sailing vessels. The Automatic Identification System (AIS) is a tracking system for identifying and locating vessels at sea through data exchange: with AIS base stations along coastlines, or satellites when out of range of terrestrial networks. Although AIS data are only legally required for larger

[1] https://www.leonardocompany.com/documents/20142/107847/body_body_mm07821_VTMS_LQ_.pdf/84e90079-3888-8765-fa7d-1cc250dd09d3?t=1538989814697

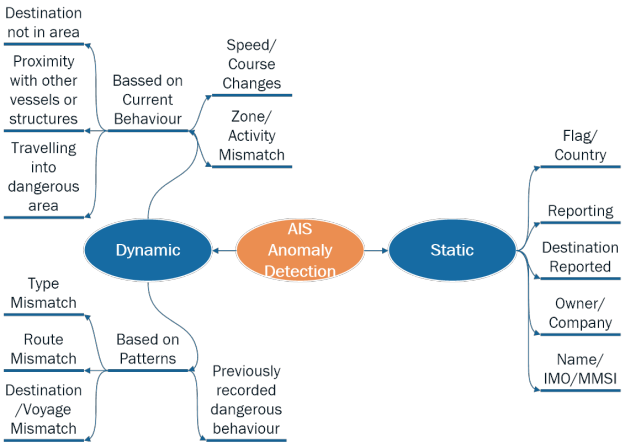


Figure 1: AIS anomaly detection.

vessels, their use is growing (e.g., their use has been recently extended to smaller fishing vessels according to the European Commission regulations in the Mediterranean Sea), and they can be effectively exploited to infer different levels of contextual information, from the characterization of ports and off-shore platforms to spatial and temporal distributions of routes. Nevertheless, as the amount of the available AIS data grows to massive scales, any approach to vessel activity characterization must contend with the higher-level challenges posed by massive data analysis, such as data representation, uncertainty, sampling bias and inference, from disparate data sources (AIS data from coastal and satellite receivers, weather data, geographical data, environmental data, archival data concerning vessels routes and trajectories), especially when these methodologies need to be validated operationally within security and defense constraints.

Data analysis methods can help in the task of modeling the behavior of vessels and answering questions related to their current or future condition, including predicting the position of the vessel in the coming minutes. Therefore, there are many applications that require real-time monitoring of abnormal sequential patterns on streams of trajectories or vessel tracking data. Figure 1 illustrates the possible anomalous events that might occur through an AIS transponder. These anomalous events can be considered as static or dynamic depending on whether the anomaly is due to static phenomena – incorrect flag, name or identifier – or dynamic phenomena – dangerous behavior or mobility patterns – which are based on the trajectory and behavior of the vessel. Consequently, a MSA system needs to be able to not only identify or warn about such anomalies and complex events, but it also needs to respond in real-time, thus increasing the maritime safety.

1.2 Application Scenarios

To facilitate the discussion about the importance of this thesis and its impact in our society, let us cite some interesting application scenarios.

According to the latest EMSA report [2] for the period 2011-2016, $16,539$ shipwreck victims (i.e. when a ship, its equipment or cargo is affected by an accident) and $5,607$ work accidents (i.e. when the accident concerns only one person) were recorded in European seas. During the same period, a total of 253 ships were lost (average: 4 ships / month), while $11,686$ ships reported some damage, with the largest category being cargo ships (43%). 600 people lost their lives, not counting the estimated $4,000$ refugees and migrants who die trying to cross the Mediterranean [3] each year. This leads to another application scenario which greatly concerns the European Union the recent years.

For the most part, large numbers of migrants arrive in Europe by sea in wooden fishing boats or rubber boats. A large number of these vessels face dangers long before they approach Euro-

pean waters, creating a humanitarian imperative to locate migration routes, understand smuggling patterns and locate ships before they reach the point of crisis. As these vessels operate covertly, methodologies that track anomalous behavior or patterns will help investigate the extent to which they can generate knowledge of migration patterns, search and rescue behavior and hazards. This can be of particular value for policy evaluation, now that immigration is at the top of the Greek and EU political agenda. The above shows the urgent need for a vessel monitoring system that serves the purpose of timely and effective response to emergencies that can help reduce the risk of accidents at sea by at least 0.1%. Even such a seemingly small size will have significant social, economic and scientific implications.

Another application of great interest is the immediate identification of Illegal, Unreported and Unregulated (IUU) fishing activities in prohibited areas or nature protection areas which has gained much attention the recent years. A study showed that 640,000 tonnes of ghost gear is left in the world's oceans each year, which entangles and kills around 136,000 turtles, whales, seals, birds, and other sea animals[2]. These animals end up suffering long and painful deaths. The US government has established the Seafood Import Monitoring Program to address the issue[3]. The Food and Agriculture Organization (FAO) of the United Nations has commissioned a study of IUU fishing activities to determine whether the Organisation should provide guidance for the future estimation of IUU fishing activities[4]. Furthermore, specific fishing activities (e.g., trawling) and fishing gear are strongly linked to the indigenous fauna, thus a mechanism that identifies such operations is of utmost importance.

1.3 Challenges

Situational awareness is key in the maritime domain; for example preventing ship accidents by monitoring vessel activity represents substantial savings in financial cost for shipping companies (e.g., oil spill cleanup) and averts irrevocable damages to maritime ecosystems (e.g., fishery closure). Nowadays, maritime navigation technology can automatically provide real-time information from sailing vessels. Considering that AIS information is continuously emitted from approximately $200,000$ ships worldwide, generating up to $16,000$ events per second and 46GB of data per day, it evidently fulfills all four "V" challenges (Volume, Velocity, Variety, lack of Veracity), as well as the "D" challenge (Distribution of data sources) in Big Data management. Therefore, maritime surveillance systems should detect threats and abnormal activity over voluminous, fluctuating, and noisy data streams from thousands of vessels, and also correlate them with static data expressing vessel characteristics (type, tonnage, cargo, etc) and geographical information (such as bathymetric data and protected areas). Although methodologies do exist that try to detect events or anomalous behavior on AIS data [4, 5, 6, 7], many of those methods assume a-priori availability of the whole dataset (i.e., batch processing), have limited availability of ground truth data and use features that make them inapplicable to streaming applications. More specifically the training data are specific to limited vessel types (e.g., only fishing) or the features are linked to trajectory information such as departure and arrival port, that can be discovered only through batch analysis of historical records (i.e., after the completion of a voyage). This calls not only for distributed algorithms that address the scalability requirements, but for low-latency and high-throughput algorithms that can tackle the high-velocity and real-time data streams. In this context, several challenges arise that need further investigation:

- how to distribute and partition the data in a way that each processing node can perform its computation independently, thus minimizing the communication cost between the nodes;

[2] https://www.worldanimalprotection.org/illegal-fishing-threatens-wildlife

[3] https://www.worldwildlife.org/threats/illegal-fishing

[4] http://www.fao.org/iuu-fishing/tools-and-initiatives/iuu-fishing-estimation-and-studies/en/

- how to take advantage of streams of events as they arrive in order to detect anomalous behavior and complex events of interest with minimum latency;
- the development of algorithms that can be deployed on site (Terrestrial AIS receivers that often have minimum processing capabilities) and achieve low latencies;
- how to address the limited availability of data and infer knowledge that would otherwise require the entire batch of historical data.

1.4 Research Methodology

Since the early days of computer science, scientists have developed novel architectures, algorithms, data structures and programming languages as a means of creating practical solutions to their problems. Little did they know that their approach would later be established as research paradigm, called Design Science Research (DSR). DSR is an approach that requires the development of a novel, purposeful artifact for a specific problem domain. Evaluation of the artifact that ensures its utility for the specified problem must be provided. A novel research contribution is defined by the fact that the artifact must either solve a problem that has not yet been solved, or develop more efficient solutions to an existing problem compared to the older ones. The development, the evaluation of the artifact and the research results must be presented effectively both to technology-oriented and management-oriented audiences. For that reason, Hevner et al. [1] presented a set of 7 guidelines for design science research within the discipline of Information Systems (IS). The workflow of the IS research framework can be seen in Figure 2. In our work, we followed these guidelines accordingly:

1. **Design as an artifact:** The result of this research was an anomaly detection framework containing a wide variety of solutions dedicated to address problems specific to the maritime industry.
2. **Problem relevance:** The motivation behind this research was driven by the industry, thus the implemented solutions were designed for the industry. Existing algorithms and methodologies were studied in order to create either improved versions of them or develop novel approaches for existing problems with emphasis to their real-time deployment and distributed processing.
3. **Design evaluation:** Solutions designed for the application scenarios of the maritime domain, need to satisfy several business requirements such as low-latency performance. The methodologies presented in this research were extensively evaluated within the technical infrastructure of MarineTraffic[5] in real-world datasets and under real-world conditions. Both their efficiency and their deployability were rigorously evaluated in every step of the research process.
4. **Research contributions:** *Effective DSR must provide clear contributions in the areas of the design artifact [1].* The nature of the maritime domain problems that need to be addressed lies within the spectrum of spatio-temporal analysis. Objects at sea (e.g., vessels) move through space and time creating a temporally sorted sequence of positions defined by their coordinates and timestamp, called trajectory. Trajectory data by these moving objects need to be mined and processed in order to infer knowledge. Therefore, the results of this research contribute to the fields of spatio-temporal analysis, trajectory classification and clustering, data mining and machine learning.

[5] https://www.marinetraffic.com

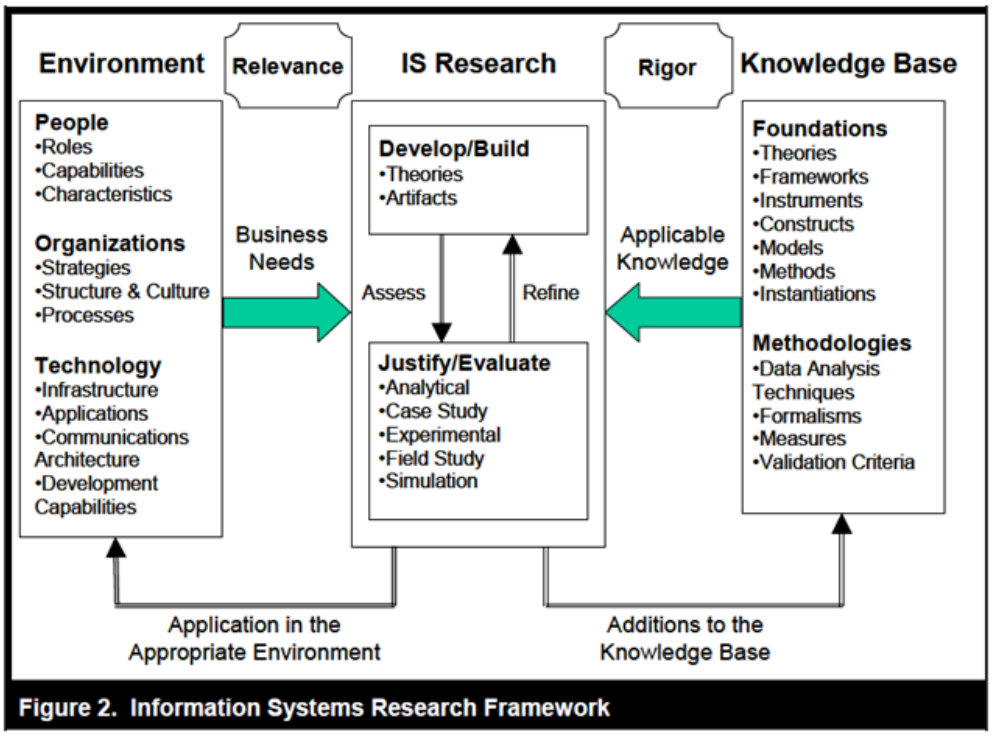


Figure 2: Information Systems Research Framework (source: [1]).

5. **Research rigor:** The research was conducted based on the evaluation standards that are required in the respective research fields. The same performance and evaluation metrics to previous research were used to assess the effectiveness of the developed solutions. Moreover, evaluation metrics derived from the industry were also employed to extensively evaluate the proposed methodologies.

6. **Design as a search process:** Simon [8] describes the nature of the design process as a Generate/Test Cycle. Existing methods for the industry problems at hand were explored and tested against their ability to satisfy the business requirements and constraints. Artifact ideas based on these requirements were developed and tested to meet the industry criteria. Alternatives of these artifacts were also explored and evaluated to result in better solutions.

7. **Communication of research:** The results of the respective research are published in high-impact-factor, peer-reviewed, international journals and conferences [9, 10, 11, 12, 13, 14, 15, 16, 17].

1.5 Contributions

The needs of the industry drive the motivations 1.1 and the application scenarios 1.2 behind this research, therefore maritime service providers such as MarineTraffic could benefit greatly by improving their existing products and applications. Several applications exist that try to address specific problems in the maritime domain such as storage and queyring of trajectories[6], yet they fail to demonstrate their feasibility in real-world settings. On the other hand, this research creates a framework containing a set of solutions that could be used as products satisfying the industry's requirements. Each solution addresses a different aspect of anomaly detection in the maritime domain but all of them satisfy the industry's needs for an improved Quality of Service (QoS): i)

[6] https://www.mobilitydb.com/

real-time analysis, ii) processing of huge volumes of trajectories, and iii) evaluation in real-world datasets that demonstrate their feasibility under real-world conditions. The solutions contained in this framework offer an alternate, yet novel approach for trajectory classification and clustering, as well as anomaly and event detection contributing to the research fields of spatio-temporal and trajectory analysis. Furthermore, the results and the findings are intended for both researchers and practitioners in the field of intelligent ship tracking and surveillance. The main contributions of this thesis and the set of solutions contained in this framework can be summarized below:

- We propose a real-time and distributed methodology that detects AIS spoofing events [9]. To do so, the Akka[7] framework was used that is suitable for highly concurrent and distributed systems based on the Actor model [18].
- As an extension of the previous methodology, we introduce an approach that detects gaps in AIS vessel trajectories in real-time [10, 11], built on top of the Akka framework.
- We propose a variation of the density-based, clustering algorithm, namely DBSCAN, able to detect events of interest or outliers in vessel trajectories [12]. In most cases, when the DBSCAN is used on vessel trajectories, only the spatial aspect is taken into account (e.g., the distance between each AIS position). The proposed variation can take advantage of both the speed and heading of the AIS positions to detect clusters of trajectories with similar behavior.
- We propose a distributed anomaly detection methodology [13] that is based on the variation of the DBSCAN algorithm. Through this methodology, maritime traffic patterns and routes are created that can be used to discover vessels that deviate from these predefined patterns and routes in real-time.
- We introduce a set of features that can be extracted online and as the streams of AIS data arrive through the system that are suitable for the classification of trajectories of fishing activities, namely trawling and longlining [14].
- We introduce a computer vision approach for trajectory classification that converts AIS trajectories into images [15]. The approach uses a set of well-established features in the field of computer vision that are exploited by common classifier such as Random Forests to identify patterns of interest in vessel trajectories.
- We propose a deep learning and streaming methodology that classifies AIS trajectories into events of interest in real-time [16]. The proposed approach uses well-known Convolutional Neural Networks (CNNs) to efficiently classify the trajectory images and acts as an extension to the aforementioned computer vision approach.
- We perform an experimental study of several trajectory compression algorithms over AIS data [17]. Several compression algorithms are evaluated that can work either on historical data (offline) or on streams of AIS data (online) and discuss the advantages and drawbacks of each algorithm in the maritime domain.

1.6 Thesis Organization

The rest of this thesis is organized as follows:

[7] https://akka.io/

Section 2 reviews the state-of-the-art in the fields of interest of this thesis. Specifically, the literature on anomaly detection, trajectory classification, and trajectory compression is discussed. The topic of anomaly detection is further segmented and analysed based on the methodologies developed in this thesis.

Part II deals with the identification of noteworthy or anomalous events in real-time with the use of hard-coded rules. More specifically, Chapter 3 introduces a distributed architecture and algorithm that identifies spoofing events in the streams of AIS data. Subsequently, Chapter 4 extends the architecture to support the detection of intentional communication gaps in the AIS streams.

In Part III, methods that develop a normalcy model of the maritime vessel traffic are introduced. In more detail, in Chapter 5, a modified version of the DBSCAN clustering algorithm for anomaly detection is presented. Then, Chapter 6, exploits the modified clustering algorithm to create a distributed framework that extracts shipping lanes of normal vessel traffic.

Part IV, investigates the classification of trajectories in real-time. As a result, Chapter 7, creates a set of features that can be extracted online and are suitable for the detection of illegal fishing activities. These features are then into well-known classifiers to classify parts of the vessel trajectories. In Chapter 8, a fusion of the fields of computer vision and trajectory classification is studied. Image features are extracted that can detect differences in the shape and speed of each trajectory that is represented as an image. Chapter 9 builds on top of the image representation of the trajectories and uses deep learning models to accurately classify vessel trajectories. Moreover, several trajectory compression algorithms are employed to study the effects of compression in the classification performance of the deep learning models. Subsequently, Chapter 10, reviews several trajectory compression algorithms and investigates their suitability in compressing streams of AIS data.

Finally, in Part VI we conclude this thesis. Specifically, Chapter 11 summarizes the merits of this thesis and Chapter 12 discusses future work directions.

2 State-of-the-art

Anomaly detection typically refers to the problem of finding patterns in data that do not conform to expected behaviour [19]. From image processing [20], sensor networks [21] and biological data [22] to vessel motion in the maritime domain [23] several approaches have been established for detecting anomalies. From the early works on anomaly detection in the maritime domain from Holst et al. [24] and the later works of Varlamis et al. [25] and Chatzikokolakis et al. [26] on the detection of search and rescue patterns, several representation models and algorithms have been developed to increase maritime situation awareness, identify potential illegal activities and detect anomalous patterns in the vessels' trajectories. In this section we try to organize and discuss the various topics of anomaly detection in the maritime domain since it is a multiple-aspect field of research. Moreover, the proposed approaches developed in this thesis are designed to work in real-time. Therefore, we also i) discuss trajectory compression algorithms that could be used to reduce the amount of data being processed by the proposed methodologies, and ii) briefly present an overview of stream-processing frameworks that could be exploited to achieve low latencies.

2.1 AIS Spoofing

A spoofing attack is a situation in which an attacker falsely impersonates a valid entity (masquerading), tricking the system into trusting the data provided. Ship AIS spoofing often involves creating a nonexistent vessel or masquerading vessel's identity with a false one, hiding or transmitting false positional data, so that a vessel appears or behaves legitimately (Figure 3). The rise of applications relying on AIS data for vessel tracking has increased the interest of researchers in detecting AIS vessel spoofing.

In [27], Balduzzi presents a method of spoofing a vessel by injecting invalid data into AIS gateways triggering fake collision warning alerts and potentially making surrounding vessels alter their course. Similarly in [27], a method of AIS hijacking is described, by which attackers can maliciously modify information provided by the AIS base station, eavesdropping on all transmissions (i.e. man-in-the-middle) or modifying data. This method makes it possible for attackers to alter any information broadcast by existing AIS stations regarding real vessel data (e.g., cargo, speed, location, and country). Iphar, Napoli, and Ray propose an architecture capable of identifying spoofing messages in AIS after it has entered the database, by using a methodology based on integrity and quality assessment of the AIS messages [28]. In Papi et al. [29], authors combine a classic radio-localization method based on time difference of arrival with an extended Kalman filter designed to track vessels in geodetic coordinates. Katsilieris, Braca, and Coraluppi focus on the problem of whether the received AIS data are trustworthy with the help of radar measurements and information from the tracking system [30].

Authors in [7] compare two methodologies for anomaly detection which both use the Gaussian Mixture Model (GMM) with a different algorithm for clustering. The first one uses the Expectation Maximization (EM) algorithm while the second one uses the greedy version of the EM algorithm. Both techniques consider momentary states of the vessel motion. As an extension of the approach proposed in [7], authors in [31] evaluate two models for detecting anomalies and their ability to distinguish simulated trajectories from real ones, the GMM and the Kernel Density Estimator (KDE). Results indicated that there is no significant difference in the performance of these two models.

Fewer works focus on real time spoofing detection. In Zissis [32], Support Vector Machines were employed on the edge of the cloud to perform low footprint unsupervised learning and analysis of sensor data, so as to detect spoofing attempts in streams of AIS data. In Salmon [33], authors propose an architecture which combines archived data analysis with real-time streams to process mobility data as they arrive, tackling the limitations introduced by either of the approaches. Moreover, they compare the results to purely offline or online approaches in terms of efficiency.

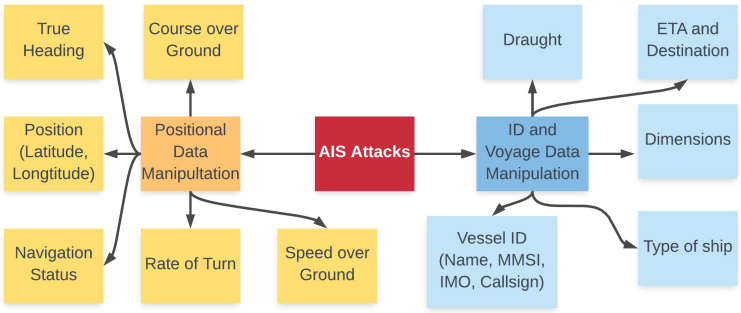


Figure 3: AIS attack tree by message type and parameter.

2.2 AIS Communication Gap

Although it is mandatory for vessels to carry an AIS transponder, it is not compulsory for the transponder to be switched on. As a result, many vessels "go dark" by switching off their transponder so as to hide potential illegal activity from the authorities or to hide their whereabouts. Guerriero, Coraluppi, Carthel, and Willett attempt to identify the intentional on-off switching of AIS, through the introduction of the notion of channel memory and the application of Hidden Markov Models (HMMs) [34, 35]. Similarly focusing also on the intentional on-off switching of AIS transponders, Mazzarella et. al. [36] explore a solution exploiting the Received Signal Strength Indicator available at the AIS Base Stations (BS). In designing such an anomaly detector, the electromagnetic propagation conditions that characterize the channel between ship AIS transponders and BS have to be taken into consideration. Furthermore, these approaches are not focused on real-time detection and are not applied to large scale datasets. In [37], authors present a methodology of online detection of communication gaps by taking into account the time period the vessel has not emitted a message.

2.3 Maritime Traffic Networks and Trajectory Clustering

Maritime traffic networks are a useful technique for compressing and abstracting multi-vessel trajectory data, but are also essential for the analysis of behavior of individual vessels or vessel groups and the detection of abnormal cases [23, 24], such as search and rescue maneuvers [25, 26], or other potentially illegal activities.

Several works on maritime surveillance have used grid partitioning of the surveillance area into tiles or hexagons [38] for mapping vessel trajectories to polylines or sequences of spatial indexes or key-points [39] in order to build maritime traffic network representations from historical AIS data [40, 41]. Arguedas et al. [40] propose a two-layer network: i) an external layer that uses way-points as nodes/vertices and routes as edges/lines and ii) an internal layer that consists of nodes or *breakpoints* that represent vessels' changes in behavior and edges or *tracklets* that represent vessel trajectories. The former layer is a traffic network abstraction, while the latter is a network that provides information about each vessel layer individually. While an edge in the first layer can a be a route from a port to another port, an edge in the internal layer comprises all the simplified trajectories (using Douglas-Peucker algorithm [42]) that sailed across this route. The complexity of the internal layer raises scalability issues that can be seen in the analysis of a real dataset. It is characteristic that the use of the 454 complete port-to-port routes in the small area of the Baltic sea resulted in an internal layer with $2,095$ tracklets. [41] create a graph-based representation of the maritime traffic, by extracting way-points (nodes of the graph), indicating areas where vessels change course and by creating sea lanes (edges of the graph) between way-points in the form of spatial ellipses. Pallota et al. [43] propose a methodology for anomaly detection based on a maritime traffic model. The model first extracts way-points or clusters from vessel positions or ports and

creates or updates the properties of the vessels in the surveillance area. It then extracts way-points and creates route objects by applying the DB-Scan algorithm, to the spatio-temporal and kinematic features of all vessel flows. The result is a model of maritime traffic patterns, called *TREAD*. At a later step, the observed vessel positions are classified as a route with a probability, and the future location of a vessel is predicted based on the selected route. The transition probabilities across the way-points are employed for detecting whether an actual vessel route is normal or not.

Clustering and outlier detection are two strongly coupled tasks in data mining. As shown from the above, the identification of outlying trajectories or unexpected route deviations is linked to the trajectory or sub-trajectory clustering. Several approaches have been used in the past to cluster moving objects and trajectory data [44, 45, 46]. [47] follow a similar approach to our methodology since they use a modified version of DB-Scan (called *DBSCANSD*) that considers vessels' speed and direction in tandem. However, *DBSCANSD* uses the maximum speed and direction variance, and employs user-defined thresholds for optimizing trajectory clustering. [47] do not evaluate their approach and only provide visualizations of the produced clusters.

Another approach to trajectory clustering is the well-known algorithm, called *Traclus* [48]. *Traclus* first partitions the trajectory into segments. The number of segments should be as small as possible (conciseness), while the difference between the segments and the trajectory itself should be small (preciseness). Then, the segments are grouped together (line segment clustering) based on three types of distances, the positional difference between segments from different trajectories, the positional difference between segments from the same trajectory, and the directional difference of the segments. While *Traclus* yields high performance results, it cannot be directly applied in our case, since we also consider the vessels' speed and we have to cluster each position separately, regardless of its parent trajectory (same or different). Finally, [49] uses the K-means clustering algorithm to form moving micro-clusters.

2.4 Trajectory Classification

Data mining techniques have been widely used to tackle the problems of anomaly detection and trajectory classification. Both problems require a classifier that is trained on several trajectory data indicating certain behaviours and then try to classify any future trajectory data that exhibit similar behaviour to a set of predefined labels. Another approach of detecting anomalies or complex behaviours is the research field of Complex Event Processing (CEP), where a set of predefined rules or patterns created by experts is given to the system, which later tries to identify such patterns in streams of events.

Several systems have been developed for the purposes of CEP [50, 51, 52, 53, 54]. The idea behind these systems is the use of a formal and expressive language which experts use to write patterns. Most systems use a SQL-like query language [50, 52, 53], while others employ logic-based rules [54] to describe complex events. These patterns are then matched against streams of events, usually in real-time, in order to produce a set of higher level events, called Complex Events. The field of CEP has attracted much attention the recent years and as such several well-known, open-source frameworks have implemented a CEP language such as Apache Flink[8]. Apache Flink offers a library, called FlinkCEP, which allows the user to perform CEP over distributed and streaming data.

Complex Event Processing is not absent in the maritime domain [55, 37, 56, 57] where the early detection of abnormal or illegal vessel behaviour is of interest to the authorities. Authors in [37] present a system which compresses streaming AIS messages to meaningful low-level events, called Simple Events. Simple events are then used to build higher, complex events with the use of the Event Calculus, a logical language for event reasoning. Authors later expand their patterns [58, 56] to support more complex events designed for the maritime domain, such as rendezvous of two or

[8] https://flink.apache.org/

more vessels, fishing and loitering. Tsogas et al. [57] developed a CEP engine, called TRITON, able to consume messages from various sources – Maritime radars, Long Range Identification and Tracking (LRIT) systems, AIS and Earth Observation satellites – and provide in near real-time, complex events describing encounters at sea, drifting, entering/exiting areas of interest and deviations of usual routes. Similarly, authors in [59] developed a distributed CEP system able to identify complex events such as AIS hijacking, engine malfunction or ship collisions.

Although CEP systems provide an expressive way to describe events, it is not always straightforward to understand when or how an event occurred, even by experts. For that reason, data analysis and classification techniques need to be employed in order to infer knowledge. In the field of trajectory classification, many works have focused on analysing the behaviour of the moving objects of interest. Several studies have used trajectories from Vessel Monitoring System (VMS) data to classify fishing activity [60, 61, 62, 63]. Huang et al. [63] tackle the problem of fishing vessel type identification based on only VMS trajectories. To do so, they extract trajectory features that are used in machine learning schemes of XGBoost in order to classify fishing vessels into nine types, achieving a classification accuracy of 95.42%. However, since the usage of AIS became compulsory for vessels and its positional transmission rate is much greater compared to the VMS, research studies have shifted towards the analysis of AIS data. In [64], the authors identified the moves and stops of fishing vessels in a specific area. To do so, they used a combination of algorithms, namely CB-SMoT [65] and DB-SMoT [66], which are able to take into account the speed variation of the trajectory and the direction of the trajectory respectively. Then, they used the DBSCAN algorithm to extract clusters indicating dense areas of fishing activity. Souza et al. [67] analysed the behaviour of fishing vessels using three different types of gear, namely trawlers, long-liners and purse-seiners. To distinguish fishing activity between gear types, they created different classification models per activity, in order to identify for each fishing activity the segments of the trajectory during which the vessels were engaged in fishing. The main drawback of this methodology is that it does not use a universal classifier for all fishing activities and the gear type is not always given by the AIS messages. Following the footsteps of [67], authors in [68] presented early promising results of classification performance with the use of neural networks and autoencoders. They evaluated their approach in 10 longline fishing vessels and compared their methodology with other classifiers such as Random Forests and SVMs. Similarly, authors in [69] use the DBSCAN algorithm to extract Points of Interest (POI) in the trajectories and create features from these points. Later, they use these features to train a classifier and achieve high-accuracy results. Finally, in [70], General Hidden Markov Models (GHMM) and Structural Hidden Markov Models (SHMM) are combined with a Genetic Algorithm (GA) in an attempt to classify trajectories. Their approach has been tested in two surveillance datasets, MIT car [71] and T15 [72], yielding promising results.

Another approach for trajectory classification is through the use of neural networks. Jiang et al. [73] employ Recurrent Neural Networks (RNNs) for point-based trajectory classification into four different transportation modes. The proposed method uses embedding of GPS data to map the original low-dimensional and heterogeneous feature space into distributed vector representations. The feature space is enriched with segment-based information, and maxout activations are employed by RNNs for an increased performance. The proposed method achieves 98% accuracy, outperforming several well-known classifiers such as Decision Trees, Support Vector Machines, Naive Bayes and Conditional Random Field.

In [74] a deep multi-scale learning model is used to model grid data under different space and time granularities, thus capturing the impact of space and time on classification results. Furthermore, an attention dense module is designed by combining an attention mechanism which is able to select major features and DenseNet which is able to enhance the propagation of local and spatial features throughout the network. Resulting feature representations are employed as the final classification results. The low-dimensional feature space and the limited feature compositions introduced by the heterogeneous AIS data, poses challenges related to the development of deep

learning models. Authors in [75] employ Deep Recurrent Neural Networks and specifically a novel partition-wise Gated Recurrent Units (pGRU) architecture for point-based trajectory classification on detecting trawler fishing activities. The proposed method maps low-dimensional features into another space with the use of a partition-wise activation function which applied before the linear transformation, and receives different parameters for distinct partitions to model them jointly in a nonlinear hierarchical deep structure. Thus, the proposed method is less sensitive to the quality of partitions and achieves better predictive results.

A more recent study of Chen et al. [76] aims at transportation mode classification of vessel trips from raw trajectory data. The proposed method integrates mobility modes identification and discovery by utilizing origin and destination point clustering, in order to discover route patterns from historical trajectories data. AIS trajectories are converted into a mobility-based trajectory structure that resembles a low-resolution image. The bounding box of the entire vessel trajectory is converted into a grid which contains multiple pixels. If the trajectory of the vessel falls within a pixel, then the pixel is colored by using a range of the grey color with each hue representing a different speed value. Furthermore, for mobility modes classification, a CNN model is utilized by taking advantage of labeled historical trajectory data, resulting in a maximum f1-score of 84.7%. The main drawback of [76] is that it does not capture the behavior of the trajectory in its entirety, since the surveillance space is segmented into large cells, especially in cases where the vessels perform micro-movements that eventually form a different mobility pattern. Another CNN architecture for vessel movement classification is employed in [77]. AIS raw trajectories are converted into trajectories images that contain three different vessel movement patterns, static, cruise and manoeuvring. Then a supervised learning method is used to train the CNN and the results showed an accuracy of about 75%. Finally, a similar structure is also used in [78] for the classification of the transportation mode of civilians (e.g. walking, bike, bus, taxi), achieving a maximum accuracy of 83.2%. These approaches do not capture the behavior of the trajectory in its entirety, since the surveillance space is segmented into large cells, especially in cases where the moving objects perform micro-movements that eventually form a different mobility pattern.

2.5 Trajectory Compression

Trajectory compression algorithms are widely used in various areas, such as cartography and computer graphics, trajectory clustering, road traffic, pedestrian movement information, weather analysis and map generalisation. The aim of trajectory compression algorithms is to balance the trade-off between the achieved compression rate and the acceptable degree of error.

Meratnia et al. [79] was among the first research studies considering three dimensional mobility data, where the temporal factor was taken into consideration in the compression techniques. Leichsenring et al. [80] present an evaluation of seven lossy compression algorithms with a view to identify the most important aspects in selecting the appropriate compression algorithm. Muckell et al. [81] also present a performance comparison of seven compression algorithms when utilizing two different errors, SED and the median difference in speed. The comparison is based on the execution times of compression algorithms and on the error committed. Chen et al. [82] present a lossy compression batch algorithm for GPS trajectories that considers both line simplification and quantization in the compression process. The quantization technique can improve the encoding procedure that takes into account speed and direction changes for selecting the approximated trajectory for compression. For the evaluation of the compression, the maximum SED is employed. Birnbaum et al. [83] present a method which exploits the similarities between sub-trajectories and creates a time mapping by finding for each time value of a trajectory the corresponding time value of a similar trajectory, using linear interpolation. A compression algorithm is applied on the time mapping, which removes the points but keeps the compression error under a user-defined threshold. The strong correlation between time values of similar trajectories, allows high compression of

time mapping.

A dynamic storage system called TrajStore which is able to co-locate and compress spatially and temporally "close" adjacent trajectories is presented in [84]. The compression algorithm combines two schemes: a lossless delta compression scheme which encodes spatio-temporal coordinates within a trajectory and a lossy compression scheme for clustering trajectories traveling on nearly identical paths. Another system called REST is presented in [85]. The proposed framework extracts a small collection of sub-trajectories, called reference trajectories, from raw trajectory data that form the compressed trajectory within a given spatio-temporal deviation threshold. In addition, greedy and dynamic algorithms achieve an optimal compression ratio and high levels of efficiency.

Regarding vessel trajectory compression from AIS data, several studies have already conducted. Liu et al. [86] present an adaptive Douglas-Peucker algorithm with automatic threshold selection for AIS-based trajectory compression. In general, Douglas-Peucker requires a static user-defined distance threshold to decide which points to retain or remove from the resulting curve. However, the choice of an optimal threshold value is difficult and differs across various applications. The proposed algorithm is able to maintain the main geometrical structures of each trajectory and automatically calculates a different threshold for each trajectory based on the actual trajectory characteristics. Then, the DTW algorithm with a warping window is employed in order to calculate the distances between the various time series. The evaluation of the proposed algorithm is based on classification and clustering experiments. Wei et al. propose a method in [87], which is able to compress AIS data by considering both spatial and motion features. Douglas-Peucker algorithm is employed to simplify trajectories, according to spatial features and a sliding window is adopted to simplify trajectories based on motion features such as the course alterations and speed variations. A fixed threshold of 0.8 times the ship length is employed in DP, as proposed in [88], while for sliding window the Gaussian distribution is utilized for threshold determination regarding speed and course variations. Then, a merging operation is performed, which gathers the simplification results of the two methods. The final compression results are evaluated based on the compression rate, length loss rate and shape similarity. DTW method is used to calculate the similarity between the original and the compressed trajectory. On the other hand, Zhao et al. [89] present an improved version of Douglas–Peucker which considers the shape of the vessel trajectory derived only from course. The method is able to recognize the straight and curve parts of a trajectory by detecting the transition point. Transition points are directly retained as the start and end points of the track segments. In the first phase of the algorithm, a window detects the clear turning while in the second, each segment is compressed by the traditional DP algorithm. For user-defined threshold elimination, the threshold of each individual trajectory is based on each ship's length. In [90], the authors propose a density-based spatial clustering of trajectories based on DBSCAN. In the clustering process, the similarities between trajectories are computed. As the number of AIS points are extremely large, the similarity process can be prohibitively costly. Thus, the DP algorithm is used again for AIS trajectory compression. The threshold of the algorithm is determined according to the changes in shape of the trajectories. Etienne et al. [91] reduced the number of positions of a trajectory by adopting the DP algorithm while retaining only the virtual positions. Their purpose was to optimize the calculation time of traffic flow pattern recognition. Vries et al. [92] employed DP algorithm to retain the stop and move information on the ship's trajectory. To the compressed trajectory data, alignment based similarity measures are applied. It is evident that Douglas-Peucker is among the most popular and successful algorithms on trajectory simplification due to its speed and accuracy.

In [93] the compression effect is evaluated using the KDE-based AIS vessel density visualization on both the compressed and uncompressed trajectory data. Thus, the threshold value selected is the one for which the visualization results are the same for both compressed and uncompressed data. Similarly in [94], the DP algorithm and kernel density estimation (KDE) algorithms are utilized for trajectory compression and visualization respectively. The massively large-scale parallel computing function computation capabilities of the Graphics Processing Unit (GPU) have been

employed to reduce the execution time of the Douglas-Peucker algorithm. The compression is evaluated through the density map visualizations produced by KDE.

As trajectory compression is very sensitive to parametrization, Fikioris et al. [95] present an approach for fine-tuning the selection of parameter values for compression. A genetic algorithm that iterates over several combinations of the parameter values is employed until converging to a suitable configuration per vessel type. This genetic algorithm is trained on a public dataset and as the system supports incremental optimization, by training in data batches, the performance continuously improves. The system aims to minimize the compression ratio, while at the same time to minimize the approximation error which is quantified with the Root Mean Square Error (RMSE). Sun et al. [96] propose an online compression of vessel trajectories, by utilizing the classic SPM (scan-pick-move) compression algorithm with the addition of a sliding window. The maximum offset distance reference trajectory point in the sliding window is utilized as the criterion of whether to retain or to discard a point. For the compression evaluation several performance metrics are employed such as compression time, compression ratio and compression error.

In our previous study [97], several trajectory compression algorithms are evaluated by employing classification techniques and similarity measures. Specifically, four classification methods which extract trajectory features are selected based on the classification accuracy. After the extraction of the features, the Random Forest algorithm is used for comparing the different techniques. Two distance measures are employed, EDR and UMS for trajectory similarity. In order to evaluate the similarity results, the F-Score metric was utilized. Four real-world datasets were selected in order to capture the full spectrum of trajectories ranging from the mobility of people, vehicles, animals even natural phenomena. On the other hand, in the present work our intention is to focus specifically on the effect of compression over AIS data. Thus, a wide range of offline and online compression algorithms are compared in contrast to our previous work which compares only five offline compression algorithms. Furthermore, the evaluation process in this work is based solely on the average distance produced by the different similarity measures. Finally, the intention of this work is to provide some insights regarding the selection of the most suitable compression algorithm for AIS data while in our previous work we focused on the effect of compression in trajectory similarity and classification problems.

2.6 Real-time Stream Processing

Real-time stream processing, as a computer programming paradigm, tries to tackle similar challenges with other big data management systems. Stream processing fulfils all four "V" challenges (volume, velocity, variety, lack of veracity) as well as the challenge of collecting, aggregating and processing data from multiple distributed sources (e.g. multiple sensors). Research on stream processing is gaining attention as a topic of the Big Data research field. As a computational paradigm, it has been established in a wide range of applications, from data processing in Web environments [98], over logistics [99] to the health sector [100]. Real-time reports on state changes of a system and its environment, thereby enabling reactive and proactive computing.

Several state of the art big data or stream processing tools include Akka, Storm[9], Spark Streaming[10], Apex[11], Flink[12] and Kafka[13]. Most of these systems are able to achieve results in real-time, over high-velocity data streams by prioritising new data over delayed streams, or by storing older data in a storage means. Multiple sources producing data, stress test the existing systems even more, thus a redesign of the system architecture is required in order to deal with latency and ac-

[9] https://storm.apache.org/
[10] https://spark.apache.org/streaming/
[11] https://apex.apache.org/
[12] https://flink.apache.org/
[13] https://kafka.apache.org/

curacy restrictions existent in stream processing applications. Novel system architectures rely on a well-known architecture, called Lambda architecture [101], a data-processing architecture designed to handle voluminous data. The Lambda architecture consists of three layers, the batch layer for batch processing, the speed layer for real-time processing and the serving layer for query responses. The batch layer is responsible for pre-computing results of queries on big data, typically stored in a database management system. The speed layer minimises latency and is responsible for filling the "gap" due to the batch layer's lag. Finally, the serving layer stores the output from the batch and the speed layer and displays the information to the end users.

Part II
Rule-based anomaly detection

3 Detecting vessel spoofing in streams of AIS data

3.1 Introduction

Unlike in the past, when maritime surveillance had suffered from a lack of data, systems are now producing unprecedented amounts of data. These systems can be classified into two broad categories: i) cooperative self-reporting systems where vessels broadcast information regarding their identity and location [these include Automatic Identification System (AIS), Long Range Identification and Tracking (LRIT) and Vessel Monitoring System (VMS)] and ii) non-cooperative surveillance systems which detect, track and/or identify vessels without any cooperation from the vessels itself e.g., radar [102]. The most commonly used is AIS, a collaborative, self-reporting system which allows vessels to periodically broadcast their identification information, characteristics, navigation, and locational data, which is then received by other vessels and coastal or satellite receiving stations. In 2002, the IMO SOLAS agreement made it compulsory for all vessels over 299 Gross Tones to carry an AIS transponder on board for safety reasons. Soon AIS was used for global scale vessel tracking, with many much smaller vessels optionally reporting their positions.

Today AIS is considered a reliable and trustworthy source of information by numerous maritime stakeholders, including port authorities and coast guard agencies across the world. Using this data, several websites provide information free of cost to the general public (e.g. marinetraffic.com). Such AIS based systems provide an almost global depiction of maritime transportation (often referred to as white shipping) in real time. These systems are constantly flooded by endless streaming data generated from millions of distributed on board sensors at rates of numerous gigabytes per day.

Unfortunately, as the AIS was not designed with security as a primary requirement, it is subject to malicious attacks, which attempt to falsify positional or identification data. These attacks aim at the integrity and reliability of the AIS data, so that it is modified, fabricated, or interrupted (loss). As a result of these attacks, stakeholders have often been deceived to believe that vessels may be conducting illegal activity when they are not, or illegal activity has been made to appear legitimate or been concealed altogether. This evolution of the ancient stratagem "poisoning the well" involves corrupting stored information by maliciously injecting false data.

On the occasion of a serious incident, popular Big Data approaches are used to analyze petabytes of historical data and shed light on what took place at a specific time and location [4, 5]. While for many other domains retrospective batch processing may be sufficient, for a wide range of real world applications, this is simply too late. For applications such as navigation, surveillance etc., timeliness is a top priority; making the right decision regarding steering a vessel away from danger, or deploying a search and rescue mission, is only useful if it is a decision made in due time. For these fast changing, massive, potentially infinite sequences of real-time data, batch processing for proactive and real-time decision making is not really an option. Unfortunately, current state of the art techniques and technologies are incapable of dealing with these growing volumes of high-speed, loosely structured, spatio-temporal data streams, that require real-time analysis in order to achieve rapid response times. For this reason, data streams research and their correlation is gaining attention as a subset of the more generic "Big Data" research field.

In this chapter we focus on detecting data spoofing and falsification attacks in real-time. We propose a distributed architecture capable of handling firehoses of data and countering "Stream Poisoning" in real time. We focus on the industrial use case of detecting spoofing events in stream of AIS and validate our approach under real world conditions. The rest of the chapter is organized as follows: Section 3.2 describes our approach, while Section 3.3 presents the preliminary results. Section 3.4 concludes this chapter by briefly outlining the main contributions of this work and suggesting future improvements.

3.2 Approach

The main objective of our work is to explore the possibility of detecting spoofing events and trajectories with a distributed method on high velocity streams of data. In short, our approach needs to:

- be able to handle large volumes of data with high velocity, or bursty event rates by distributing the load across several machines, and
- maintain high accuracy rate and detection performance in real world conditions, such as detecting spoofing events from an incoming stream of AIS messages received by a transceiver.

In the following section we describe our approach, which is designed for maritime data and architecture set up, before evaluating our results.

3.2.1 Spoofing Detection Algorithm

To tackle the problem described in Section 2.1 we employ an approach that calculates the average speed required to move, following the shortest path, between two consecutive positions reported through AIS. This is achieved by calculating first the Haversine distance for each pair of positions and then and the time needed to cover that distance based on the timestamps (ingestion time) recorded by the AIS receivers. If the calculated speed is within a feasible range (i.e., the vessel's average speed cannot exceed 50 knots), then the succeeding message becomes the new last valid position for that vessel. Otherwise, the message is flagged as possible spoofing. This is independent of the vessel type, or the area in which the vessel is travelling.

Algorithm 1 below shows the algorithm that decides and flags accordingly each message in real time, either as spoofing or valid, based on the transition feasibility criterion shown in Algorithm 2. We keep track of last valid position per MMSI in a hash map structure, where the MMSI is the hash key. Algorithm 2 shows the implementation of the transition feasibility criterion, which calculates the average speed needed for a vessel to move from the position recorded in M_1 to the position recorded in M_2.

Even though we can detect possible trajectory outliers, we have no prior knowledge of what a valid trajectory is. This is often referred to as an algorithmic cold start problem, where initially, the system has no messages and no trajectories recorded. As the messages are consumed, the system waits until it detects two consecutive valid transitions (i.e., the system buffers up to two positions) before assigning the first valid last known position which will be used to detect possible spoofing events. All positions that fail to meet the transition feasibility criterion (i.e., the calculated average speed threshold), before the first last known position for a specific vessel is assigned, are marked as spoofs.

3.2.2 Architecture for data processing

The system needs to be capable of handling large volumes of data with high velocity or bursty event rates. To tackle these demands, we use an asynchronous, message passing framework, called the Akka[14] framework, which makes the distribution of workload possible.

The system architecture has been implemented with Akka, a framework for highly concurrent and distributed systems based on the Actor model. This framework is preferred over other popular distributed systems. While Apache Flink[15] is such an alternative option, with built-in protocols for load balancing and sanity checks, that has been used for real-time trajectory and mobility analysis [33], Akka offers a low-level, message-exchange interface (a less restrictive programming model)

[14] https://akka.io/
[15] https://flink.apache.org/

Algorithm 1 Spoofing Detection Algorithm

Input: List of last valid positions $lastValidPositions$, a buffer $buffer$, a position M^a
Output: List of positions F^b

function SpoofingDetection($lastValidPositions, buffer, M^a$)
    $F \leftarrow M$
    if $buffer.size < 3$ **then**
        if $isSpoofing(buffer.lastElement, M)$ **then**
            $buffer.append(M)$
            for N `in` $buffer$ **do**
                $N.spoof \leftarrow TRUE$
            $F \leftarrow buffer$
            $buffer \leftarrow List()$
            return F^b
        else
            $buffer$ += M
            return NULL
    else if $M.mmsi$ in $lastValidPositions$ **then**
        $lastValidPosition \leftarrow lastValidPositions(M.mmsi)$
        if $isSpoofing(lastValidPosition, M)$ **then**
            $F.spoof \leftarrow TRUE$
        else
            $lastValidPositions$ += $(M.mmsi \rightarrow M)$
    else
        $lastValidPositions$ += $(M.mmsi \rightarrow M)$
    return $List(F^b)$

Algorithm 2 Transition Feasibility Criterion

Input: Position one M_1, position two M_2
Output: Boolean

function isSpoofing(M_1, M_2)
    $distance \leftarrow haversineDistance(M_1, M_2)$
    $duration \leftarrow M_2.timestamp - M_1.timestamp$
    $speed \leftarrow distance/duration$
    if $speed > 50$ **then**
        return true
    else
        return false

AIS message
!AIVDM,1,1,,A,14eG;o@034o8sd<L9i:a;WF>062D,0*7D

Table 1: Example of an AIS message.

making it ideal for flexible and customizable approaches. Our proposed distributed architecture consists of three types of nodes: one Master node, n Worker nodes and a Consensus node. Every AIS message is received from the Master node and is distributed to the worker nodes using a load balancer solution.

Master node. The Master is responsible for receiving timestamped AIS messages in the form of "timestamp, AIS message" tuples (as recorded by AIS stations), decoding them, and passing them to the Worker nodes. The default format of the AIS messages is the NMEA message format. An example of the AIS message format can be seen in Table 1. In the NMEA encoding each character corresponds to 6 binary bits, unlike ASCII, which uses 8 bits. This encoded string is converted to a sequence of ones and zeros. The sequence can then be divided into sets of strings, each set corresponding to a different piece of information, such as vessel MMSI, recorded speed or heading. At first, the master node receives a stream of raw messages with "timestamp, AIS message" tuples from a source (e.g., AIS transceiver) in the NMEA standard format and decodes them. Then, it must select a worker node to forward the message to. This is accomplished through a routing table maintained in the Master node that contains state information about the MMSIs associated with the Worker nodes and a Hash Map – hereafter "Balancer" – that contains processing status information for each Worker (i.e., number of trajectories processed). In case the MMSI has already been routed to a Worker based on the routing table, the Master will immediately send the message to the corresponding Worker node. Otherwise, the Balancer lets the Master know which Worker has processed the least number of trajectories, defined by the number of MMSIs processed (each MMSI corresponds to a trajectory). The Master sends the message to that node and updates the routing table. Finally, the Master node periodically sends messages to the Consensus node, the role of which will be explained in the following paragraphs. These messages contain information about how many messages have been decoded and routed by the Master.

Worker node. Each Worker is independent and responsible for a distinct set of trajectories. To detect possible spoofing, it employs the detection algorithm described in Section 3.2.1. The output of each worker is a stream of decoded AIS messages containing a flag indicating whether a message has been spoofed or not. Finally, Worker nodes, similarly to the Master node, send periodically messages to the Consensus node which contain the number of messages processed by each worker.

Consensus node. This node is responsible for collecting information from the other nodes and measure the reliability and the performance of the system. In a distributed system it is often hard to evaluate its reliability; therefore, we have adopted a naïve approach with checkpoints to perform sanity checks periodically (i.e., once every x messages). At each checkpoint, the Consensus node receives information from the Master node and the Worker nodes about the total number of messages sent from the Master to the Workers and the number of messages processed cumulatively by the workers. If the two numbers are equal, we consider our system to be reliable. Furthermore, the Consensus node collects data about spoofing events and message consumption rates from the Worker nodes and evaluates our system in terms of performance, by comparing the number of messages processed between each pair of successive checkpoints and calculating the system's throughput.

Figure 4 illustrates the overview of the system architecture with four worker nodes. The master receives raw AIS messages from the transceiver and sends decoded messages to the workers. Every x messages, both the workers and the master send the number of events processed to the Consensus node for sanity check and throughput calculation.

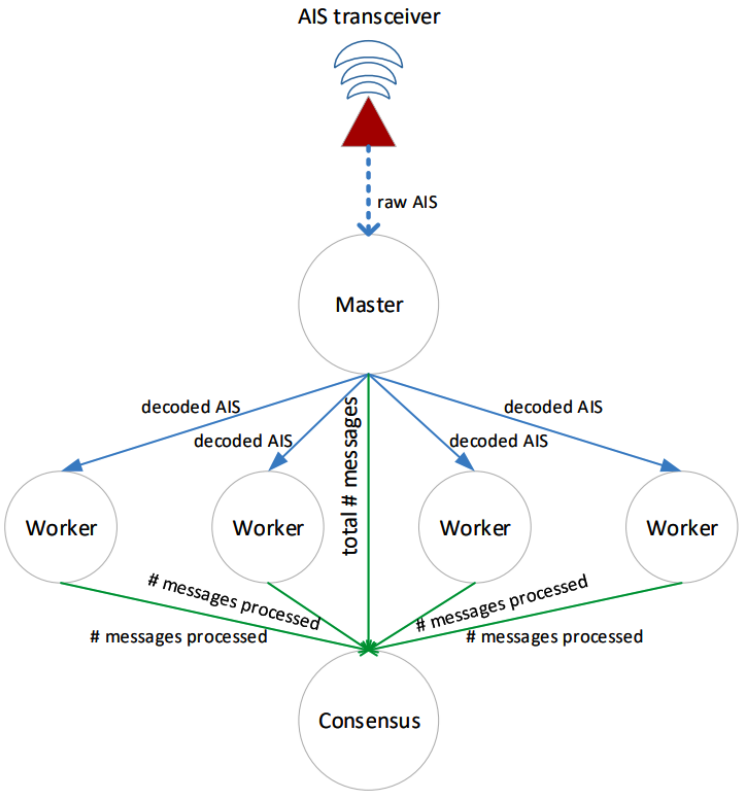


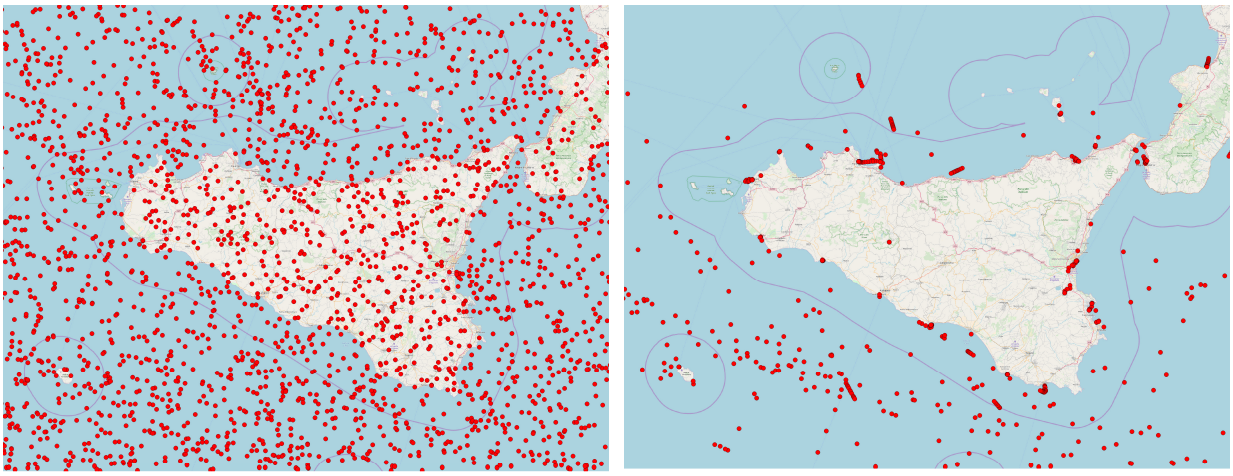
Figure 4: System architecture overview.

3.3 Experiments

3.3.1 Dataset Description

The dataset that was used was provided by MarineTraffic[16] and contains $43,912,236$ AIS messages received from $28,961$ vessels sailing in the Mediterranean in May 2016. The total size of the dataset is 2.4GB. At this point it should be noted that the Search and Rescue (SAR) aircrafts, which can have speeds greater than the max speed of a vessel, are considered outliers and thus have been removed from the dataset. To evaluate our approach, we artificially induced noise into the dataset in two different ways. First, we defined a random selection function that decides randomly in real time if a message will be replaced by noise or not. This was achieved by picking numbers out of a normal distribution and comparing them to a threshold that corresponds to the noise level that we target at each iteration (we demonstrate results for five different noise levels in the following section). In case that the corresponding message is selected to be replaced, we sample the pair of artificial coordinates (i.e. longitude and latitude) out of their corresponding normal distributions that are projected to cover the Mediterranean area. In addition, we allow a cold start for our spoofing detection algorithm by replacing messages only after our algorithm has detected the minimum number of consecutive valid messages for each MMSI (i.e., three messages corresponding to two valid transitions). Secondly, we aimed to synthetically create a worst-case scenario dataset, by introducing entire spoofing trajectories into valid ones. Such spoofing is far more complex to detect compared to sole signals. To generate such a dataset, we initially filtered from the dataset all the positional data of MMSIs with less than 30 messages transmitted during the whole interval (i.e., 1 month). Then, we transformed the filtered data into noise, by changing the MMSI of each entry with an MMSI coming from the rest of the data (i.e. MMSIs that have more than 30 transmissions during the considered time interval). Finally, we merged the noise data to the rest of the dataset. In both cases the noise ratio is approximately close to noise observed in real systems.

[16] https://www.marinetraffic.com/



(a) Induced noise by randomly changing coordinates of AIS messages.

(b) Induced noise by changing MMSIs.

Figure 5: Examples of induced noise in the dataset.

Average noise ratio	min F1-score	average F1-score	max F1-score
0.2272597	91.93045	92.88076	93.40959
1.1362555	90.29008	91.87921	92.42006
2.2732047	91.44832	91.71793	92.11491
4.5471772	90.99060	91.11057	91.36623
6.8167651	90.2043	90.46079	90.7442

Table 2: Spoofing detection algorithm performance evaluation.

Figure 5(a) and Fig. 5(b) illustrate the artificially induced noise in the area of Sicily, Italy. Red dots denote AIS messages which correspond to spoofing events. Figure 5(a) illustrates the noise induced using the first method, which most resembles the spoofing events occurring in a real-world setting, while Figure 5(b) demonstrates the noise induced by changing the MMSIs, which is a worst-case scenario, not so commonly seen in real world cases. It is observed from those figures that the second scenario contains less spoofing events, which is normal considering the AIS transmission frequencies and the strict threshold of 30 messages for a whole month.

3.3.2 Experimental Evaluation

The experimental evaluation presented below was performed on a machine with 2 cores (4 logical processors), an AMD A10-7870K Radeon R7 CPU, and 8 GB of RAM, running Windows 10 with Java OpenJDK 1.8, Scala 2.11.7 and Akka 2.5.1.

When evaluating the first scenario, we performed five rounds of experiments, each one introducing a different level of noise to the dataset. To evaluate our methodology, we measured the number of spoofing messages the system was able to detect (true positives), the number of messages that were falsely detected as spoofing (false positives) and the number of spoofing messages that were falsely ignored and classified as valid (false negatives) for each dataset using two Worker nodes. Each round of experiments was executed multiple times (i.e., 5 times) and the average values of those measurements were calculated. Table 2 shows the results from these experiments. The first column highlights the average noise to total messages ratio of each round. The noise ratio ranges from 0.22% to 6.81%. The rest of the table shows the minimum, maximum and average values of f1-score the algorithm managed to achieve in each case.

Figure 6 is a visual illustration of Table 2. Moreover, it shows the precision and the recall of

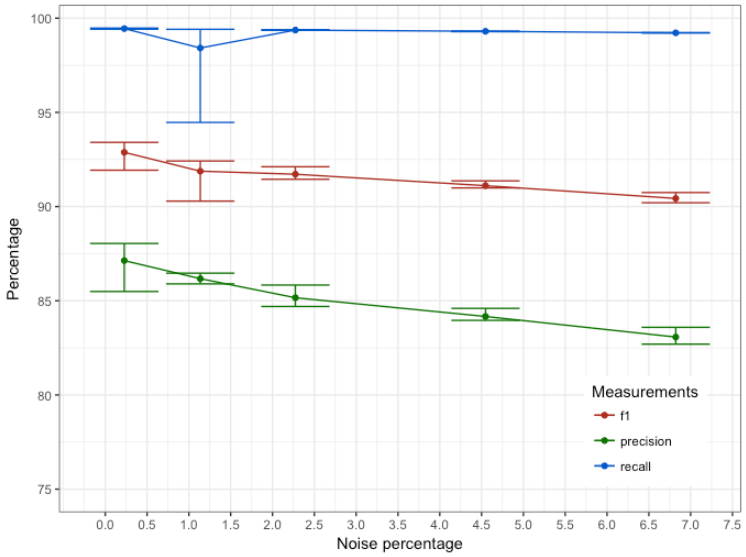


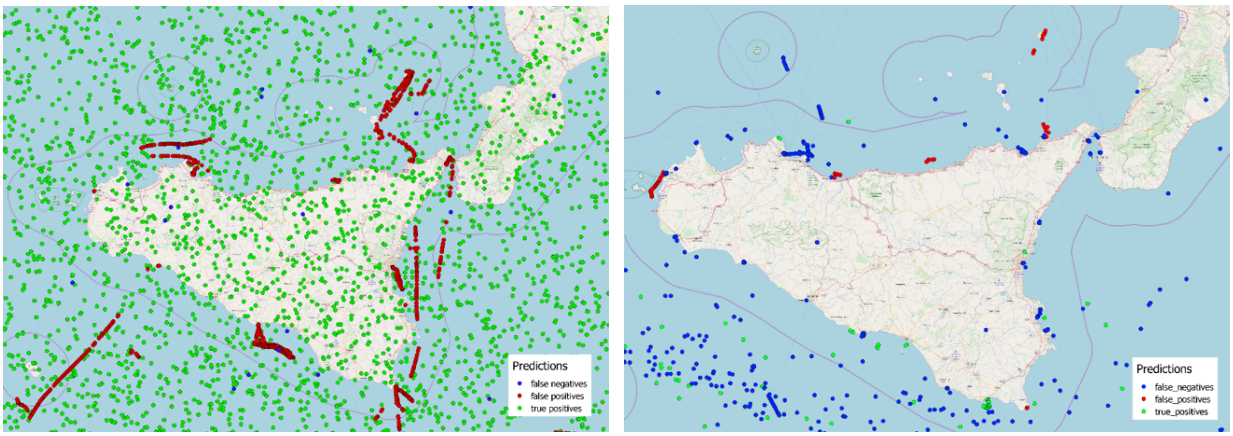
Figure 6: Average performance, measuring precision, recall and f1-score for the spoofing algorithm.

the algorithm. From Figure 6 we can see that as the noise increases the f1-score of our solution decreases. In cases where there is increased noise in the stream, and since in the beginning the system had no prior knowledge about a valid trajectory, it was difficult to distinguish the real trajectories from the fake ones, and thus the false positive/negative rate increased. A visual illustration on the map is given by Figure 7(a). It was observed that several spoofing events were not detected (false negatives, blue dots), while others were falsely detected as spoofing events (false positives). False positives can easily be seen by comparing Figure 5(a) and Figure 7(a). Several trajectories (dense red dots that form trajectories around Sicily) can be seen in Figure 7(a), while in Figure 5(a) they are missing. This means that an entire trajectory was falsely identified as spoofing.

In extreme cases, such as when two vessels use the same MMSI, either for a long period of time, or share close geographic proximity, a fake trajectory can be falsely classified as valid, which leads to an increase of false positive/negative detections. This is mainly because the system has no knowledge about a valid trajectory in the beginning, and thus the first seen trajectory is classified as valid. This case is represented by the second type of noise that was created (i.e., the second scenario). The f1-score achieved by the algorithm in the second scenario was 22.43%, while the precision was 41% and the recall was 15.3%. Figure 7(b) visualizes the second scenario, which is an extreme case, and compares the true positives with false positives/negatives. Comparing Figure 5(b) and Figure 7(b), we can see that the algorithm detected as spoofing events trajectories that were not. Also, in the second scenario we had an increased number of false negatives (messages that were falsely classified as valid), which is why the recall was so low.

3.4 Summary

In this chapter, we presented a distributed architecture for real-time spoofing detection. We demonstrated its use in a real-world application and showed its increased performance in terms of accuracy, using a varying number of noise levels injected artificially in the stream.



(a) Spoofing events vs Detected events in the first scenario. Green dots are the correctly identified noise messages, blue dots are not detected noise (false negatives) and red dots are missclassified valid trajectories (false positives).

(b) Spoofing events vs Detected events in the second scenario. Green dots are the correctly identified noise messages, blue dots are not detected noise (false negatives) and red dots are missclassified valid trajectories (false positives).

Figure 7: Visual illustration of spoofing detection events.

4 Detecting intentional AIS switch-off

4.1 Introduction

Unlike in the past, when maritime surveillance had suffered from a lack of data, systems are now producing unprecedented amounts of data. These systems can be classified into two broad categories: i) cooperative self-reporting systems where vessels broadcast information regarding their identity and location [these include Automatic Identification System (AIS), Long Range Identification and Tracking (LRIT) and Vessel Monitoring System (VMS)] and ii) non-cooperative surveillance systems which detect, track and/or identify vessels without any cooperation from the vessels itself e.g., radar [102]. The most commonly used is AIS, a collaborative, self-reporting system which allows vessels to periodically broadcast their identification information, characteristics, navigation, and locational data, which is then received by other vessels and coastal or satellite receiving stations. In 2002, the IMO SOLAS agreement made it compulsory for all vessels over 299 Gross Tones to carry an AIS transponder on board for safety reasons. Soon AIS was used for global scale vessel tracking, with many much smaller vessels optionally reporting their positions.

Today AIS is considered a reliable and trustworthy source of information by numerous maritime stakeholders, including port authorities and coast guard agencies across the world. Using this data, several websites provide information free of cost to the general public (e.g. marinetraffic.com). Such AIS based systems provide an almost global depiction of maritime transportation (often referred to as white shipping) in real time. These systems are constantly flooded by endless streaming data generated from millions of distributed on board sensors at rates of numerous gigabytes per day.

Although it is mandatory for vessels to carry an AIS transponder, it is not compulsory for the transponder to be switched on. As a result, many vessels "go dark" by switching off their transponder so as to hide potential illegal activity from the authorities or to hide their whereabouts. Regarding the latter case, cargo or tanker vessels hide their location in order to avoid piracy attacks and fishing vessels do so in order to prevent other fishing vessels to fish in the same waters. These incidents seem as a communication gap in the tracking data of vessels, but it is not clear whether it comes from AIS switch-off or not. Due to bad weather conditions or lack of shore-based receiving stations in an area under surveillance (e.g. in oceans), the network coverage might be limited,

and thus positional data transmitted might not be received from the network. The significance of distinguishing AIS switch-offs can be seen in a few examples. Global Fishing Watch, which is an independent, international non-profit organisation, highlights the importance of maritime tracking data in keeping fisheries around the world healthy [103]. For instance, a vessel departed from a port, travelled to the edge of the Exclusive Economic Zone (EEZ) – an area where a coastal nation has jurisdiction over natural resources – and turned off its AIS for 24 days [104]. When the vessel turned on again its transponder and started sending AIS messages, it seemed to be fishing shortly before going back to the port it departed from. During that time the vessel could be fishing inside or outside the EEZ. In another example, vessels near Dutch Harbour, Alaska, tend to turn off their AIS transponder when they depart from port and remain hidden until they return to the port [105]. Although, Alaska fisheries are closely monitored, vessels turn off their AIS to conceal their exact location from other fishing vessels that are competing for the same catch. On another case, the Jerusalem Post, a famous Israeli news website, revealed in an article that oil tankers cloak their tracks by switching off their AIS when departing from Iran ports or when arriving into Syria ports in order to conceal the oil exports from Iran.

On the occasion of a serious incident, popular Big Data approaches are used to analyse petabytes of historical data and shed light on what took place at a specific place and time [4, 5]. While for many other domains retrospective batch processing is sufficient, for a wide range of real-world applications, this is simply too slow. For applications such as navigation, surveillance etc., timeliness is a top priority; making the right decision regarding steering a vessel away from danger, or deploying a search and rescue mission, is only useful if the decision was made in due time. Unfortunately, current state of the art techniques and technologies are incapable of dealing with these growing volumes of high-speed, loosely structured, spatio-temporal data streams, in real-time in order to achieve rapid response times. For this reason, data streams research and its correlation is gaining attention as a subdomain of the more generic "Big Data" research field.

To tackle the problem of AIS switch-off, we developed a maritime surveillance system which notifies the user for communication gaps in real-time and distinguishes AIS switch-off from network coverage issues. The system needs to be able to handle the large volume of data. This is handled efficiently by the designed system architecture and implementation. To the best of our knowledge, no previous work has focused on detecting anomalies regarding AIS switch-off in real-time.

The work that is more relevant to our approach is the one presented in [9], in which a distributed real-time approach to detect spoofing identity events was introduced. Our contribution extends that functionality and can offer detection of one more type of anomaly (i.e., AIS switch-off) that can be seen in real world maritime use cases. Furthermore, the system presented in [9] was applied in specific geographic region and was not intended to be applied at global scale. On the contrary, our work takes full advantage of the lightweight Akka framework, which is an open-source framework ideal for developing concurrent and distributed systems that can scale up, using the resources of one or more computers efficiently. Furthermore, Akka offers load balancing and adaptive routing across multiple nodes. In contrast to the implementation of [9], our system is not limited to creating one thread (actor) per CPU.

The remainder of this chapter is organised as follows. Section 4.2 describes the problem formulation. Section 4.3 and 4.4 present the algorithmic approach and the designed architecture respectively. Experimental results on a real-time, large scale dataset are presented in Section 4.5. Finally, in Section 4.6, we summarise our approach and discuss steps for future work.

4.2 Problem Formulation

In this section, we define the basic concepts and terminology used in this chapter and we define the problem of identifying the potential cause of AIS switch-off.

Trajectory. Let D be a spatio-temporal data set that consists of temporally sorted points $p \in D$, where each point p has the following attributes:

- Timestamp: an integer that denotes the number of seconds that have elapsed since January 1, 1970 (e.g., epoch time).
- Longitude: the horizontal position on earth's surface.
- Latitude: the vertical position on earth's surface.
- Speed: the speed of the vessel measured in knots.
- Course over ground (cog): vessel's course relative to true north with up to 0.1° accuracy ($cog \in [0, 359]$).

Communication gap or gap. AIS communication protocol defines the transmission frequency of a transponder which is affected by the vessel's speed and cog changes. "Gap" is defined as the absence of points p from D for an unjustified time period t (i.e., positions are not received but based on the last known location, speed and cog the vessel would be expected to send messages within t).

Network coverage. Let C be a set of squares (also called cells) $c \in C$, where each cell c has the following attributes:

- Minimum longitude: the longitude of the cell's lower left corner.
- Minimum latitude: the latitude of the cell's lower left corner.
- Size: the size of each cell's side in degrees.

Together, these attributes are considered to be the spatial index i of the cell c. Each cell contains a set A of points, such that $A \subseteq D$. Given a time period $[T_{start} - T_{end}]$ a cell c is characterised as a "dark cell" if and only if $A_c = \oslash$ for $t \in [T_{start} - T_{end}]$, which is a user defined parameter. "Lack of coverage" loc is defined as the set of all the dark cells and can be expressed as a set of cells B, such that $B \subseteq C$ and for each cell $c \in B$, $A_c = \oslash$.

Projection. The projection P of the vessel is defined as the destination point given a distance and bearing from the vessel's initial location or point p. To calculate the projection, based on the polar coordinate system[17], we use the following formulas:

$$P_{latitude} = \alpha \sin(\sin p_{latitude} \times \cos\delta + \cos p_{latitude} \times \sin\delta \times \cos\theta) \quad (1)$$

and

$$P_{longitude} = p_{longitude} + \text{atan2}(\sin\theta \times \sin\delta \times \cos p_{latitude}, \cos\delta - \sin p_{latitude} \times \sin P_{latitude}) \quad (2)$$

where θ is the bearing, δ is the angular distance d/R; d being the distance travelled, R the earth's radius. To calculate d we take into account the vessel's speed and the time period t it has not emitted any AIS messages.

The problem of AIS switch-off addressed in this work is to identify "gaps" that are due to "dark cells" or vessels that switched off their AIS transponders (e.g., stop transmitting AIS messages for a predefined period of time t).

[17] A two-dimensional coordinate system in which each point on a plane is determined by a distance from a reference point and an angle from a reference direction.

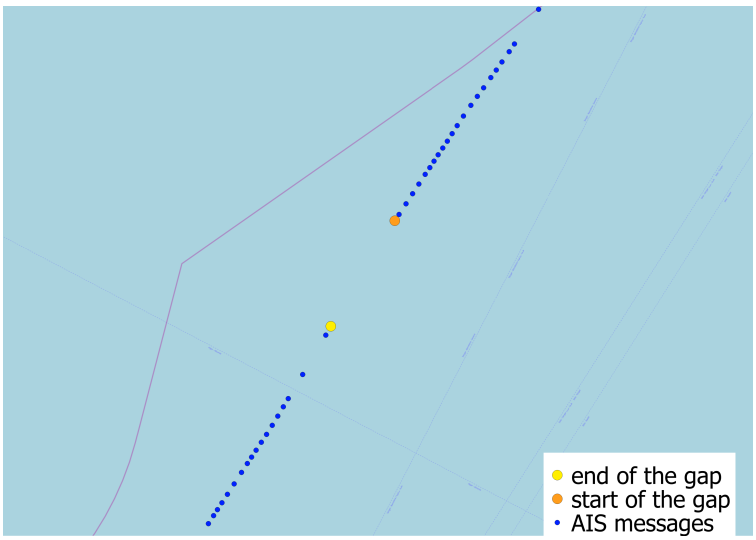


Figure 8: Example of AIS switch-off.

Problem 1 *Given a spatio-temporal data set D and a network coverage C, find the projection P, such that $P \cap d_c$ and $d_c \in B$, where d_c is a dark cell.*

To visualise the formulated problem, consider Figure 8 which illustrates an example of an AIS switch-off in the Adriatic Sea, between Lecce, Italy and Albania. The vessel has been traveling from north to south in a narrow bounding box with good network coverage (network coverage with no dark cells) and it has switched off its AIS transponder. The orange point indicates the last message received from the vessel and thus, it is the start of the communication gap. The yellow point indicates the first message received from the vessel after the communication gap and it is considered to be the end of it.

4.3 Approach to detecting AIS switch-off

To tackle the problem of AIS switch-off we introduce an algorithm that relies on time outs and a given network coverage, which is the surveillance area. Inside the network coverage we receive AIS messages. The network coverage is then divided into equally sized cells, in which we receive a varying number of AIS messages. We can take advantage of this information by detecting whether an area has coverage by the receivers or not. Lack of coverage in an area (e.g., dark cells) is due to weather conditions, lack of terrestrial base stations or due to the fact that receivers could be jammed deliberately to reduce their coverage. Out of these cases only the lack of terrestrial base stations is known a priori, while the other two cases can be discovered in real-time by the number of AIS messages per cell. Due to the fact that the dark cells of the network coverage change over time – dark cells are not "dark" anymore and vice versa – this problem is even harder. The algorithm uses the network coverage to infer whether communication gaps are due to AIS switch-off caused by the vessels themselves or due to the lack of AIS messages in a cell. Positional information of vessels is distributed to the corresponding actors of the Akka system. Each actor is responsible for monitoring a single vessel and always stores the last known position of the vessel along with its speed over ground and its course over ground. A vessel transmits messages with rates that range from once every 2 seconds up to once every 3 minutes, depending on its speed. In our approach when the actor has not received a message for X minutes (where X equals 10, 30, 60 or 120), which is a relatively relaxed switch-off threshold with respect to the vessel's transmission rates, it evaluates whether the vessel is sailing in a dark cell or has its AIS transponder switched. Specifically, when the time threshold expires, the actor sends a time out message to itself. When the message is received, a possible position based on the last known position, speed and heading at the current timestamp

– projection – is calculated. Then, if the projection is not inside a dark cell within the network coverage, the last known position is flagged as the start of an AIS switch-off. Otherwise, the last known position is flagged as the start of an AIS communication gap. Algorithm 3 below shows the steps that detect AIS switch-off.

Algorithm 3 AIS switch-off algorithm

Input: The last position received $lastPosition$, the time threshold $timeThreshold$, the network coverage NC

Output: Boolean

```
1: function isSwitchOff(lastPosition, timeThreshold, NC)
2:     projection ← predict(lastPosition, timeThreshold)
3:     cell ← getEnclosingCell(projection)
4:     if cell ≠ cell_dark then
5:         return true
6:     else
7:         return false
```

Firstly, we predict the location of the vessel at the point of the time out threshold – projection, (line 2). To do this, we use the speed over ground, course over ground and position from the last known AIS message. Then, we calculate vessel's projection the moment that the time threshold expired by applying the formulas defined in Section 4.2. Then, we find the enclosing cell in which the projection is located (line 3). If the enclosing cell is not a dark cell, then we assume that the vessel switched off its AIS transponder (line 5), otherwise it is due to the lack of network coverage (line 7).

To find the enclosing cell of the projection, we introduce an algorithm, Algorithm 4, that indexes a vessel position to a longitude and a latitude, which correspond to the coordinates of the lower left corner of the enclosing cell in the network coverage. We assume that the coordinates have at most six decimal digits, hence we define the Decimal Number Size (DNS) to 1000000. For that reason, we multiply the coordinates of the position with the DNS and keep only the integer part of the decimal number (lines 3 and 4). Let C_{start} be the first cell of the network coverage, e.g. the reference point of the network coverage. Again, we multiply its coordinates with the DNS. Let $cell_{step}$ be the size of each cell or alternatively the distance required to move from one cell to the next. Since, the cells are square-shaped, the step for both x and y axis is the same. Similarly, we assume that the step has at most six decimal digits and multiply it by the DNS. Then, the difference between the longitude/latitude and the reference point is found and the modulo operation between the difference and the step is computed in order to find the number of steps required to reach the coordinates from the reference point. By subtracting the resulting value from the longitude/latitude, the coordinates of the enclosing cell are found (lines 8 and 9). Finally, we convert the coordinates back to decimal numbers with a 6-digit decimal part (lines 10 and 11).

4.4 Architecture for Data Processing

The system needs to be able to handle large volumes of data in the stream in a global scale, withstand the high velocity/bursty event rates of the stream and in the meantime, it needs to react upon events instantly. The architecture builds upon our previous work [9] which uses the Akka framework and takes full advantage of its lightweight components, called actors. Actors have a small memory footprint – 2.5 million actors per GB of heap – which makes it possible to create many actor instances on a single machine. Although Flink outperforms other streaming engines such Spark Streaming and Storm [106], we selected Akka due to its configurability, since Akka is a fun-

Algorithm 4 Spatial indexing

Input: The position $position$
Output: The coordinates of the enclosing cell

1: **function** getEnclosingCell($position$)
2: $DNS \leftarrow 1000000$
3: $x \leftarrow position_{longitude} \times DNS$
4: $y \leftarrow position_{latitude} \times DNS$
5: $xStart \leftarrow C_{start_{longitude}} \times DNS$
6: $yStart \leftarrow C_{start_{latitude}} \times DNS$
7: $step \leftarrow cell_{step} \times DNS$
8: $llx \leftarrow x - ((x - xStart) \mod step)$
9: $lly \leftarrow y - ((y - xStart) \mod step)$
10: $lowLeftX \leftarrow llx/DNS)$
11: $lowLeftY \leftarrow lly/DNS)$
12: **return** ($lowLeftX$,$lowLeftY$)

damental part of Flink's underlying computation engine and allows more customisable and flexible approaches. The main components of the system are the consumer actor, n coordinator actors and m worker actors. Our work here extends the architecture proposed in [9] by using coordinator actors, the use of which is explained in the next paragraphs.

Consumer Node. This node is responsible for consuming messages from a Kafka topic, which contains decoded AIS messages, and distributing the messages to the coordinators. The consumer node handles the load balancing of the messages. Each coordinator must receive and process the same number of ship ids. The consumer node tracks the number of ship ids received per coordinator and sends the next message with a new ship id to the coordinator with the least received number of ids.

Coordinator. There is one coordinator per machine in the cluster. Each coordinator receives messages from the consumer node and is responsible for creating new worker actors and routing the messages to the existing ones. When the coordinator receives a message which contains a new ship id, it creates its corresponding actor instance and from that point on it routes all the new messages with that ship id to this actor. Otherwise, if the message received contains the same ship id with a previously consumed message, then it is routed to the worker actor that is responsible for the ship id.

Worker. Each worker is responsible for one ship id. This node receives messages from a ship and applies the algorithm described in Section 4.3. When it detects an AIS switch-off, it checks if it is due to the network coverage or not. Then, the worker produces an anomaly event and registers it in a Kafka topic that contains all the detected anomalies from all workers. Using one worker actor per ship id makes the architecture robust and resilient to reception errors, as each worker analyses the track of a single vessel making thus the worker isolated from the analysis performed in other workers. Furthermore, since each actor instance is lightweight, the creation of thousands of actors poses limited to no overhead in the system. Figure 9 below visualises the system architecture.

4.5 Experimental Evaluation

For the experimental evaluation, MarineTraffic provided a dataset which contains $43,912,236$ AIS messages received from $28,961$ vessels sailing in the Mediterranean in May 2016. To evaluate our approach, we artificially created gaps of different time thresholds (ranging from 10 minutes up to 2 hours) in the vessels' trajectories (gaps were created in approximately $20,000$ trajectories

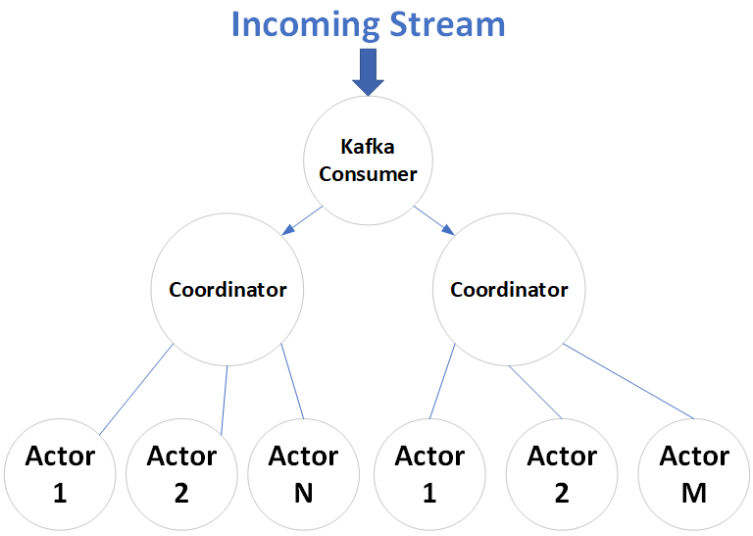


Figure 9: System Architecture: The consumer actor receives the decoded stream from the Kafka topic, sends the messages to the coordinators and the coordinators forward the messages to the corresponding worker actors. Each machine in the cluster has one coordinator and each machine may have a large number of worker actors.

depending on the time threshold). Furthermore, we generated a network coverage, extracted by the AIS messages of the area under surveillance, which consists of equally sized cells of 400 square kilometres each. Then, we randomly annotated 5%, 10%, 15% and 20% of the total cells in the network coverage as dark cells which indicate that we do not have coverage by the receivers and we removed positional data located inside these cells. Cells that are annotated as dark cells have low traffic of vessels in order to simulate a real-world scenario. In real-world conditions these cells might be dark or out-of-coverage either due to bad weather conditions or due to lack of terrestrial AIS base stations. A 5% lack of coverage is a realistic scenario in areas where there are terrestrial AIS base stations but the receivers are either deliberately jammed or "blocked" due to weather conditions. A 20% lack of coverage is a worst-case scenario in areas where there are few base stations and the receivers are either jammed or "blocked". Because the AIS messages inside the dark cells were discarded, the remaining messages of the vessels located just before entering the dark cells are annotated as "gap starts" due to limited coverage. Note that these messages are annotated as gap starts only when the time delta between the message received just before the vessel entered the dark cell and the one received just after the vessel exited the dark cell is greater than or equal to the time threshold set up in each experiment.

To simulate the AIS switch-off in the data and to evaluate the methodology when the time threshold is increased or when the percentage of dark cells is increased, we selected vessels that do not pass through any dark cells when the percentage of dark cells is 20% of the total cells (worst-case scenario) and in each experiment, regardless of the percentage of lack of coverage, we removed positional data from the trajectory after having received more than 3 AIS messages. The number of AIS messages removed from the trajectory depends on the time threshold each time (one AIS message per minute of time threshold). The last AIS message before the cut of the trajectory is annotated as gap start due to AIS switch-off. We repeated the experiments five times, each time annotating a different portion of cells as dark cells. For a visualisation, see Figure 10 which illustrates the network coverage we created. Green dots indicate the lower left corner of cells marked as "in network coverage" and thus messages from vessels inside those cells should be received and red dots indicate the lower left corner of dark cells where messages transmitted from vessels are not received. In the zoomed area of Athens, Greece we can better distinguish the cells from each other. Table 3 shows the number of gaps in each experiment.

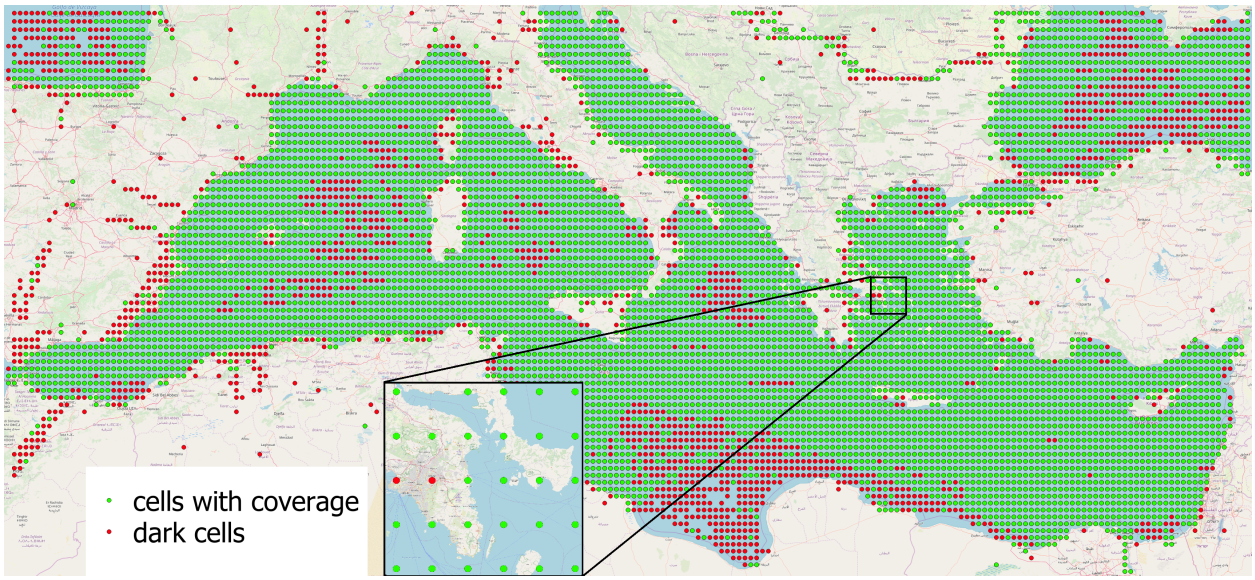


Figure 10: A worst-case scenario network coverage.

Lack of coverage	Time threshold (min)	#Switch-off gaps	#Out-of-coverage gaps
5%	10	20, 572	2, 104
	30	20, 572	1, 766
	60	20, 572	1, 050
	120	20, 572	468
10%	10	20, 572	3, 717
	30	20, 572	3, 165
	60	20, 572	2, 081
	120	20, 572	1, 073
15%	10	20, 572	4, 838
	30	20, 572	4, 120
	60	20, 572	2, 930
	120	20, 572	1, 720
20%	10	20, 572	5, 455
	30	20, 572	4, 593
	60	20, 572	3, 468
	120	20, 572	2, 220

Table 3: Averaged number of gaps per experiment.

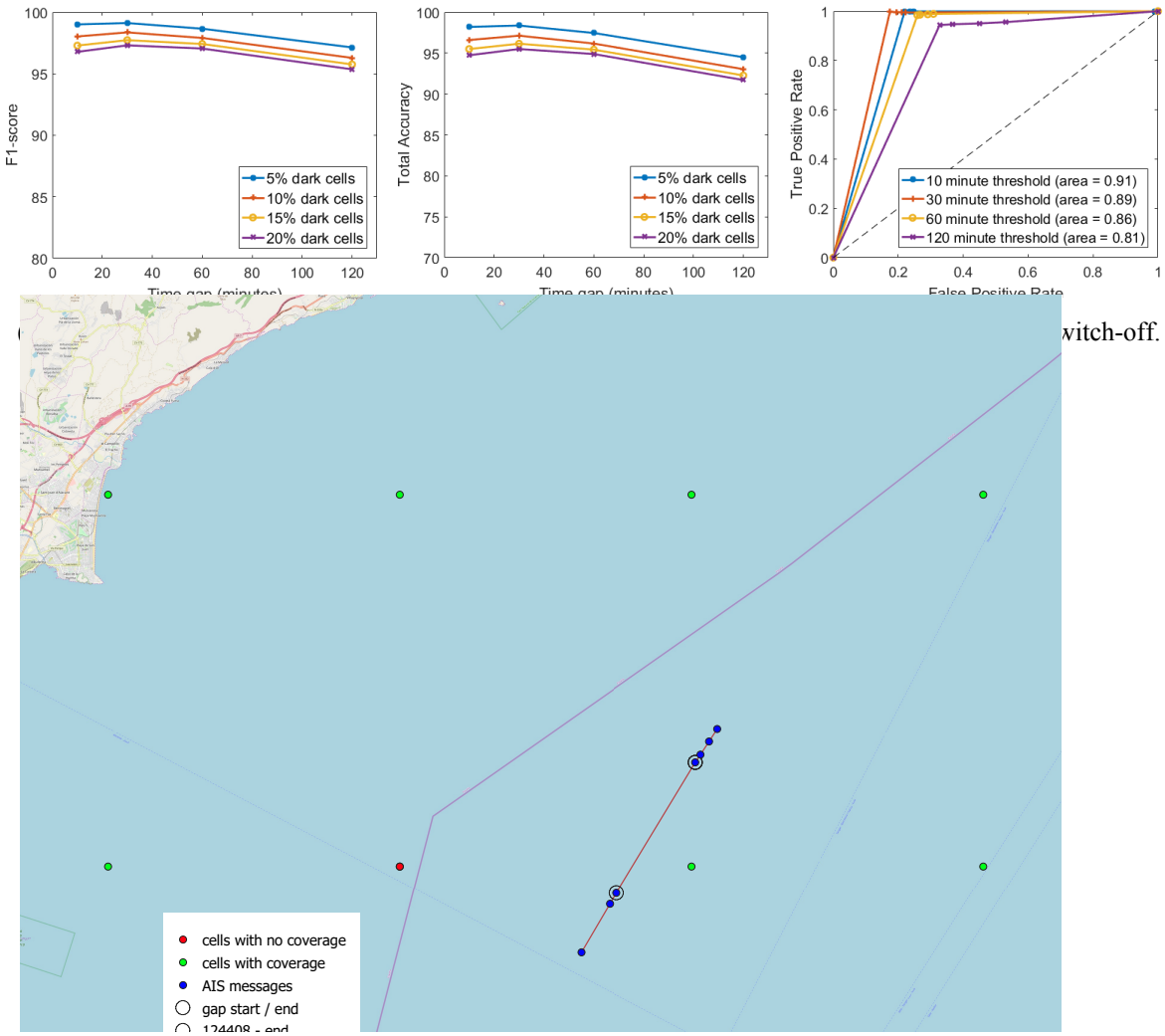
% dark cells	AIS switch-off	Precision	Recall	F1-score
5%	10 min	98.08%	99.97%	99.01%
	30 min	98.52%	99.72%	99.12%
	60 min	98.43%	98.90%	98.66%
	120 min	98.74%	95.59%	97.14%
10%	10 min	96.18%	99.96%	98.03%
	30 min	97.09%	99.67%	98.36%
	60 min	97.11%	98.72%	97.91%
	120 min	97.58%	95.04%	96.29%
15%	10 min	94.76%	99.97%	97.29%
	30 min	95.93%	99.60%	97.93%
	60 min	96.28%	98.57%	97.42%
	120 min	96.85%	94.70%	95.76%
20%	10 min	93.81%	99.94%	96.78%
	30 min	95.23%	99.48%	97.31%
	60 min	95.75%	98.39%	97.05%
	120 min	96.37%	94.36%	95.36%

Table 4: AIS switch-off performance evaluation.

The experimental evaluation was performed on a machine with 8 logical processors (4 physical ones), an Intel(R) Core^{TM} i7-7700-HQ CPU @ 2.8 GHz and 32 GB of RAM, running Windows 10 with Java OpenJDK 1.8, Scala 2.12.8 and Akka 2.5.1.

To evaluate our methodology, we measured how accurately it estimates if the gap is due to network coverage or if the AIS transponder was switched off. We tested our methodology when vessels stopped sending AIS messages for a time that ranges from 10 minutes to 2 hours and when the dark cells range from 5% to 20% of the total cells of the network coverage. In our experiments false positives are defined as out-of-coverage gaps that are falsely classified as switch-off gaps. False negatives, similarly, are predictions that falsely classified switch-off gaps as out-of-coverage gaps. Both false positives and false negatives are of equal importance in our scenario. In alerting systems such as the one we propose, if users are frequently notified for false positives, the usability is decreased due to the high rate of false alerts. Similarly, if the system does not recognise switch-off gaps and has an increased false negative rate, the usefulness of the system is decreased because there are minimal or no alerts for events of interest.

In our evaluation, we need to assess the proposed methodology in making correct predictions out of all the synthetically generated gaps and the rate of false alerts. For that reason, we take into account all of the metrics required for a classification experiment, precision, recall, f1-score and accuracy. To have better results we randomly created five different network coverages, with approximately 5%, 10%, 15% and 20% of each network coverage to be dark cells. Then, we run the experiments five times, each time with a different portion of network coverage annotated as dark cells and recorded the scores. Table 4 shows the averaged results for the AIS switch-off gaps. We notice that when we increase the time threshold, the F1-score decreases. This is due to the fact that as the time passes it is harder to predict the exact location of the vessel [107]. During the time which the vessel does not send AIS messages, the heading and the speed can change significantly, which renders impossible the correct prediction of the vessel's location. Similarly, when the percentage of dark cells is increased, the F1-score decreases. This happens because predictions fall more frequently into dark cells as the number of dark cells increases. Figure 11(a) illustrates the decrease in F1-score as the time gap increases. One interesting fact that can be seen in Figure 11(a) is that although the time threshold and the percentage of dark cells is increased, our methodology manages to maintain a high prediction rate. Any decrease in the F1-score is because we predict the future



witch-off.

Figure 12: Gap falsely detected as AIS switch-off.

location based on the last known position, speed and heading, but in fact vessels do not keep the same course during the entire time interval when we did not receive any messages. Since we predict the location in a straight line, the projection places the vessel in a cell that it is not dark, which consequently leads to a false classification of the gap.

Finally, to further visualise the performance of our approach, we measured the total accuracy which is defined as the number of correct predictions to the number of the total predictions. Figure 11(b) illustrates the total accuracy, from which we can infer that it decreases as the time threshold increases and as the percentage of the dark cells increases. Even in real-world, worst-case scenarios, our methodology achieves an accuracy of 91.71% while in a best-case scenario it achieves an accuracy of 98.2%. Moreover, we visualised the ROC curves for AIS switch-offs in Figure 11(c) in order to illustrate the correct predictions compared to the false ones as the time threshold and the percentage of the dark cells change. Based on Figure 6 we can infer that the algorithm is resilient in identifying the true switch-offs even at higher time thresholds or higher percentages of dark cells in the network coverage, in which case there will be more false positives. Two distinct examples of false predictions are Figures 12 and 13. In Figure 12 a vessel is travelling from north to south near the coast of Spain. The vessel is initially located in a cell with coverage, then passes through a dark cell and lastly enters a cell with coverage again. Although this gap is annotated as out-of-coverage, the algorithm predicts the vessel's location half an hour later inside a cell with coverage and classifies this gap as AIS switch off. Note that the next AIS message is received a few seconds after the time threshold expired.

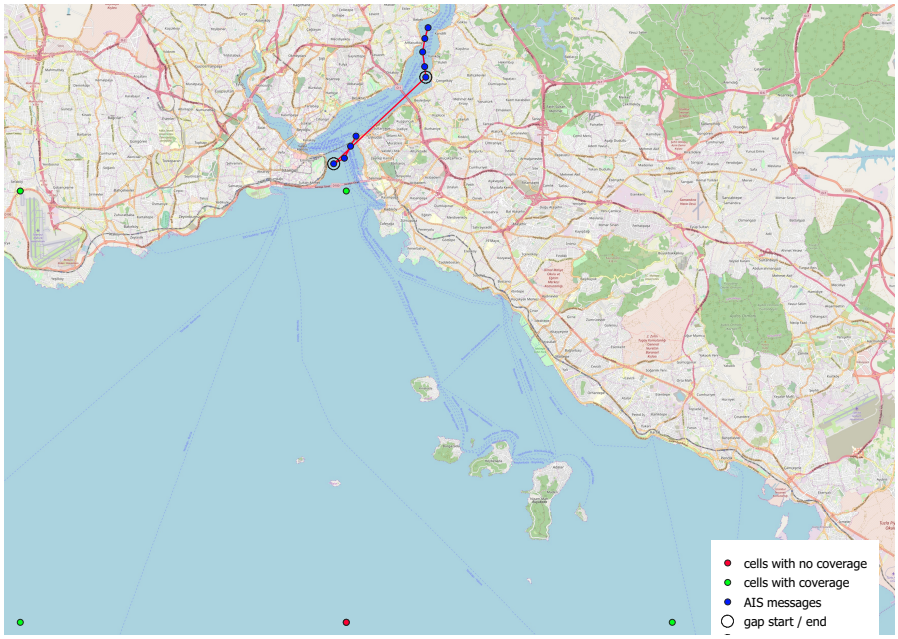


Figure 13: Gap falsely detected as a network coverage gap.

In Figure 13 a vessel is travelling from north to south near the coast of Istanbul and at some point it switches off the AIS transponder. During the entire time the vessel is located in areas with coverage. Based on the last known AIS message before the switch-off the vessel seems to be heading south, where there is a dark cell. Half an hour later, when the time threshold was exceeded, the prediction estimates that the vessel is inside the dark cell. For that reason, the algorithm misclassifies this gap as an out-of-coverage gap.

4.6 Summary

In this chapter, we presented a distributed system architecture for real-time anomaly detection. The system can be employed in the AIS receivers and detect communication gaps online. We developed an algorithm that can efficiently detect vessels which switch-off their AIS transponder and demonstrated its increased performance in a real-world dataset and in real-world scenarios achieving an F1-score no less than 91%.

Part III
Model-based anomaly detection

5 A variation of the DBSCAN algorithm for anomaly detection

5.1 Introduction

Today's maritime surveillance systems are constantly flooded by data coming from AIS transponders, which are embedded in vessels. The use of AIS transponders was made compulsory for all vessels over 300 Gross Tonnage and all passenger vessels in 2002 by the Regulation 19 of SOLAS Chapter V[18]. However, even smaller vessels, from yachts to fishing boats [108], are now using AIS to report their positions to the nearby vessels, usually for safety purposes, making AIS the number one system for global vessel tracking. Each vessel transmits two kinds of AIS data, dynamic and static. The former, periodically sends data regarding vessel's position, speed and heading. The transmission rate depends on the vessel's speed and becomes higher when the speed is greater. The latter, sends data, every six minutes approximately, regarding vessel's destination, type, size and draught of its hull.

Due to the fact that AIS data are sent periodically with high transmission rates, they are of utmost importance to the maritime authorities for vessel tracking purposes. Therefore, a system that takes advantage of such data and is able to notify the authorities in real-time for any abnormal vessel behavior can be valuable for the authorities. This work contributes directly towards anomaly detection from AIS data. It builds upon our previous work in the field [109], which defined a methodology for extracting a network abstraction of the maritime traffic in an area. The input in that work was a lengthy log of AIS data collected from vessels that sailed in that area and the output was a network representation model of the typical routes that the vessels have followed. In that network representation, the nodes (also called *way-points*) are regions of special interest for the routes of vessel and they usually correspond to ports, capes, offshore platforms etc., where multiple vessels usually stop for short or longer periods, or perform major changes in their direction. Similarly, the connections between nodes represent the vessel movement from one way-point to another and thus, a vessel trajectory is a traversal of the network, from a certain way-point to a distant way-point. This traversal either follows the existing connections (and the trajectory can be considered normal) or deviates and hops from one node to another, not directly connected, node. The aggregated information from all vessels that crossed a network connection are used to extract features for this connection (a potential sub-trajectory for other vessels), such as the average, minimum and maximum speed etc.

In this work, we take this simple aggregation one step ahead, and provide a methodology that can be used to process this multi-vessel information in a more proficient manner. The proposed method adds richer information to each connection that have been traversed by multiple vessels. To extend the previously proposed network abstraction we use a clustering algorithm, that manages to identify different movement patterns for the same connection. This information is then used as a reference in the analysis of a vessels' journey and can allow to identify routes that deviate from the previously extracted patterns. Furthermore, it builds upon the semantic information of the edges of the network abstraction and adds to these connections common patterns the vessels must follow in order to travel between two way-points. Therefore, the common pathways and behavior of the vessels in terms of space, speed and heading are integrated to the already proposed network abstraction.

The main idea behind this work is that vessels of the same type (e.g., cargo vessels) that travel towards the same destination, follow common routes that pass through certain way-points and have similar moving patterns such as the same speed or heading. The major contributions of this work are:

- A variation of the popular density based clustering algorithm (DBScan) that takes into account the difference in speed and course as well as the spatial distance of trajectory points

[18] http://solasv.mcga.gov.uk/regulations/regulation19.htm

and extracts common navigation behaviors.

- A framework for taking advantage of these common navigation behaviors, by constructing movement models for different regions and vessel types and using them to detect deviations from the models.

A framework like this, allows further analysis by using well-known network analysis or data mining techniques enabling easier understanding of the maritime traffic.

The rest of the chapter is structured as follows. In Section 5.2, the proposed methodology of enriching the network abstraction model is presented in detail and Section 5.3 discusses the preliminary results of our methodology in anomaly detection. Finally, Section 5.4 concludes the chapter by summarizing the presented methodology and highlighting the impact of this work in the domain of the maritime surveillance by showing the possible use cases in the field of anomaly detection.

5.2 Approach

The proposed approach is applied to AIS data collected from multiple vessels of the same type (e.g., cargo vessels) for a predefined period of time and a predefined bounding box (e.g., geographic surveillance area of interest), but it is also applicable to larger geographic areas, periods of time and more types of vessels. Since, different types of vessels vary in size and shape, they may follow different routes even if they want to reach the same destination. Furthermore, specific vessel types such as cargo vessels might make much more intermediate stops (e.g. in middle sea platforms) than others. Although, the network abstraction model is the same for all types of vessels, the detailed information that it carries may vary per vessel type. So, in the following we present the model and the way its information is extracted but we demonstrate our approach on an AIS dataset from cargo vessels only.

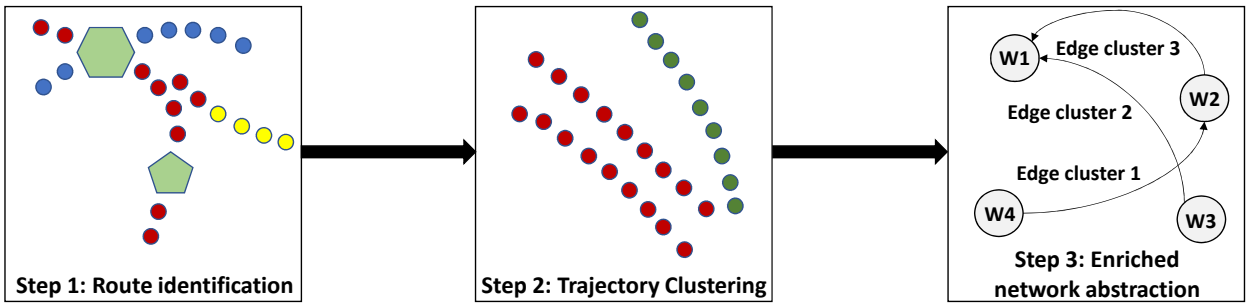


Figure 14: The steps of the proposed approach

The main steps of our approach are illustrated in Figure 14.

- In the *route identification* step, the way-points are extracted from multi-vessel trajectory data, following a methodology proposed in [109] and summarized in Section 5.2.1. Vessel trajectories are then expressed as sequences of sub-trajectories that connect intermediate way-points.

- The (sub-)*trajectory clustering* step is the main contribution of this work, which introduces a novel use of the DB-Scan algorithm that takes into account 3 parameters to identify neighboring points. The methodology followed in this step is explained in details in Section 5.2.2.

- In the *network abstraction model enrichment* step, several statistics are extracted for each cluster. The statistics summarize the movement of multiple vessels along the network edge. The details of these statistics and their extraction method is given in Section 5.2.3.

The final output model, comprises a set of way-points (vertices) dispersed across the monitored region and several sub-trajectory clusters (edges) with their statistics per cluster to represent the different ways of moving between two way-points. This output can be used for many use cases in the field of anomaly detection.

5.2.1 Route Identification

The first step of our methodology is the identification of way-points, which represent areas where many vessels have stopped (stop points) or did a major directional change (turn points) in the past. As already demonstrated in [109], way-points are created by clustering stop and turn points using a spatial density clustering algorithm (i.e. DB-Scan). The resulting way-points are the nodes of the network abstraction model, which contains information about way-points' size and density (number of stop or turn points per area unit). The size and density of way-points is strongly connected to the parameters of the DB-Scan algorithm. In our working examples, we focus only on the bigger way-points (i.e. those that contain more than 50 points). The idea behind this filtering is that bigger and denser way-points would belong to the trajectories of more vessels.

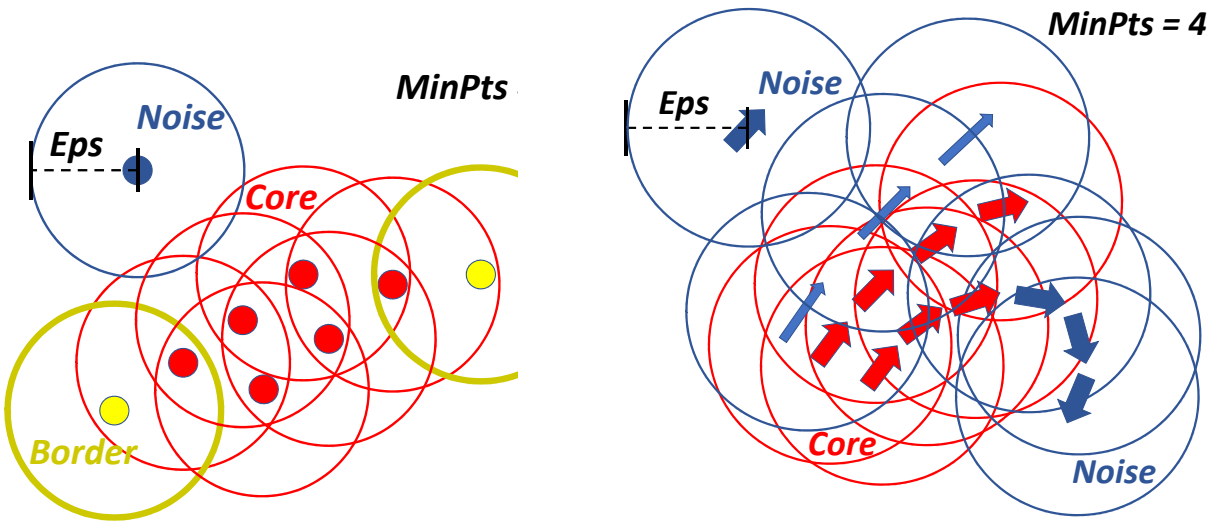
In our prototype analysis, we focus only on the trajectories that have at least 2 way-points, although the same methodology can be applied in all trajectories and respectively to all the edges of the network. Using different selection thresholds may result either in losing semantic information or in keeping too much information and this is a subject of further experimentation. For example, using higher thresholds (e.g. keeping even larger way-points only) will result in a higher level of abstraction and will probably loose the fine grained details of multiple vessel patterns, whereas using lower thresholds will result in keeping too much information and achieve low or no abstraction at all.

5.2.2 Trajectory Clustering

The second step refers to the clustering of the trajectories that have the same origin and destination way-points. The typical algorithm for clustering the points of one or more trajectories is DB-Scan [110], which is employed as a density-based spatial clustering method. DB-Scan takes two parameters, *epsilon* which specifies how close two points must be to be considered neighbors, and *minPts* which specifies the number of neighbors a point must have to be included in a cluster. Our proposed DB-Scan version uses 3 parameters to specify the proximity of candidate vessel AIS signals (positions):

- s: absolute difference of the speed between two positions (speed-based)
- h: absolute difference of the course over ground between two positions (heading-based)
- eps: harvesine distance between two positions (spatial-based)

To the best of our knowledge, this DB-Scan variation has not been used in the related literature. Therefore, each vessel position contains three types of information: i) the vessel speed at this position, ii) the vessel course over ground at this position, iii) the latitude and longitude of the position. Also, for a vessel position to be clustered together with another vessel position, the absolute difference in their speed must be below a threshold s, the absolute difference in their heading below a threshold h and the distance between them must be below a threshold eps at the same time. This type of clustering groups together trajectory points that have similar speed, heading and are close to each other. An example of this type of clustering can be seen in Figure 15 which compares the two implementations of the DB-Scan algorithm. Figure 15(a) shows the typical DB-Scan implementation, which creates a cluster if points are spatially close to each other. On the other hand, Figure 15(b) illustrates the modified DB-Scan for the positions of moving objects, which considers



(a) Typical DB-Scan implementation. (b) Modified DB-Scan.

Figure 15: Comparison of DB-Scan implementations.

two points (actually two vectors with position, direction and speed) to be in the same neighborhood when the vectors' positions are spatially close to each other, but they also have similar direction and speed. In the modified version blue arrows indicate noise vectors, which are either away, or have different speed or have a different direction from all their neighbouring vectors.

To have more accurate clustering results, we exclude positions that are located inside the way-points. Since way-points are areas of interest through which vessels frequently pass, it can be easily inferred that the way-points might be ports, platforms, canals or waterways. Inside these way-points, vessels tend to alter their speed or heading frequently, which may corrupt the clustering results.

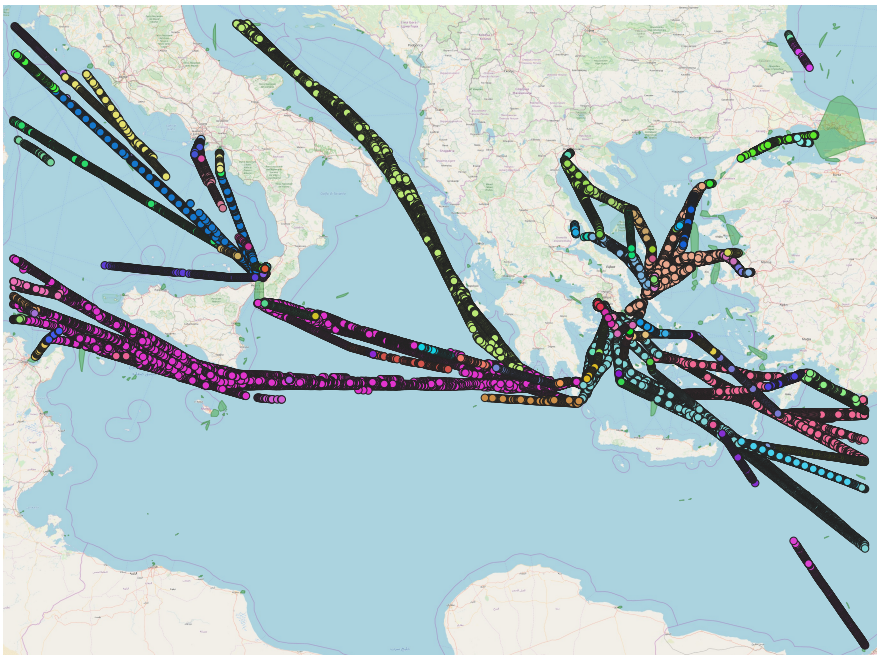


Figure 16: Example of the trajectory clustering.

Figure 16 illustrates the result of running the proposed trajectory clustering method[19] to all

[19]DB-Scan parameters have been empirically determined

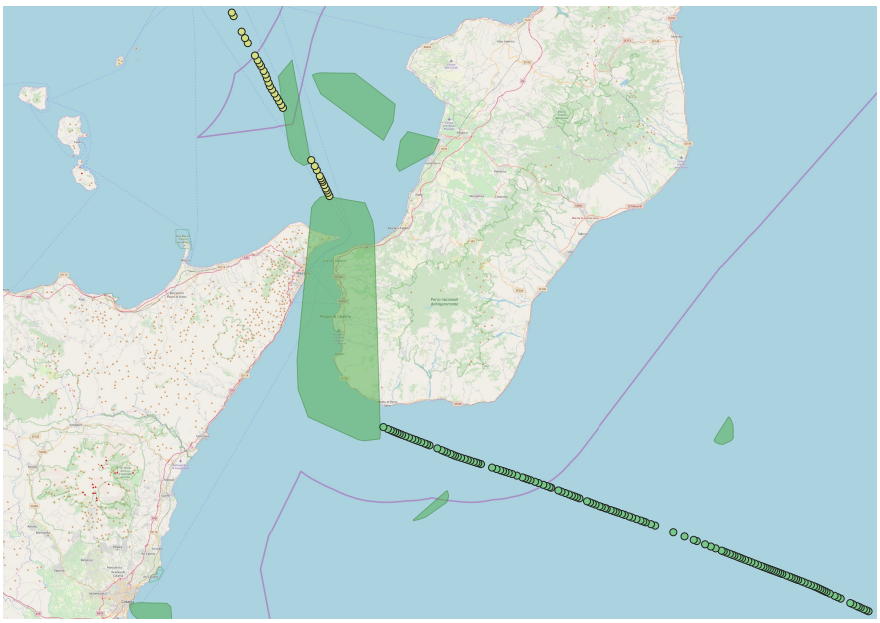


Figure 17: Example of the edges of the network abstraction.

cargo vessels that sail in the east Mediterranean sea and are headed to the port of Piraeus, Greece, using $s = 3$, $h = 3$, $eps = 20km$ and $minPts = 10$. We can see that trajectories with similar speed and heading are placed in the same cluster, which resembles to the behavior of the basic DBScan (e.g. the cluster formed in the Adriatic sea), whereas points of the same trajectory may belong to different clusters, even though they are spatially close, because of the differences in speed or heading (e.g. the clusters that are formed near the port of Tripolis, Lybia, on the left part of Figure 16).

5.2.3 Enriched Network Abstraction

The final step of the process is the enrichment of the network model with information about the clusters of sub-trajectories in each network edge (or in selected edges, e.g. the most frequently traversed). Since, we have created clusters of trajectories (edges) between way-points, we can add information to these edges to form a comprehensive network of the maritime traffic. To this end, for each cluster or edge of the network we calculate the average travelling speed and heading of the vessels. Moreover, the typical deviation of these values is also calculated. Finally, the start point and the end point of the cluster are computed (beginning and ending of the trajectories) along with the average temporal distance of each cluster (average time taken to travel from the start of the cluster to the end of it). Figure 17 illustrates a small snapshot of the network near Sicily, Italy. The green shaded convex hulls represent way-points (vertices) and the green and yellow dots are the points that comprise the trajectory of a single vessel[20]. For the (yellow) sub-trajectory points that connect the two way-points of the figure, the centroid of the respective cluster has a heading of 319.15 whereas the centroid of the other (green) (sub-trajectory) has a heading of 322.28.

5.3 Application to a real dataset

To examine the results of our enriched network model, a dataset provided by MarineTraffic was used, containing 2.9 million AIS messages received from 1,716 distinct "cargo" vessels sailing in the eastern half of the Mediterranean Sea during August 2015. Since no information about the existence of anomalous behaviors existed in this dataset, we employed unsupervised techniques

[20]For demonstration purposes, this cluster contains points from a single vessel's trajectory.

to detect potential anomalies or outliers. Although outliers can be detected, further examination is required to understand the reason behind the unusual behavior and the characteristics of the trajectories selected.

5.3.1 Network creation from real AIS positions

The first step in building the enriched network abstraction is the creation of the way-points (vertices of the network). The identification of the way-points is a two-step process that requires to $i)$ identify key-points in the trajectories of the vessels and $ii)$ spatially cluster together dense key-points. To identify the key-points we used a speed threshold of 2 knots and a bearing rate threshold of 0.1 degrees per minute, which resulted in several thousand low speed AIS positions and turns in the trajectories of the surveillance area. To create the clusters of key-points, we used the DB-Scan algorithm with a minimum number of ten key-points ($minPts = 10$) within a radius of $2km$ ($eps = 2000$), resulting in 616 clusters.

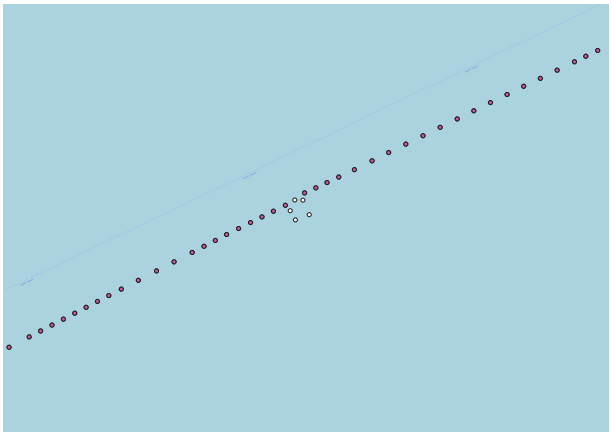
The second step involves clustering of the trajectories with similar characteristics. For this step, we grouped the trajectories per destination and applied the proposed modified version of DB-Scan, which requires for 3 parameters to be satisfied in order for a point to be considered a neighboring one (speed-based, heading-based, spatial-based). For a point to be in the same cluster, its speed must not differ more than 3 knots and its heading more than 3 degrees within a $20km$ radius. Moreover, a minimum number of 10 points is required to form a cluster.

In the remainder of this section we demonstrate cases of vessels that had unusual behavior in terms of the way they deviate from their route or in terms of the way they suddenly change course to reach the same destination.

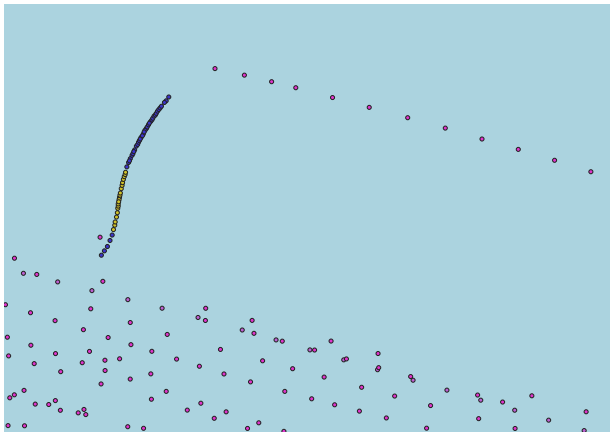
5.3.2 Detection of outliers in the trajectories

The lack of Maritime Situational Awareness (MSA) is a key factor in many incidents that are due to crew fatigue, stress or even engine failures, despite the major improvements in maritime safety. A sudden change in the course of a vessel is considered a noteworthy or anomalous event for the maritime authorities for several reasons, either due to human factors or technical ones. Several cases have been recorded in the past, in which engines fail during a vessel's voyage and the vessel starts drifting away from its normal route. This type of deviation in a vessel's route could potentially lead to collisions with nearby vessels or collisions with rocky islands, endangering multiple vessels in the vicinity or the environment (e.g., oil spills). Such small deviations from the normal route cannot be detected by algorithms that seek for major turns, and the same holds for temporal decelerations or accelerations and algorithms that seek for sudden stops. Similarly, when vessels are in distress due to piracy attacks or when they take part in search and rescue operations and they perform manoeuvres, it is not always feasible to detect such combined actions that include speed and route change and deviation from the normal route. These types of behavior require an immediate course of action by the authorities. The proposed network abstraction model, with the information it carries on each edge concerning the clusters of movement patterns (in terms of speed, course over ground and location) is able to capture such cases that comprise small or larger deviations in the trajectories. A few outlier cases that have been detected (Figure 18) on a real dataset are presented in the following.

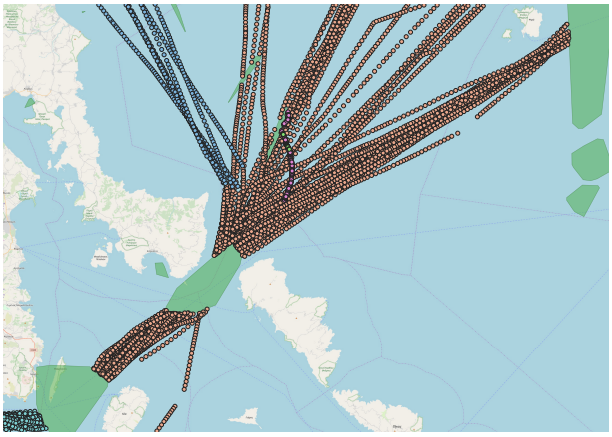
Figure 18(a) illustrates a vessel's trajectory towards Naples, Italy. During its voyage the vessel makes a small circle and then continues its journey as before. Since its heading and speed changed dramatically the points in the circle (i.e. white) are considered outliers. Figure 18(b) illustrates the maritime traffic from the west to east, near Sicily, Italy. The trajectories from multiple vessels are grouped in the same cluster, since they share the same course and speed values and are drawn with the same colour (i.e. magenta). The centroid of this cluster has a heading of 102.3 degrees and a speed of 13.91 knots. However, the part of the trajectory of a vessel that deviates from the normal



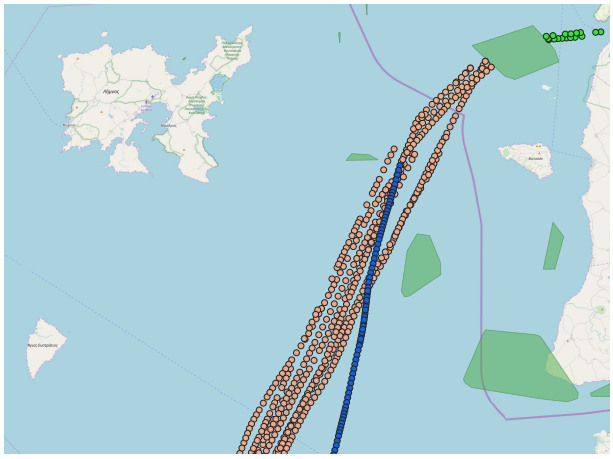
(a) Example of an unusual loop in a vessel's trajectory.



(b) Example of an unusual and steep deviation of a vessel.



(c) A trajectory that does not follow the usual maritime traffic has been detected.



(d) A trajectory that slowly deviates from its course.

Figure 18: Outliers detected by the proposed trajectory clustering.

route, starts heading to the north and after a while follows the same direction as before is marked with blue and yellow dots, since it moved to a different cluster. The blue cluster centroid has a heading of 30.2 degrees and a speed of 1.2 knots (with a standard deviation of 5.25), whereas the respective centroid for the yellow cluster has a heading of 11.4 degrees and a speed of 1.2 knots (with a standard deviation of 2.75). The actual centroid values clearly indicate an outlying behavior from a vessel that changed its route in slow speed in an area where similar (i.e. cargo) vessels move in different speed and direction. In a different case, Figure 18(c) visualizes the maritime traffic of cargo vessels in the Aegean sea, showing all the vessels heading to the port of Piraeus, passing south of the island of Evia and near the island of Andros, Greece. There are two distinct clusters in the plot: i) a big one that contains the trajectory of vessels traveling from the north-east Aegean sea, with a centroid of 227.3 degrees ($stdev = 21.32$) and 13.1 knots ($stdev = 2.49$) and ii) one that contains vessels traveling from the north-west, with a centroid of 137.8 degrees ($stdev = 14.26$) and 13.0 knots ($stdev = 1.65$). The two clusters eventually merge into one cluster when the vessels pass south of Evia. Almost hidden among the two clusters is a third smaller cluster (marked with purple points) which illustrates a large deviation of a vessel that does not follow the patterns of all vessels with the same destination. This last cluster has a centroid of 145.8 degrees ($stdev = 2$) and 1.4 knots ($stdev = 0.16$). With the proposed clustering algorithm, this subtrajectory, which does not contain any large and sudden course change or a stop has been identified as an outlier. Finally, Figure 18(d) shows the maritime traffic near the island of Lemnos, Greece. From the plot it is obvious that while all vessels follow a specific route (the same big cluster as in Figure 18(c)), when they head towards the port of Piraeus, using similar speed and heading values, there is one vessel that slowly deviates (marked with blue colour) from the common route, for unknown reason. This outlier has an average heading of 192.7 degrees and 9.8 knots speed. The comparison between the normal behavior (227.3 degrees, with $stdev = 21.32$ and 13.1 knots, with $stdev = 2.49$) shows that this outlier moved much slower that all other cargo vessels too.

All the cases presented above, are extracted from a dataset of $1,716$ cargo vessels, following a totally unsupervised method (clustering). As a consequence, it provided us with useful feedback on the applicability of the proposed method and on the type of deviations it can detect. However, the same methodology can be used as a basis for a supervised (classification) technique that will detect vessel deviations using pre-trained cluster information.

5.4 Summary

In this chapter, we proposed a clustering technique, which can be used to enrich our previously proposed maritime traffic network [109] that can efficiently model the behavior of vessels using only free and openly transmitted AIS data. The modelling of the normal vessel behavior will allow us to further distinguish outliers in the trajectories that are of interest to the maritime authorities. In this work, we showcased a few real world examples which our model managed to accurately detect. Identifying specific cases of anomalous behavior ([25, 26, 9, 37]) will allow us to fine-tune, improve and exploit the proposed unsupervised technique as a basis for a supervised model for the detection of events of interest in the maritime sector.

6 A distributed framework for extracting maritime traffic patterns

6.1 Introduction

Despite the importance of maritime surveillance for the safety of passengers, vessels and their cargo, it was only since July 2002, when the Regulation 19 of the International Convention for the Safety of Life at Sea (SOLAS-V) came into force, that large cargo vessels are obliged to fit and use AIS for reporting their position, speed and heading, for serving surveillance purposes. This regulation resulted in an abundance of data that completely changed the maritime surveillance scenario. AIS transponders are today used also by smaller vessels, such as yachts or fishing boats ([108]), that transmit their position and route to nearby vessels in order to avoid collisions. Additional data, concerning the vessel's origin and destination, its type, size and draught are also periodically transmitted by AIS equipped vessels, thus providing a rich data set for further analysis.

In this direction maritime authorities can use this data for tracking vessels and analysing their behavior in order to early detect abnormalities, which may happen due to an engine failure, an accident in the sea, or on purpose. Building on the ideas of previous works in the field of anomalous vessel behaviour detection [109, 12], this work focuses on the extraction of traffic patterns, which can be the basis for detecting such behaviours.

In this work, we extend [12] by employing concepts of the network abstraction model proposed in [109] and develop a methodology for processing multi-vessel AIS data more efficiently, using richer information on the network edges. The new network abstraction aims to extract multiple movement patterns for the same network edge, which correspond to a fine-grained clustering of the collected AIS data. Using Lagrange interpolation for adding intermediate points to the vessel trajectories improves the performance of the clustering algorithm [11]. The algorithm parameters are fine-tuned by considering vessels with similar properties (e.g. vessel type) in the same region and the processing speeds up by employing map-reduce paradigms. The resulting network is used as a basis for detecting vessel route outliers, that at some point decline from the 'typical' route in position, speed or direction.

The main assumption of this work is that vessels of the same type travel between the same way-points by following the same route and similar speed, location and direction, does not hold for all types of vessels (e.g. for sailing boats or fishing boats), but is an assumption that holds for large vessels such as cargo vessels or passenger ships, which mainly employ AIS. The major contributions of this work are:

- It modifies the definition of distance in the DB-Scan density based clustering algorithm, by considering in tandem the speed, course and spatial position of the trajectory points. The resulting clusters represent the different ways of sailing from one way-point to another.
- It provides a distributed implementation of the proposed solution, which allows more efficient processing framework for taking advantage of these common navigation behaviors, by constructing movement models for different regions and vessel types and using them to detect deviations from the models.
- A movement model that can work well with either sparse data (vessels that do not transmit positions frequently due to limitations such as weather conditions) or dense data (high traffic regions at sea that are flooded with AIS positions).
- A movement model that can adequately extract movement patterns in surveillance areas with many islands, such as the Aegean sea.

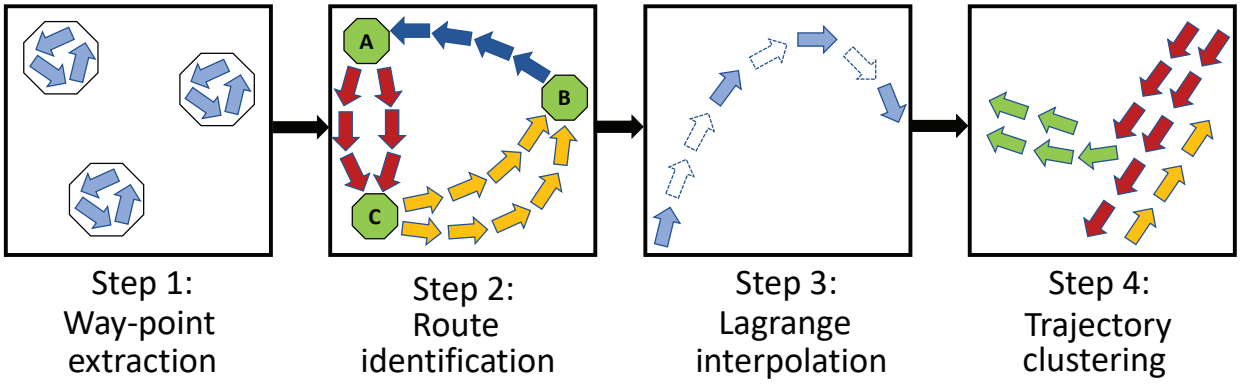


Figure 19: The steps of the proposed approach.

The proposed framework can be the basis for a more profound analysis and understanding of the maritime traffic of specific types of vessels in a region.

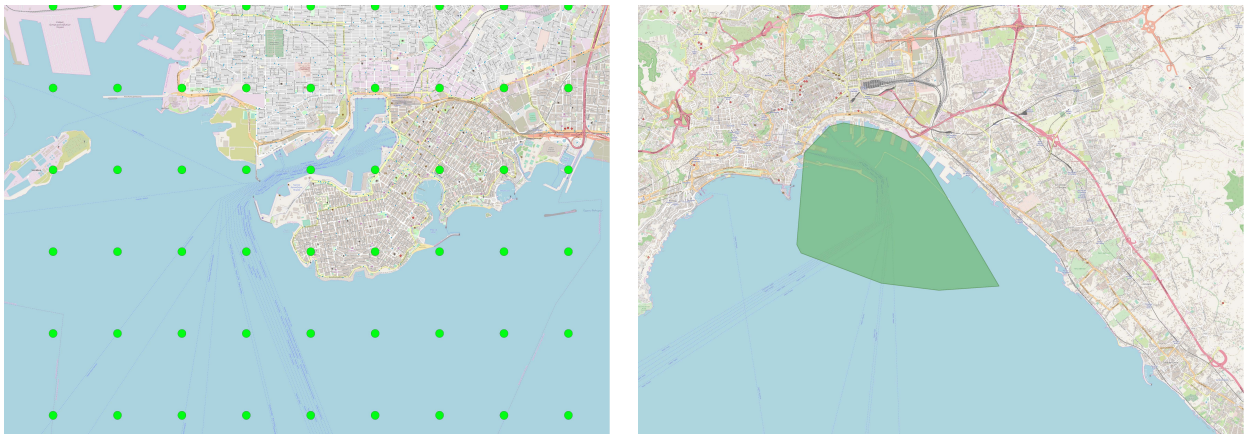
The rest of the chapter is structured as follows. The next section mentions the related work in the field. Section 6.2, describes in detail the proposed methodology of extracting maritime traffic patterns, while Section 6.3 provides details on the implemented distributed architecture. Section 6.4 evaluates the proposed framework in its ability to identify these patterns. Section 6.5 discusses our findings in comparison with those of the related literature. Finally, Section 6.6 summarizes the findings of this work and highlights some perspectives of the proposed framework that need further investigation.

6.2 Proposed Approach

In order to demonstrate the use of proposed framework in maritime surveillance, we applied it to a dataset that contains AIS data from a single type of vessels (e.g., cargo vessels). The methodology for extracting maritime traffic patterns can be the same, but since vessels of a different type may also vary in size and shape, the resulting patterns may vary per vessel type. Although the methodology can be applied to large geographic areas and time periods, we choose a dataset that spans a few months' period (in order to capture the seasonality of patterns) and covers a specific area of interest. In the following we present the maritime traffic model and detail on how the information it contains is extracted from raw AIS data.

The main steps of our approach are illustrated in Figure 19 and presented in detail in the sub-sections that follow.

- In the *way-point extraction* step, which is explained in Section 6.2.1, way-points are extracted from historical AIS data.
- In the *route segmentation* step, which is presented in Section 6.2.2, the compete trajectories are represented as sequences of sub-trajectories, each one connecting a pair of intermediate way-points, in the path from the origin to the destination port way-points.
- In the *Lagrange interpolation* step, trajectories that are incomplete are reconstructed using polynomial interpolation. Section 6.2.3 provides a detailed description.
- The (sub-)*trajectory clustering* step introduces a variation of the DB-Scan algorithm that examines spatial, speed, and direction distances (or differences) in order to identify neighboring points and thus groups together sub-trajectories that are located at a close distance and have similar speeds and directions. The methodology followed in this step is explained in detail in Section 6.2.4.



(a) The tiles in the port of Piraeus. Green dots denote the lower left corner of each tile.

(b) Waypoint in the port of Napoli.

Figure 20: The way-point extraction process

The resulting maritime traffic model contains way-points (vertices) that group the stop or turn points of multiple vessels and sub-trajectory clusters (edges) that contain statistics extracted from all vessels that sailed from one way-point to another. Edge statistics represent the preferred ways of sailing between two way-points. This output can be used for many use cases in the field of anomaly detection.

6.2.1 Way-point Extraction

The first step of the approach identifies areas in which the vessels remain stationary (way-points) for a reasonable amount of time. These way-points can be either ports, anchorages or off-shore platforms, and will be used later in the route identification step. To this end, from the historical records we only keep the AIS positions which report zero speed and apply a spatial density-based clustering algorithm on them. In order to further reduce the amount of the data to be clustered and speed up the clustering process that follows, we apply a simple compression technique, in which the surveillance area is segmented into small tiles of 0.01 degrees of longitude and latitude (approximately $1km^2$). Then, we further compress the dataset and keep only the temporally first position of each vessel per tile. The idea behind this simple compression step is that stationary vessels keep transmitting (almost) the same AIS position, thus adding no significant information for the steps that follow. Due to the fact that vessels can remain stationary for large periods of time, the amount of positions that are filtered out is significant (the percentage of positions that can be discarded can reach up to 99%). The tile size selection was based on the dimensions of larger vessels such as Cargo, Tanker, or Passenger, which can reach up to 500 meters in length. The tile size needs to be as small as possible, but not small enough to discard more positions from such larger vessels. Therefore, we selected the width and length of the tiles to be twice the size of the vessels ($1km$ of length and width). Figure 20(a) illustrates the tiles in which the surveillance area is segmented, while Figure 20(b) illustrates the resulting way-point in the port of Napoli.

When the compression step is complete, the DB-Scan algorithm is used to group the remaining positions together into clusters. DB-Scan takes two parameters, *eps* (or epsilon), which specifies how close two points must be to be considered neighbors, and *minPts*, which specifies the number of neighbors a point must have to be included in a cluster. The parameters used in the algorithm are $eps = 2km$ (an average radius for small or larger ports) and $minPts = 10$ (minimum number of neighbouring vessel positions in a radius). In real-world terms, the aforementioned values means that we are seeking for regions, where the density is at least 10 vessels in a $12.5km^2$ area, which is expected to detect rather dense clusters.

In the absence of port geometries, this way-point extraction methodology can be useful, since the detected clusters denote the areas where vessels are moored or anchored. In order to better represent these areas we find the convex hull [111] of each cluster (i.e. the minimum bounding geometry that contains all the cluster points).

6.2.2 Route Identification

The second step in the pattern extraction process is the route identification. This step is of utmost importance since patterns must be extracted per itinerary. Let A be the starting way-point of a vessel and B be the destination way-point of the same vessel. The path a vessel followed to go from point A to point B is defined as a route $R_{A\rightarrow B}$. Itinerary $I_{A\rightarrow B}$ is defined as the sum of all routes of all vessels from A to B:

$$\sum_{n=1}^{n=paths} R_{A\rightarrow B} = I_{A\rightarrow B} \tag{3}$$

Several vessels may follow the same route in order to move from the starting to the destination point, thus an itinerary may consist of routes travelled from one or more vessels. Trajectory is defined as a list P of temporally sorted points or positions p of a vessel. In short, a trajectory contains all of the AIS positions during a vessel's lifespan. Algorithm 5 describes the process of route identification per vessel in detail. Initially, the algorithm is given a trajectory and a set of way-points W. The previous way-point W_p is set to -1 (line 2) since in the beginning no way-points have been encountered. All points of a vessel's trajectory are examined (line 6) against the previously detected way-points, in order to split the trajectory into routes. The *within* function is responsible for finding the way-point W_{p_k} (if any) that contains the trajectory point p_k (line 7). When a trajectory point is found $within$ a way-point for the first time (line 8) a new route starts and runs inside the way-point (i.e. a vessel begins its route within the port) (lines 9 - 13). All the continuous trajectory points that intersect with the same way-point (lines 15) are added to the route. When the first trajectory point that is out of the way-point is found, then a new route begins, which heads to the next way-point (i.e. the vessel sails to the next way-point) (lines 16-22). This way-point can either be the destination point of the route or simply an intermediate way-point (e.g. an off-shore platform, or an intermediate port).

The process continues until all points of the trajectory are examined and the output is a list of routes within or between way-points. Trajectories without starting or destination way-points are ignored in the following steps of the proposed methodology. "Empty" starting points occur when the first position of the trajectory is not within a way-point, in which case the initial way-point is set to -1 (line 2), while "empty" destination way-points occur when the vessel exited the surveillance area and we have no knowledge of its intermediate trajectory.

The output of the process is a set of lists of positions, each list representing a route. Several routes may refer to the same itinerary (equation 3). Figure 21 illustrates the output of the route identification step in the area between Italy and Greece. Each line depicted with different color represents a different route.

6.2.3 Lagrange Interpolation

The next step in the proposed workflow is the interpolation of the trajectories of every route. The reason behind this step is to fill the gaps that may arise in real vessel trajectories. Such gaps may affect the quality of the traffic patterns extracted with the proposed methodology, since they directly affect the quality of the clustering step that follows (see Section 6.2.4). It is quite common to have such gaps in vessel trajectories because although it is mandatory for vessels to carry an AIS transponder, it is not compulsory for the transponder to be switched on [34, 36, 11]. This is

Algorithm 5 Route identification algorithm

Input: List of trajectory points P, Set of way-points W
Output: List of routes L_s

function RouteIdentification($P[\,]$)
    $W_p \leftarrow -1$
    $N \leftarrow length(P)$
    $S \leftarrow []$
    $L_s \leftarrow []$
    for $k \leftarrow 1$ to N **do**
        $W_{p_k} \leftarrow within(p_k, W)$
        if $W_{p_k} \neq \emptyset$ **then**
            if $W_{p_k} \neq W_p$ **then**
                $S.append(p_k)$
                $L_s.append(S)$
                $S \leftarrow []$
                $W_p \leftarrow W_{p_k}$
            else
                $S.append(p_k)$
        else
            $W_{p_k} \leftarrow -1$
            if $W_{p_k} \neq W_p$ **then**
                $S.append(p_k)$
                $L_s.append(S)$
                $S \leftarrow []$
                $W_p \leftarrow -1$
            else
                $S.append(p_k)$
    $L_s.append(S)$
    return L_s

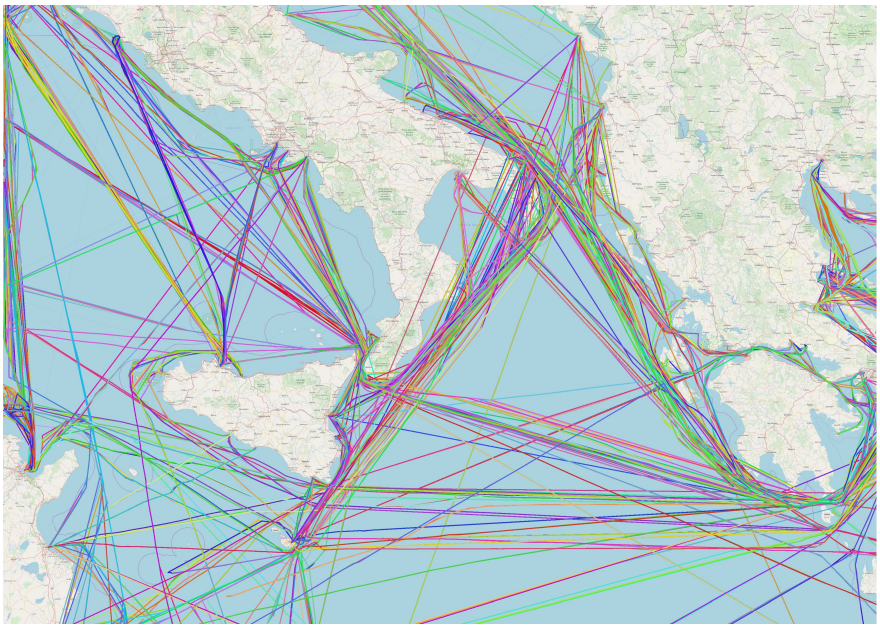


Figure 21: The output of the route identification step.

a common tactic when vessels want to hide their tracks and conceal their whereabouts, in order to avoid piracy attacks, or perform an illegal act themselves (e.g. fishing in prohibited areas). Furthermore, AIS gaps in the trajectories may also happen, either due to bad weather conditions or because the receivers are deliberately jammed or, in rare occasions, because of packet collisions that take place when the AIS receivers are flooded with messages.

To tackle the problem of trajectory gaps, several studies have focused on finding an optimal interpolation algorithm for movement data only in recent years, either in the maritime domain [112, 113, 114, 115, 116] or not [117]. [114] proposed a new method which includes linear interpolation and Cubic Hermit interpolation along with a mechanism to overcome problems of both approaches in order to fill in the gaps of missing AIS data. [112] developed an interpolation model which is able to accurately estimate the vessel's vectors of speed, heading and acceleration. [117] proposed a new approach, called kinematic path interpolation, for interpolating movement data. A new interpolation technique is proposed in [116] and compared against Lagrange interpolation, Cubic spline interpolation and Hermite interpolation over AIS data, with all of the algorithms yielding an error between 0.072 and 0.022. In our work, we used the Lagrange interpolation [118], a well-known, established algorithm, that is preferred over Newton interpolation because it gives better approximations [119]. The interpolation itself is not the focus of our study, nevertheless, the various interpolation techniques can be used in the future to further study their effects in our approach. Given a set of n data points

$$(x_0, y_0), ..., (x_j, y_j), ..., (x_{n-1}, y_{n-1}) \tag{4}$$

where no two x_j are the same, the Lagrange interpolation is a linear combination

$$L(x) = \sum_{j=0}^{n-1} y_j \times l_j(x) \tag{5}$$

of Lagrange basis polynomials

$$l_j(x) = \prod_{m=0, m\neq j}^{n-1} \frac{x - x_m}{x_j - x_m} \tag{6}$$

where $0 \leq j < n$. In our case, x and y correspond to longitude and latitude respectively.

To apply equation 5, we run a sliding window of $step = 1$ and $size = 3$ ($W_{1,3} = [p_0, p_1, p_2]$, where p_j is a data point) in each trajectory, in order to always use $n = 3$ data points[21] to create the polynomials (2^{nd} degree polynomial). Since the Lagrange interpolation requires $O(n^2)$ multiplications and subtractions to evaluate the interpolation at a point, higher degree polynomials require more time to calculate. Furthermore, higher order polynomial interpolations suffer from the Runge's phenomenon [120], which means that the generated curve tends to oscillate at the edges of an interval, resulting in erroneous interpolations. At each window $W_{1,3}$, a curve is generated by applying equation 5 on $W_{1,3}$. Next, we calculate the x-axis distance and the temporal distance between data points p_1 and p_2 such that:

$$D_x = p_{x_1} - p_{x_2} \tag{7}$$

and

$$D_t = p_{t_1} - p_{t_2} \tag{8}$$

[21] The minimum number of data points required to create a curve.

In order to calculate the number of interpolating points to be added to the trajectory, we take into account the minimum transmission frequency of the AIS, which is $T_f = 3$ minutes[22]. By dividing the temporal distance with T_f we calculate the number of interpolating positions k (equation 9). Later, we divide the spatial distance with k to calculate the spatial increment x_{incr} (equation 10), which indicates how much the vessel has moved in the x-axis at each time-frame of the missing trajectory.

$$k = \frac{D_t}{T_f} \tag{9}$$

$$x_{incr} = \frac{D_x}{k} \tag{10}$$

Since we know that every 3 minutes the vessel moves by x_{incr} and that the vessel has transmitted k positions in total during the communication gap, it is easy to calculate the corresponding values (y, speed and heading) of the interpolating positions. Algorithm 6 describes the process of calculating the aforementioned values in detail.

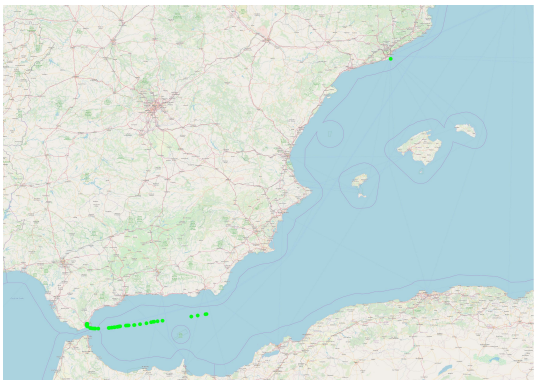
Algorithm 6 Position interpolation per window $W_{1,3}$

Input: p_1,k, T_f, x_{incr}
Output: List of interpolating positions L_p

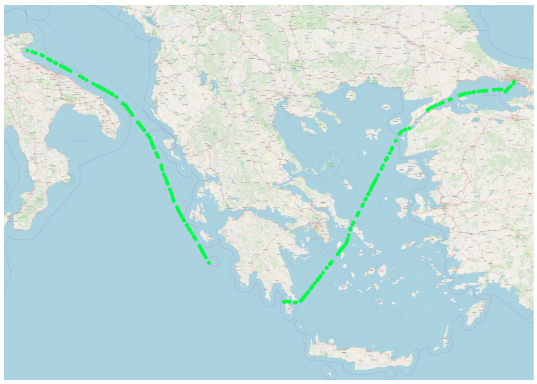
1: **function** PositionInterpolation(p_1,k,T_f,x_{incr})
2: $P_{previous} \leftarrow p_1$
3: $L_p \leftarrow []$
4: **for** $i \leftarrow 1$ until k **do**
5: $timestamp \leftarrow P_{previous}.timestamp + T_f$
6: $longitude \leftarrow P_{previous}.longitude + x_{incr}$)
7: $latitude \leftarrow lagrangeInterpolation(longitude)$
8: $distance \leftarrow haversineFormula(P_{previous}, longitude, latitude)$
9: $speed \leftarrow distance/T_f$
10: $bearing \leftarrow calculateHeading(P_{previous}, longitude, latitude)$
11: $L_p.append((timestamp, longitude, latitude, speed, bearing))$
12: $P_{previous} \leftarrow (timestamp, longitude, latitude, speed, bearing)$
13: **return** L_p

At each step of the iteration (line 4) the new interpolating timestamp and longitude is calculated by adding the corresponding increment values (lines 5 and 6 respectively), while the new latitude value is calculated using the Lagrange formula (line 7). To calculate the speed (line 9), we first need to calculate the distance between the previous and the current position (line 8). We use the Haversine formula because it takes into account the earth's curvature. Then, the new heading is calculated by using simple trigonometric functions (line 10). Finally, the newly created interpolating position is added to the list (line 11) and becomes the previous position (line 12) as the iteration proceeds. To constrain the interpolation from creating unrealistic positions, we apply the interpolation to each route $R_{A \to B}$, after positions inside way-points have been removed. Way-points contain a large amount of positions that are anchored or moored, which means that positions will be scattered in a small area inside the way-point, rather than follow a specific kinematic path, introducing extreme curvatures and extreme speed values. Moreover, routes that have a gap equal to or more than 24 hours are removed from the process.

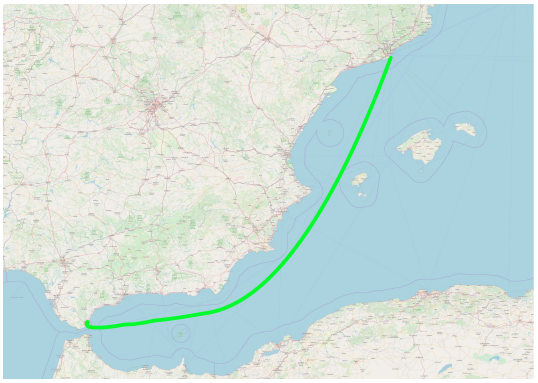
[22]The transmission frequency of the AIS depends on the speed and the rate of change in course. The higher the values, the higher the transmission frequency. The highest transmission frequency is one position every 2 seconds.



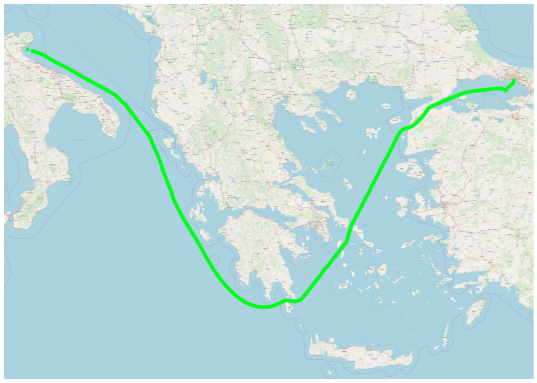
(a) **Before.** Trajectory of a vessel traveling from Gibraltar to Barcelona.



(b) **Before.** Trajectory of a vessel traveling south of the Peloponnese and towards Istanbul.



(c) **After.** Trajectory of a vessel traveling from Gibraltar to Barcelona.



(d) **After.** Trajectory of a vessel traveling south of the Peloponnese and towards Istanbul.

Figure 22: Examples of Lagrange interpolation applied to two trajectories.

Figure 22 visualizes the result of the Lagrange interpolation applied to two distinct trajectories with large communication gaps. Figures 22(a) and 22(b) illustrate the trajectories with the communication gaps. Figure 22(a), shows a vessel that is traveling towards Barcelona, Spain, but the majority of its trajectory is missing. Similarly, Figure 22(b) visualizes a vessel that is traveling from the Ionian sea towards the Aegean sea, when a large communication gap appears at the south side of the Peloponnese. Figures 22(c) and 22(d) illustrate the same trajectories after the Lagrange interpolation. In both cases, our approach at interpolating missing positions manages to fill in the missing segments efficiently.

6.2.4 Trajectory Clustering

The final step of the approach refers to the clustering of the trajectories that have the same itinerary $R_{A \to B}$. The typical algorithm for clustering the points of one or more trajectories is DB-Scan [121], which is employed as a density-based spatial clustering method. Our proposed DB-Scan version uses 3 parameters to specify the proximity of candidate vessel AIS signals (positions):

- s: absolute difference of the speed between two positions (*speed-based*)
- h: absolute difference of the course-over-ground (CoG) between two positions (*heading-based*)
- eps: Haversine distance between two positions (*spatial-based*)

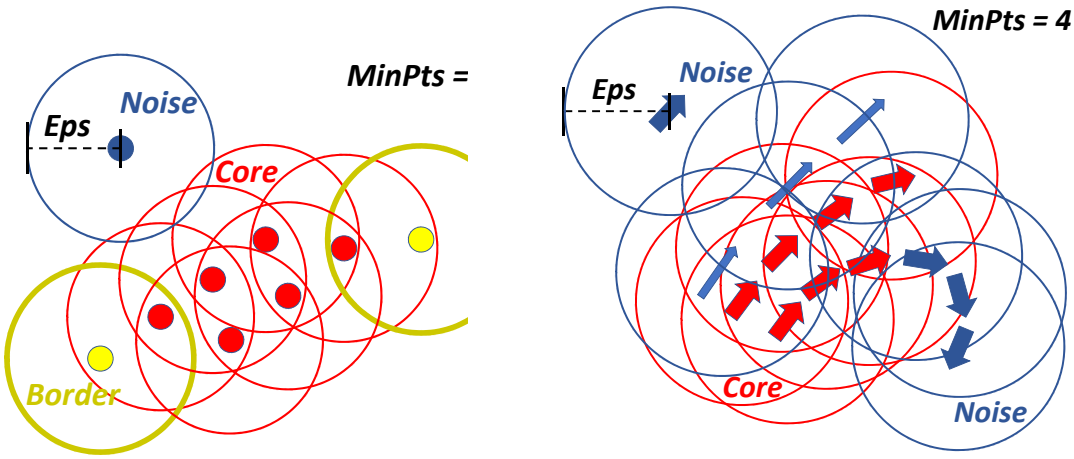
Therefore, for each position in a vessel's trajectory we keep: i) the vessel's speed, ii) the vessel's direction (course over ground), and iii) its latitude and longitude coordinates. Then, three different thresholds are employed for deciding whether the positions from two different vessel trajectories are neighbors: the speed difference threshold s, the heading difference threshold h and the distance threshold eps. When two positions from the same or different vessel trajectories have an absolute difference in all three dimensions that is below the threshold, the trajectory points are clustered together.

Figure 23 illustrates the differences between the two variants of the DB-Scan algorithm. Figure 23(a)[23] on the left illustrates the original DB-Scan algorithm, that uses only the spatial distance as a neighboring criterion. Figure 23(b), on the right, depicts the modified DB-Scan for moving objects, which considers two (multi-dimensional) points (comprising position, direction and speed dimensions) to be close to each other ('neighbors' in DB-Scan notation) when the positions are spatially close to each other, and there are small differences in direction and speed.

Each color in Figure 23(b) represents a different cluster. Red arrows belong to the same cluster because they are close to each other and have approximately the same speed and heading. Dashed thin blue arrows indicate a different cluster because they have a higher speed from the red arrows. Dotted thick blue arrows indicate outliers, which are either away, or have different speed or direction from all their neighbouring vectors. Specifically, dotted thick blue arrows at the bottom right have a different direction from the rest, while the dotted thick blue arrow at the top is away in terms of space.

Before applying the modified DB-Scan algorithm, we first segment the surveillance area into tiles of 0.2 degrees. The choice of this resolution of the grid was made because most of the weather services[24] use the same resolution, thus integration with such services is easier, allowing us to further extend our approach in the future (e.g. enrich each tile with weather information to explain unexpected vessel movement behaviors). Furthermore, we do not need a tile size as small as the one mentioned in Section 6.2.1 because the small tile size is not ideal for the representation of the movement of multiple vessels (in small tile sizes significantly less number of vessels is located

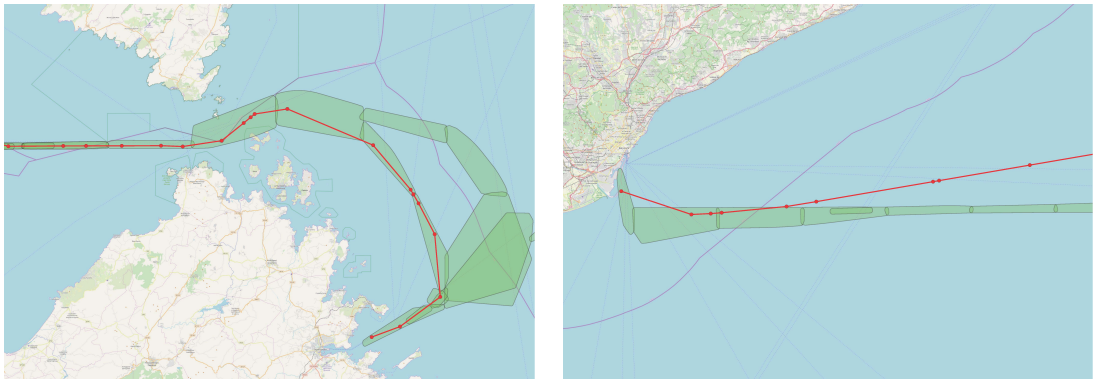
[23] https://en.wikipedia.org/wiki/DBSCAN
[24] https://www.noaa.gov/weather



(a) Typical DB-Scan implementation.

(b) Modified DB-Scan.

Figure 23: Comparison of DB-Scan implementations.



(a) A vessel traveling towards Olbia city in northern Sardinia and moves within the convex hulls.

(b) A vessel that departed from Barcelona, Spain making a deviation in its route.

Figure 24: Examples of normal and anomalous behavior.

inside the same tile, especially in more sparse areas). On the other hand, the segmentation in larger tiles allows for better examination of multiple vessel behavior per area, since vessels in different areas might move in a different way. Then, we apply the clustering algorithm in each tile and for each itinerary $R_{A\to B}$ in each tile separately (each tile may contain more than one itinerary). The parameters of the DB-Scan are extracted from the data in each tile-itinerary pair. The s parameter is the standard deviation of the speed between the positions of each pair. Similarly, h and eps is the standard deviation of heading and Haversine distance between the positions of each pair, respectively. We use the standard deviation because it measures the amount of variation or dispersion of a set of values. Positions with the lowest dispersion of speed, heading and distance need to be grouped together. $minPts$ is set to 6 because we need at least two 3-position-length routes per itinerary. 3 is the minimum number of positions a trajectory should have to be considered valid. Therefore, clusters with one route are not considered common and are excluded from the process.

To have more accurate clustering results, we exclude positions that are located inside the way-points. Since way-points are areas of interest through which vessels frequently report zero speed, it can be easily inferred that the way-points might be ports, platforms or anchorage areas. Furthermore, since way-points are extracted from all the vessels frequently reporting zero speed, it is safe to assume that vessel behavior will be similar inside them.

Finally, after the trajectory clustering process completes, the clusters are converted into convex hulls. The result is a set of convex hulls, each one annotated with the itinerary $R_{A\to B}$ it belongs

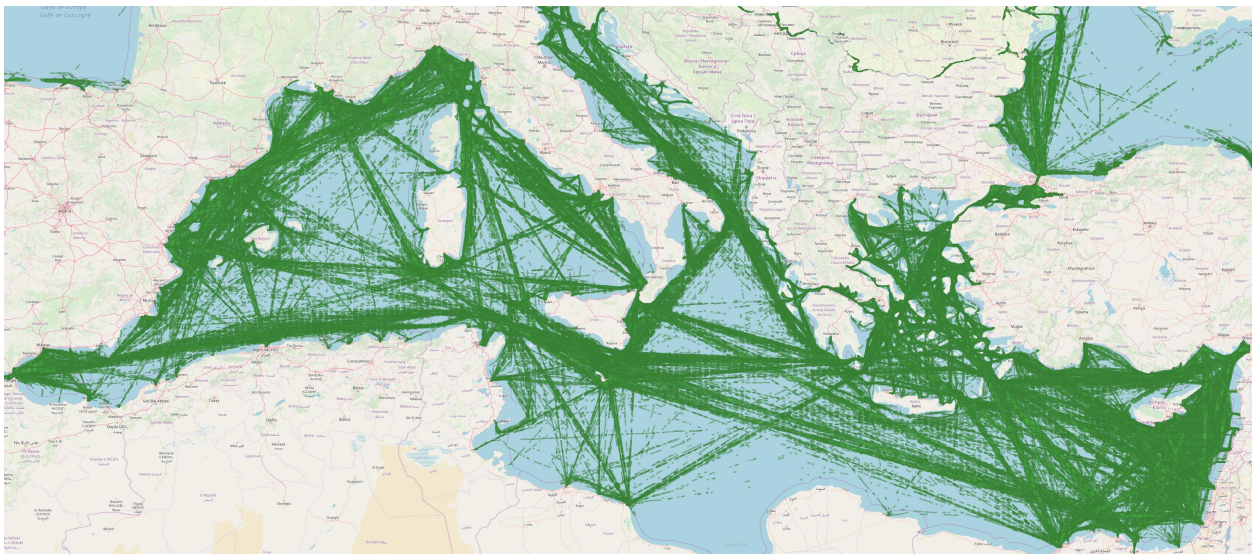


Figure 25: The output of the proposed methodology in the Mediterranean sea.

to. Therefore, each itinerary can be represented as a set of convex hulls indicating the spatial boundaries the vessels must move in when following $R_{A \to B}$. Moreover, each convex hull contains statistics for the speed and heading such as minimum/maximum speed/heading, representing behavioral boundaries for the vessels. Any deviation from these spatial/behavioral boundaries is an anomaly which requires further attention and examination from the authorities. For a visual illustration of anomaly detection in maritime routes, consider Figure 24, which demonstrates examples of normal and abnormal or deviating behavior. The green semi-transparent polygons represent the convex hulls. The red circles indicate the positions of the vessel while the red lines simply connect these positions to illustrate the path followed. Figure 24(a) illustrates a vessel travelling towards Olbia city in Northern Sardinia which moves inside the spatial boundaries of the itinerary indicated by the convex hulls. It can also be seen that vessels that travel towards Olbia city follow two paths before entering the port. Figure 24(b) illustrates a vessel that departed from Barcelona, Spain which shortly after its departure, deviates from the spatial boundaries indicated by the convex hulls. These convex hulls can be used to evaluate in real-time, the moment an AIS position is received, whether the vessel deviates from the normal behavior or not. Figure 25 visualizes the output of the proposed methodology presented in Section 6.2 applied in all Cargo vessels sailing in the Mediterranean sea representing their "patterns-of-life".

6.3 Distributed Architecture

The methodology described in the previous section requires the processing of a tremendous amount of historic AIS data. Moreover, the aforementioned algorithms described, such as the DB-Scan algorithm, perform CPU, and memory intensive operations. For that reason, the entire approach is implemented in a distributed fashion using the Apache Spark 2.4.4[25] platform.

Apache Spark is a well-known, mature, and distributed MapReduce framework which allows the in-memory batch processing of Big Data. Due to its nature and its maturity in the field of Big Data and batch processing, it is often preferred over other frameworks such as Apache Hadoop[26], Apache Storm[27], Apache Flink[28] when processing historic data. Apache Hadoop, similar to Spark, is a distributed framework that allows for the distributed processing of large data sets across clusters

[25] https://spark.apache.org/
[26] https://hadoop.apache.org/
[27] https://storm.apache.org/
[28] https://flink.apache.org/

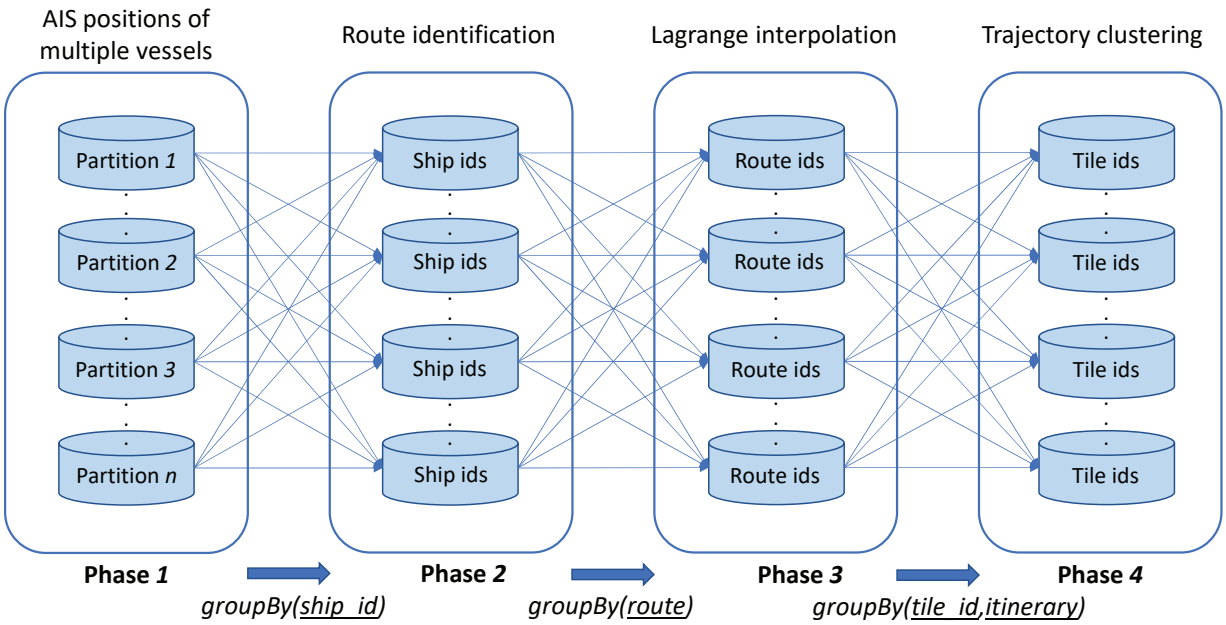


Figure 26: The distributed workflow of the proposed methodology.

of computers. It is famous for the use of the MapReduce programming paradigm and the Hadoop Distributed File System (HDFS), which stores files in a Hadoop-native format and parallelizes them across a cluster. The main difference between these two frameworks is that Spark works in memory which results in a huge performance increase. Apache storm is a framework for real-time distributed computation of data. It can process unbounded streams of data and it is the equivalent of Hadoop for real-time processing. Apache Flink is a framework for in-memory distributed stateful computations over unbounded and bounded data streams. It can work well with both batch and stream processing due to its inherent ability to treat data as streams of events. Unfortunately, although Apache Flink is promising, it lacks the maturity of Apache Spark. Finally, it is worth noting Kafka streams[29] which is a fast-emerging framework for building real-time applications and microservices. Taking into account all of the properties of these platforms, we chose Apache Spark for the implementation of our methodology because of its maturity in batch processing data.

Figure 26 illustrates the distributed workflow of the proposed methodology. Initially, the historic AIS data are distributed over n partitions with no partition strategy (Phase 1). In order to extract the routes of each trajectory and run Algorithm 5 in parallel (Phase 2), data are grouped by the ship id. Therefore, a partition contains all of the AIS data that refer to the same trajectory or ship id. In addition, a partition may contain one or more trajectories. After Phase 2 is complete and each trajectory is segmented into sub-trajectories or routes, we need to apply the Lagrange interpolation on each route (Phase 3). To this end, data are now grouped by the route id which allows for better re-balancing of data – trajectories are not segmented into an equal number of segments. Next, each AIS position is mapped to a spatial index or tile id. Each tile id may contain one or more itineraries, therefore, data are now grouped by pairs of tile id/itinerary in order to apply the modified DB-Scan algorithm to each pair (Phase 4). This results in the processing of several thousands or several millions of DB-Scan algorithms in parallel. The number of DB-Scan algorithms depends on the number of tiles and the number of itineraries per tile. Finally, after the clusters are formed, each cluster is converted to a convex hull.

[29]https://kafka.apache.org/documentation/streams/

Vessel Type	# Positions	Size	# Interpolated positions	Size
Cargo	21,049,968	2.7 GB	181,392,045	32 GB
Tanker	6,762,563	888 MB	68,028,555	12 GB
Passenger	7,229,579	974 MB	83,225,227	15 GB

Table 5: Dataset size.

Vessel Type	# itineraries	# vessels
Cargo	388,059	6,089
Tanker	117,473	2,027
Passenger	355,561	1,219

Table 6: Dataset characteristics.

6.4 Experimental Evaluation

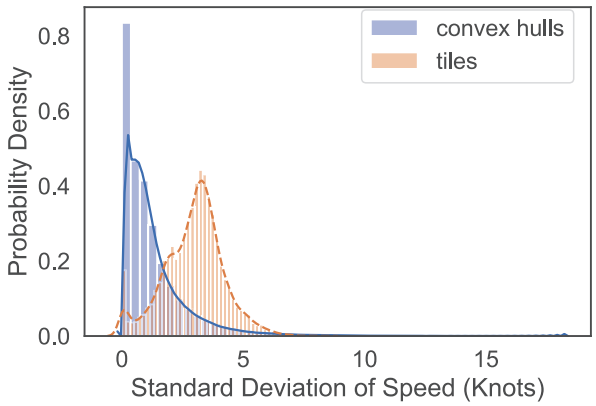
This section describes the datasets used in the experiments as well as the performance of the proposed methodology. The experimental evaluation presented below was performed on a cluster consisting of 4 machines with 4 cores (Intel(R) Xeon(R) CPU E5-2650 0 @ 2.00GHz) and 8 GB of RAM each, totalling in 16 cores and 32 GB of RAM, running Ubuntu 18.04 LTS with Java OpenJDK 1.8, Scala 2.11.8 and Apache Spark 2.4.4.

6.4.1 Dataset Description

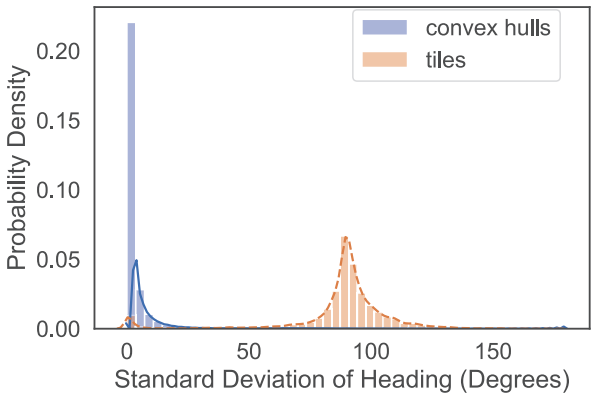
The dataset that was used was provided by MarineTraffic and contains in total $35,042,110$ AIS messages from $9,335$ vessels sailing in the Mediterranean during a three-month period (May 2016 to July 2016). The total size of the dataset is, originally, 4.4 GB. To evaluate our proposed methodology, we separated the dataset into three vessel types, namely Cargo, Tanker, and Passenger. These types of vessels typically follow predefined shipping lanes between two ports or way-points, making them suitable for the evaluation of our approach. For each vessel type and for each unique vessel identifier per type, routes were extracted as described in Section 6.2.2 and Lagrange interpolation was used in the routes as described in Section 6.2.3. The resulting dataset was comprised of $861,093$ itineraries and $332,645,827$ AIS positions, totalling in 59 GB. Table 5 shows the size of the dataset and the number of AIS positions per vessel type, while Table 6 shows the number of itineraries and the number of vessels per vessel type. In Table 5, the first two columns after the vessel type indicate the size of the dataset before the interpolation, while the next two columns indicate the size of the dataset after the interpolation was performed. To the best of our knowledge, this is the first time in the literature that a dataset of this magnitude of maritime data has been used to evaluate similar traffic models.

6.4.2 Experimental Results

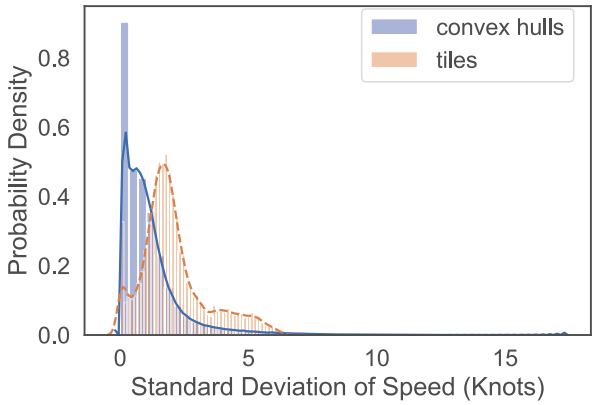
The evaluation of the proposed methodology is two-fold. First, we need to evaluate the model on how successful it is at representing the maritime traffic patterns. Second, we need to evaluate the compactness of the resulting convex hulls representing the movement of the vessels. To test the former, we used 10-fold cross-validation on the itineraries of each of the three datasets containing different vessel types, as shown in Table 6, thus always maintaining a training-test split of $90\% - 10\%$, respectively. On each fold, 90% of the itineraries were used to perform the Lagrange interpolation, the modified DB-Scan algorithm and create the convex hulls, while the rest of the itineraries were used to measure the accuracy of the convex hulls. Since each DB-Scan algorithm is performed on each tile, this results in running several thousands of DB-Scan algorithms in parallel



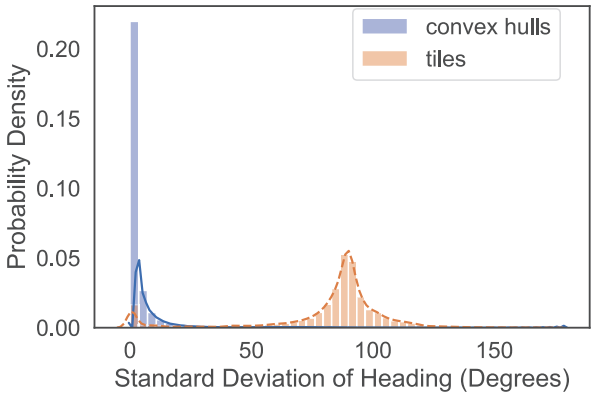
(a) Distribution plot of the standard deviation of speed of Cargo vessels.



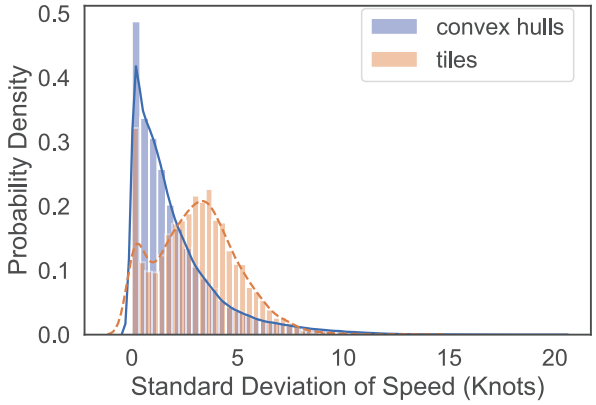
(b) Distribution plot of the standard deviation of heading of Cargo vessels.



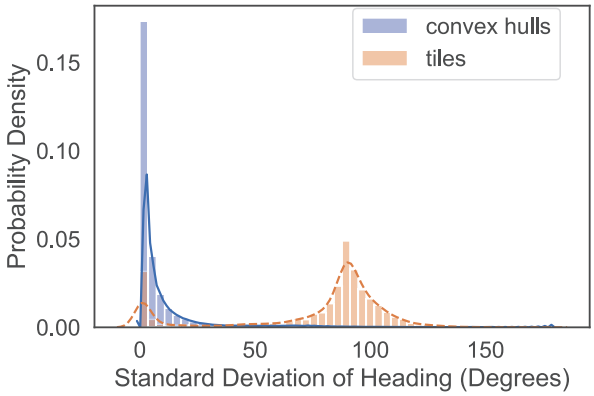
(c) Distribution plot of the standard deviation of speed of Tanker vessels.



(d) Distribution plot of the standard deviation of heading of Tanker vessels.



(e) Distribution plot of the standard deviation of speed of Passenger vessels.



(f) Distribution plot of the standard deviation of heading of Passenger vessels.

Figure 27: Distribution plots for speed and heading of each vessel type.

Vessel Type	Accuracy (%)
Cargo	98.56
Tanker	94.62
Passenger	94.4

Table 7: 10-fold cross validation results.

Vessel Type	Accuracy (%)
Cargo	97.65
Tanker	90.97
Passenger	97.95

Table 8: Accuracy results on the third month (July).

(see Section 6.3). In our case, performance is measured as in [43] and [41] as *the percentage of AIS positions of the test set that fall within the convex hulls* and use the term "accuracy" to refer to it. Specifically:

$$Accuracy = \frac{\#\ of\ AIS\ positions\ within\ the\ convex\ hulls}{\#\ of\ total\ AIS\ positions} \tag{11}$$

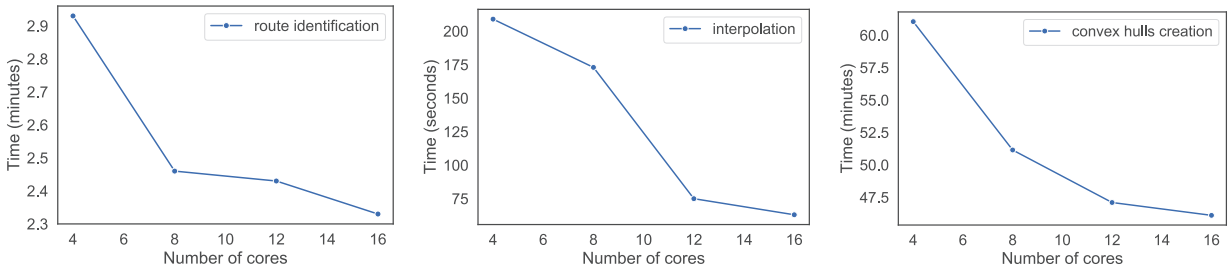
Table 7 shows the averaged accuracy achieved by our methodology for each vessel type. The results show that our approach can model, in the best case, up to 98.56% of the maritime traffic of the Cargo vessels. Passenger vessels achieve the lowest performance of 94.4% and Tanker vessels lie in the middle (94.62%). This can be explained since the number of itineraries and the number of Cargo vessels is greater compared to the other two vessel types, yielding a better representation of the maritime traffic.

To further test our approach, this time, we used as a training set the first two months of the dataset (May and June) and tested the accuracy on the remaining month (July). Similar to the 10-fold cross-validation experiment, we used the training set to perform the Lagrange Interpolation, the modified DB-Scan algorithm, and create the convex hulls, while the test set was used to measure the accuracy (Equation 11). Table 8 shows the achieved accuracy per vessel type.

To evaluate the compactness of the convex hulls we compared our methodology with a baseline approach that instead of using clusters and convex hulls, it only uses the statistics extracted per tile of the surveillance area. To show that the statistics of the convex hulls are more representative of the vessel movement and more compact, we measured the standard deviation of the speed and heading of the AIS positions they contain. Similarly, we measured the standard deviation of the same features of the AIS positions contained in each tile. We note that there may be more than one convex hull per tile. Figure 27 illustrates the distribution plots of the standard deviation of speed and heading for each vessel type. Figures 27(a), 27(c), and 27(e) illustrate the distribution plot of the standard deviation of speed, while Figures 27(b), 27(d) and 27(f) illustrate the distribution plot of the standard deviation of heading. The blue line demonstrates the distribution of the convex hulls and the orange dashed line demonstrates the distribution of the tiles. We can observe that the standard deviations of the convex hulls are gathered to the left which indicate that we have lower values for the deviations. On the other hand, the standard deviations of the tiles tend to be in the middle of the Figures, which indicate larger values of standard deviations. From Figure 27 we can infer that the convex hulls are more compact and that the AIS positions located inside them have similar speed and heading.

Finally, to demonstrate the scalability of the solution, we run for each step of the approach (route identification, interpolation, convex hulls creation), four experiments, each one with a different number of processing cores. To this end, due to the volume of the dataset and the time required for

the entire methodology to extract the final set of convex hulls, we chose the vessel type with the least number of AIS positions, the Tanker vessels. Furthermore, we run the experiments in the first month of the dataset (May) and only in 10% of the vessels of the month, thus significantly reducing the size of the dataset. Consequently, we measured the time required for each experiment to be completed and illustrated the results in Figure 28. Figure 28(a) illustrates the time required for the route identification step to be executed using 4, 8, 12 and 16 processing cores. Similarly, Figures 28(b) and 28(c) illustrate the time required for the interpolation and the convex hulls creation steps to be completed, respectively. In each step, we can see that as the number of processing cores increases, the execution time of each step decreases, demonstrating that the proposed methodology can take advantage of multiple number of CPUs in a cluster to improve the execution time.



(a) Route identification processing time.

(b) Interpolation processing time.

(c) Convex hulls creation processing time.

Figure 28: The scalability of the approach in each step.

6.5 Discussion

The proposed network abstraction model provides a compact way for representing repeating trajectories, such as the routes followed by cargo or passenger vessels. For this purpose, it performs a simplification of the route that relies on the main way-points and the edges connecting them, making it comparable to other abstraction models. Compared to [40], our methodology scales in much larger AIS datasets as shown in Section 6.4 (the Mediterranean sea, an area of $2.5 \times 10^6\ km^2$ compared to the Baltic sea, a much smaller surveillance area of $404.354\ km^2$). Moreover, [40] rely on several user-defined parameters and thresholds, which makes the fine-tuning of the maritime network more difficult.

As far as it concerns anomaly detection the proposed framework relies on the extracted network abstraction model for defining normal behaviours. In order to detect deviations, it is important for the extracted abstraction model to cover, as much as possible, the normal vessel route positions. The performance of the proposed model is at similar level to that of TREAD [43] in terms of AIS positions that fall within the extracted model against the total AIS positions. TREAD highly depends on the traffic density, thus managed to capture 95% of the total traffic in small high-density regions (e.g. the strait of Gibraltar) and much smaller percentages in regions with sparse vessel traces (e.g. 75% in the North Adriatic Sea and 40% in the Indian ocean). Our proposed methodology is able to capture 95% of the total maritime traffic in either dense or sparse regions of sea, due to the Lagrange interpolation that we perform on trajectories. Similarly, the proposed methodology of [41], which creates a graph-based representation, achieves a performance of 76.1% in the Atlantic Ocean off the coasts of Spain, France and in the English Channel.

6.6 Summary

In this work, we presented a distributed framework that captures the vessels' behavior and extracts the patterns of their movement. Specifically, the findings of the proposed approach can be summa-

rized in two major points: i) the resulting convex hulls manage to accurately represent the spatial boundaries of the vessel movement indicating that any future AIS position can be effectively predicted and placed in a certain geographic region ii) the convex hulls can effectively represent the behavior of the vessel movement in terms of speed and heading and detect possible anomalies and deviations. Although many early alerts and false alarms may arise, using the richer pattern information (direction and speed and position) as well as windowed averages of the latest vessel status it can be easier to distinguish true from false alarms [12]. It is among our future steps to further examine this problem.

Part IV
Supervised anomaly detection

7 Classification of vessel activity in streaming data

7.1 Introduction

The upsurge in mobility data volume has attracted researchers' interest in data-driven knowledge discovery and distributed data processing techniques. Discovering patterns in huge surveillance datasets is of paramount importance for delivering accurate insights on vessels' activities at sea. Today, almost all vessels worldwide are required to carry an Automatic Identification System (AIS) transponder. AIS is a global tracking system that allows vessels to be aware of vessel traffic in their vicinity and to be seen by that traffic. Through this tracking system vessels broadcast information about their location (i.e., GPS coordinates) and behaviour (e.g., speed, course, etc.), as well as information about their characteristics such as vessel size, draught and destination. Although AIS was initially designed for safety purposes and intended to assist officers on board and maritime authorities to monitor vessels' mobility, it soon became apparent that accessing vessels' mobility data can provide useful insights for the identification of illegal activities or abnormal behaviour. Such a use case is hosted by MarineTraffic.com[30] where AIS data is used to monitor vessels and extract meaningful information from their transmitted positions through an anomaly detection toolkit [10]. The toolkit consumes AIS data in real-time to search for anomalies such as deviating or abnormal vessel behaviour [122]. This toolkit can be further extended to support the classification of patterns from the trajectory data. By employing trajectory classification techniques, it is possible to classify or match a vessel mobility pattern to a set of predefined labels. Such a label can be the vessels' fishing activity during which they tend to perform characteristic patterns in their trajectories.

Consequently, such an extension of the anomaly detection toolkit can be used for the immediate identification of Illegal, Unreported and Unregulated (IUU) fishing activities in prohibited areas or nature protection areas which has gained much attention the recent years. A study showed that 640,000 tonnes of ghost gear is left in the world's oceans each year, which entangles and kills around 136,000 turtles, whales, seals, birds, and other sea animals[31]. These animals end up suffering long and painful deaths. The US government has established the Seafood Import Monitoring Program to address the issue[32]. The Food and Agriculture Organization (FAO) of the United Nations has commissioned a study of IUU fishing activities to determine whether the Organisation should provide guidance for the future estimation of IUU fishing activities[33]. Furthermore, specific fishing activities (e.g., trawling) and fishing gear are strongly linked to the indigenous fauna, thus a mechanism that identifies such operations is of utmost importance. For that reason, several techniques have been developed that take advantage of the spatio-temporal features of trajectories.

In spatio-temporal analysis, data mining includes trajectory pattern clustering, frequent pattern discovery, trajectory classification, forecasting and outlier detection. Trajectory classification is a process of creating a model that can match the mobility pattern of an object to a specific label based on certain decision criteria [6]. Though several trajectory classification solutions have been proposed and applied in many mobility applications, less focus has been given in the maritime domain and specifically in classifying a vessel's type [6, 123, 26, 124], characterising shipping operation areas (e.g., fishing areas [64]), or discovering vessel activity such as fishing [125, 67], search and rescue operations [25, 126] based on its trajectory.

However, many of those methods assume a-priori availability of the whole dataset (i.e., batch processing), have limited availability of ground truth data and use features that make them inapplicable to streaming applications. More specifically the training data are specific to limited vessel types (e.g., only fishing) or the features are linked to trajectory information such as departure and

[30] https://www.marinetraffic.com/anomaly-detection

[31] https://www.worldanimalprotection.org/illegal-fishing-threatens-wildlife

[32] https://www.worldwildlife.org/threats/illegal-fishing

[33] http://www.fao.org/iuu-fishing/tools-and-initiatives/iuu-fishing-estimation-and-studies/en/

arrival port, that can be discovered only through batch analysis of historical records (i.e., after the completion of a voyage).

Although it is possible to perform event detection in streaming data, this has been realised only for simple events such as proximity events and route deviations [10], potential vessel spoofing [9] or intentional AIS switch-off [11]. Performing complex event analysis (such as distinguishing vessels' activities at sea) poses significant challenges when applied in streaming data (e.g., limited execution time, memory consumption, data cleaning, etc.).

This chapter presents a novel approach for classifying vessel activity in large volumes of AIS data streams. Each day approximately 46GB of AIS data are generated from up to $200,000$ vessels worldwide, the velocity of which can reach up to $16,000$ events per second. For this reason, our proposed classification method exploits the benefits of Akka[34] (which is an open source lightweight framework) in order to develop a distributed system that can efficiently handle large volumes of data and predict vessels' activities in real-time based on their data streams. Our system relies on the well-known Lambda architecture [127] in an attempt to balance latency and throughput for the 4Vs of the Big Data (i.e., Volume, Velocity, Veracity and Variety). More specifically, a 'batch-processing' layer is responsible to train the classification model, which is then used to distinguish the vessel activities in the 'stream-processing' layer at which data continuously arrive from vessels at high rates. Our approach is tested against vessels sailing at the seas of Northern Europe in 2018 and is capable of effectively distinguishing between trawling and longlining operations with high accuracy of over 90%.

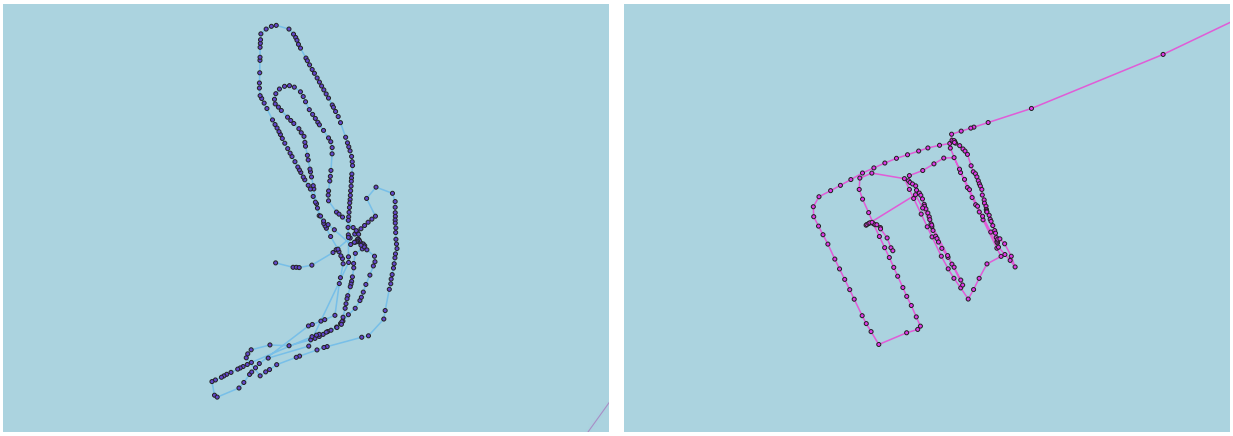
The rest of the chapter is structured as follows. Section 7.2 describes in detail the proposed methodology for the classification of fishing activities. Specifically, in Section 7.2.1, a brief analysis of fishing vessel behaviour in such activities is presented. Section 7.2.2 presents how the fishing behaviour is processed to create the classification model, while Section 7.2.3 presents the two layers of the approach, namely the 'batch-processing layer' and the 'real-time processing layer'. Finally, Section 7.3 illustrates the experimental evaluation of our approach and Section 7.4 summarizes the merits of our work.

7.2 Proposed Methodology

In this section we describe the investigated fishing methods (i.e., trawling and longlining) and the vessels' behaviour and mobility patterns when engaged in fishing activities. Furthermore, we thoroughly detail the features selected for the proposed classification model and introduce a lambda-architecture scheme in which the classification model can be applied for online fishing activity detection.

[34] https://akka.io/

7.2.1 Fishing patterns



(a) The movement pattern of a trawling trajectory. (b) The movement pattern of a longlining trajectory.

Figure 29: The movement patterns of fishing activities.

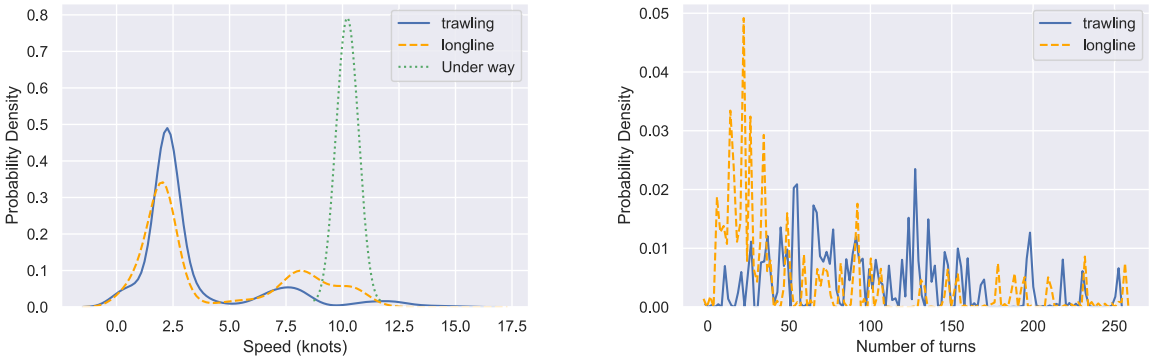
Two different fishing methods have been studied in our work, namely trawling and longlining. Specialised gear equipment is used in each fishing method, leading to different mobility behaviour while fishing. Understanding the nature of each operation will give insights on the characteristics distinguishing one activity from the other.

Trawling: Fishing vessels engaged in trawling activity use a fishing net located in the stern of the boat, called trawl, which is dragged through the water. The net is typically pulled by one or more fishing vessels, either on the sea floor (i.e., bottom trawling) or mid-water (i.e., mid-water or pelagic trawling), although single-boat trawling is usually the case. In single-boat trawling the spread of the net depends on the trawl doors, also called "otter boards" and act as wings. To keep the wings steady the vessel must be travelling at a constant speed and for that reason, during trawling, vessels usually sail with lower steady speeds.

Longlining: During this type of activity vessels set multiple fishing lines with baited hooks attached to them, called snoods. The length of the fishing lines can reach up to a kilometer, while the total length of all the fishing lines in the entire activity can reach up to several kilometers [128]. The lines can be deployed either near the surface (i.e., pelagic longline) or along the sea floor (i.e., demersal longline). While setting the lines, vessels travel at their steaming speed or slightly less and they maintain a constant speed. When all lines are set, they are left in the water for several hours and the vessel drifts slowly with them. To retract the lines, vessels follow the same procedure as when setting the lines. The process of setting the lines, waiting and retracting the lines can be repeated several times before returning to a port.

In both fishing methods vessels maneuver and make frequent turns to remain in the same fishing area of interest. However, longlining is characterised by long straight-line trajectories followed by sudden turns. This leads to less maneuvers compared to trawling when the vessels are observed over the same amount of time. At this point, it should be noted that both fishing activities may take up to several hours or even days.

To understand better the behaviour of the fishing activities, their movement patterns have been unveilled in Figures 29 and 30. Figure 29 depicts two movement pattern paradigms of the studied fishing activities. In Figure 29(a) the trajectory of a trawling vessel is illustrated and shows that the trajectory is characterised by frequent and irregular turns. Similarly, in Figure 29(b), which illustrates the trajectory of a longlining vessel, it is apparent that the trajectory, despite the irregular and frequent turns, included also long and straight lines.



(a) Speed distribution of fishing vessels.

(b) Distribution of number of turns.

Figure 30: Distribution of speed and number of turns.

Figure 30(a) illustrates the activities' speed kernel density distribution. The distribution was calculated from the AIS messages transmitted by the fishing vessels during January and February of 2018. The blue lines refer to the trawling activity, the orange dashed lines refer to the longlining activity and the green dotted lines refer to non-fishing activity. From Figure 30(a) we can observe that both trawlers and longliners follow similar speed patterns. Both types of fishing activity have two peaks in their speed distribution, the first one being in the range of 1 to 3 knots and the second one being in the range of 7 to 10 knots. A closer inspection though, reveals that the probability density of trawlers' speed being in the first peak is higher than the probability density of longliners' speed in the same peak. A similar pattern can be observed for the second peak as well but with the longliners' speed now having higher probability density. Furthermore, in longlining activity, the first peak is slightly shifted to the left and the second peak to the right compared to the trawling activity. This can be explained because longliners have higher speeds when setting the lines and the duration of the process takes up a large part of the longlining activity. Moreover, they have lower speeds (range of 1 to 3 knots), compared to the trawlers, after setting the lines and before retracting them, which indicates that they remain stationary drifting along with the lines. Finally, Figure 30(a) shows that when vessels travel from the ports to the fishing areas and vice versa, they have higher speeds (range of 9 to 12 knots).

Figure 30(b) illustrates the kernel density distribution of the number of turns per fishing activity. The blue line represents the trawling activity and the orange dashed line represents the longline activity. It can be observed that there is not a clear distinction between the two fishing activities. However, a closer look reveals that probability density of trawling activity follows a bimodal distribution with two modes at 50 turns and 125 turns while the probability density of longliners is a right skewed normal distribution. This phenomenon indicates that, during the longline activity, vessels tend to turn less compared to the trawling vessels. This is due to the fact that longliners, as already explained in the previous paragraphs, have long straight-line trajectories.

7.2.2 Classification of trajectory patterns

To create a proper classification model, several features that are capable to capture the observed behaviour described in Section 7.2.1 have been selected. Authors in [63] presented a set of features based on speed for the classification of various fishing vessel types with the use of XGBoost classifiers on VMS data, yielding results of high accuracy. Among these features, the average speed and its standard deviation are of high importance. Following their footsteps, we selected some of their features and we extended them to be able to capture all of the trajectory characteristics described in

the previous Section. The features selected do not require batch analysis of data and can be computed online over streaming data coming in the Akka system in real-time. The selected features fall into three dimensions; vessels' speed, vessels' drifting and turn frequency.

Features based on speed: Fishing vessels when engaged in fishing activity tend to maintain a constant speed without any significant deviations. Therefore, the *Average Speed* and the *Standard Deviation* of speed play an important role in the identification of the activity.

- *Average Speed:* The average speed during a vessel's trajectory indicates the value of speed most vessels have during the fishing activity, which, based on Figure 29(a), revolves around 2.5 knots, especially in the trawling activity.
- *Standard Deviation:* The standard deviation is able to reveal whether the speed is constant or not during a vessel's trajectory. A standard deviation close to 0 indicates the steadiness of the speed.

Features based on drifting: The drifting can be inferred from the difference between the course over ground (Cog) and the heading of the vessel. The course over ground represents the actual direction the vessel has along its path, while the heading represents the direction of the vessel's bow. Figure 31 visualises an example of drifting. The two of them might differ due to the effects of wind, tide or currents of the sea. Two features that may indicate such behaviour are the *Average Drifting* and the *Standard Deviation of Drifting*.

- *Average Drifting:* Excessive drifting is indicative of the behaviour longline vessels have after they set the lines.

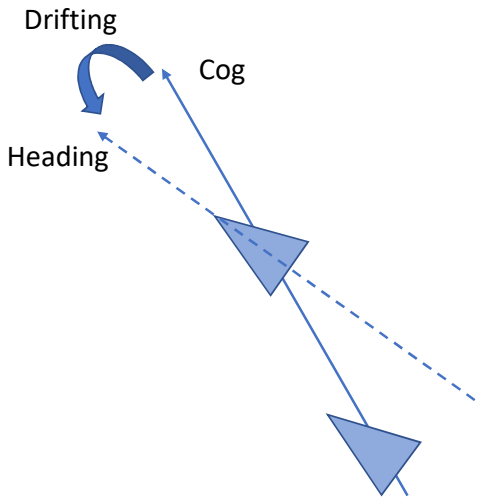


Figure 31: Example of drifting.

Features based on turns: Frequent turns are apparent on fishing vessels regardless of the fishing method (i.e., trawling or longlining). Although vessels during both activities seem to have a similar behaviour, three features that are able to reveal hidden characteristics of the patterns have been selected, namely *Standard Deviation of Cog*, *Number of Turns* and *Accumulated Angle*.

- *Standard Deviation of Cog:* This feature indicates that the vessel does not maintain a steady course, thus it does not move in a straight line.
- *Number of Turns:* As the name suggests, it shows the number of turns the vessel made during a trajectory.
- *Accumulated Angle:* When the vessel starts turning, either right or left, it has a certain Cog c_1. Again, when the vessel stops turning, the Cog has another value c_2. Each turn t is terminated when another turn of the opposite direction starts (e.g., a left turn stops when a right turn starts) or when the vessel maintains a steady course. The difference between these two values is the angle of the turn $a = c_1 - c_2$. The sum of all angles of all turns is the *Accumulated Angle* $A = \sum_{i=1}^{i=t} a_i$. This feature indicates how much the vessel turned during its trajectory.

The next step, after all features have been extracted from representative trawling, longlining and under way trajectories, is to use an algorithm able to effectively classify unseen trajectories based on the features given as a training set. To this end, we chose the Random Forest (RF) classification algorithm due to its high performance results in the maritime domain [129, 26] and because RFs combine the predictions of many Decision Trees into one model, thus they are less prone to overfitting [130]. Furthermore, a Random Forest classifier is less computationally expensive creating a good basis for online predictions or online re-training. Finally, Random Forests require far less data than other state of the art algorithms such as Neural Networks and it is easier to interpret their predictions based on the features [131], which in an industry setting, such an interpretation gives meaning to interested stakeholders.

7.2.3 Rea

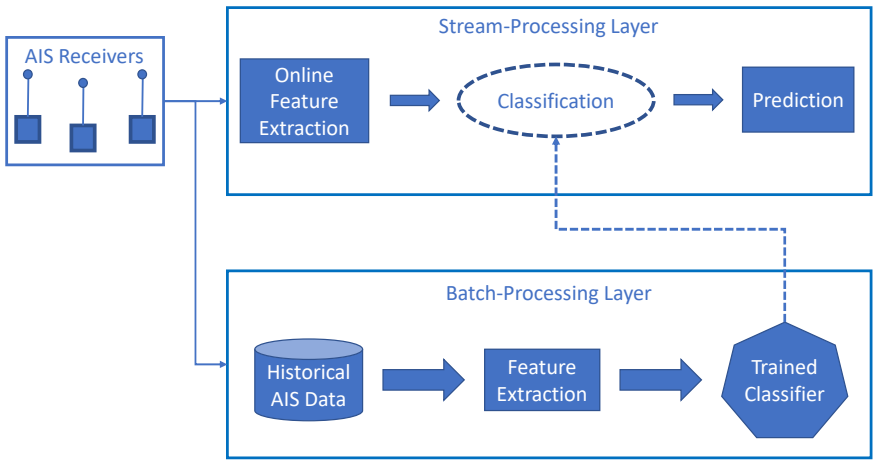


Figure 32: System architecture.

Low latencies and high throughput in a streaming system that supports fast decision making are of utmost importance. Specifically, events must be predicted in real-time, meaning the moment a new message is consumed, a prediction must be provided with a sub-second latency. In practice, websites such as MarineTraffic, are flooded by more than $16,000$ AIS messages per second received from over $4,000$ terrestrial AIS receivers (without satellite receivers in the equation). These volumes of data originate from almost $200,000$ vessels globally, totalling in a more than 50GB per day increase rate. Therefore, an architecture needs to be developed that is able to balance latency and throughput. To this end, we used the λ-architecture developed in previous work [10] which achieves sub-second latencies and high throughput. This architecture allows the reduction of the performance cost of the online computations by taking into account pre-computed results.

Specifically, the λ-architecture relies on two layers, the "batch-processing layer" and the "stream-processing layer". The former is responsible for creating pre-computed views of data which in our case is the classification model. The latter is responsible for processing streaming events and providing views into the most recent data by taking advantage of the views from the batch-processing layer, hence predictions based on the classification model. To this end, we extended the batch-processing layer of the previously developed architecture by adding a step responsible for constructing the classification model. In this section, both the "batch-processing layer" and the "stream-processing layer" are described.

Batch-processing layer: This layer is responsible for the construction of the Random Forest classification model. For the training of the model, ground truth trajectories are required. Thus, this layer consumes as input already labeled trajectories, namely trawling, longlining and underway. Subsequently, features are extracted per trajectory and the RF model is created and saved to an "xml" file which is then consumed by the stream-processing layer. The Statistical Machine Intelligence and Learning Engine (SMILE)[35] library was used for the implementation. SMILE is a Scala library that supports a wide range of supervised and unsupervised learning algorithms. Internally, it uses a "DataFrame" structure similar to the one used in the Pandas[36] python library. The benefits of this library is its performance in terms of speed and memory consumption and its compatibility with our existing architecture which is implemented in Scala. According to a third party benchmark[37], SMILE outperforms R, Python, Spark, H2O and xgboost significantly.

Stream-processing layer. In order to achieve high-throughput, low-latency performance, the stream-processing layer is implemented in the Akka framework which takes advantage of the Actor model [18]. Actors in the Akka framework exploit the concurrency capabilities of threads, they are lightweight and millions of actor instances can be deployed in a single machine since they have a small memory footprint (i.e., 2.5 million actors per GB of heap). To assist the implementation of the classification of fishing activity, one type of actor from the architecture is used, namely the Vessel Actor.

The Vessel Actor is responsible for consuming AIS messages originating from a single vessel and classifying parts of its trajectory based on the model created from the batch-processing layer. Thus, each vessel has a dedicated actor which is created after the first AIS message has been received. The actor remains idle and acts only when a new message is received. To classify parts of a trajectory of a vessel, each actor takes into account features extracted from an event-based window of user-defined length. To reduce memory consumption, features are extracted online, thus the need of storing a batch of AIS messages is eliminated. To demonstrate the online feature extraction, we present the calculation of the moving average and the standard deviation of speed. The average value of speed is given by Formula 12:

$$\overline{x}_n = \frac{\sum_{i=1}^{i=n} x_i}{n} \tag{12}$$

which can be broken down to the average of $n-1$ speed values plus a speed value of a newly received AIS message x_n, where n is the total number of messages received in our window:

$$\overline{x}_n = \frac{(\sum_{i=1}^{i=n-1} x_i) + x_n}{n} \tag{13}$$

Since the average value equals to the sum divided by the total count:

$$\overline{x}_{n-1} = \frac{\sum_{i=1}^{i=n-1} x_i}{n-1} \Rightarrow \sum_{i=1}^{i=n-1} x_i = \overline{x}_{n-1}(n-1) \tag{14}$$

[35] https://haifengl.github.io/
[36] https://pandas.pydata.org/
[37] https://github.com/szilard/benchm-ml

we can substitute Equation 14 to Equation 13 which results in:

$$\overline{x}_n = \frac{(n-1)\overline{x}_{n-1} + x_n}{n} \Rightarrow \overline{x}_n = \bar{x}_{n-1} + \frac{x_n - \overline{x}_{n-1}}{n} \tag{15}$$

where x_{n-1} is the average speed value of $n-1$ AIS messages. Upon the arrival of the n-th AIS message, we calculate $\frac{x_n - \bar{x}_{n-1}}{n}$ and add it to the average speed value of $n-1$ AIS messages using Equation 15. When the window closes, all values are reset to zero and the online calculation of the average value starts over. To calculate the standard deviation in a streaming fashion, we employ B. P. Welford's method [132]. Similarly to the calculation of the moving average, we need to reach in a state where we have the standard deviation of $n-1$ values and we need to recalculate for the n^{th} value. To begin with, the formula of variance, which is the square of standard deviation, is given by Equation 16:

$$s^2 = \frac{\sum_{i=1}^{i=n}(x_i - \overline{x}_n)^2}{n-1} \tag{16}$$

Then, we can multiply both sides by $n-1$ and define the first part of the equation as d_n^2:

$$(n-1)s^2 = \sum_{i=1}^{i=n}(x_i - \overline{x}_n)^2 \Rightarrow d_n^2 = \sum_{i=1}^{i=n}(x_i - \overline{x}_n)^2 \tag{17}$$

Later, we can apply the following identity to Equation 17:

$$(a-b)^2 = a^2 - 2ab + b^2 \tag{18}$$

which results in:

$$d_n^2 = \sum_{i=1}^{i=n}(x_i^2 - 2x_i\overline{x}_n + \overline{x}_n^2) \Rightarrow d_n^2 = \sum_{i=1}^{i=n} x_i^2 - 2\overline{x}_n \sum_{i=1}^{i=n} x_i + \overline{x}_n^2 \sum_{i=1}^{i=n} 1 \tag{19}$$

Similarly to the calculation of the moving average, since we know that $\sum_{i=1}^{i=n} 1 = n$ and that the total equals to the mean times the count, we get the following:

$$d_n^2 = \sum_{i=1}^{i=n} x_i^2 - 2n\overline{x}_n^2 + n\overline{x}_n^2 \Rightarrow d_n^2 = \sum_{i=1}^{i=n} x_i^2 - n\overline{x}_n^2 \tag{20}$$

After reaching Equation 20, we can get the value of d^2 for the first $n-1$ values, resulting in:

$$d_{n-1}^2 = \sum_{i=1}^{i=n-1} x_i^2 - (n-1)\overline{x}_{n-1}^2 \tag{21}$$

By subtracting Equation 21 and Equation 20 and after a few rearrangements to the equations we have the resulting equation:

$$d_n^2 = d_{n-1}^2 + (x_n - \overline{x}_n)(x_n - \overline{x}_{n-1}) \tag{22}$$

As with Equation 15 of the moving average, we have a relation which allows us to calculate the new d^2 value by adding an increment, $(x_n - \bar{x}_n)(x_n - \bar{x}_{n-1})$, to its previous value, d_{n-1}^2. We can retrieve the variance by dividing d_n^2 with $n-1$, which subsequently gives us the standard deviation s_n, since s_n is the square root of the variance:

$$s_n^2 = \frac{d_n^2}{n-1} \Rightarrow s_n = \sqrt{\frac{d_n^2}{n-1}} \tag{23}$$

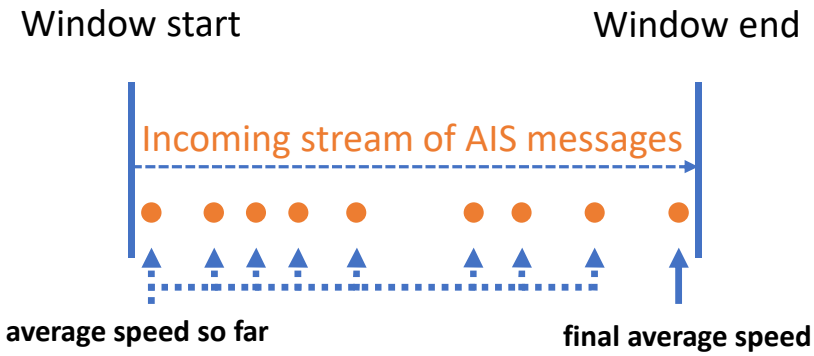


Figure 33: Online calculation of the average speed.

Again, when a window is complete, all values are reset and the calculations start over for the next window. After all features have been extracted, they are fed to the RF model which outputs a decision along with its probability, indicating the activity the vessel performs at the current window. Figure 33 illustrates the online calculation of the average speed. At each incoming AIS message a new average is calculated. The new average speed calculated based on the last AIS message, represents the average speed of the entire window. When the last AIS message of the window is received, the classification is triggered. Finally, Figure 32 illustrates the system architecture. The batch-processing layer extracts features from the historical AIS data and trains a classifer. The stream-processing layer consumes AIS messages from the receivers and makes a prediction by using the pre-trained classifier generated from the batch-processing layer.

7.3 Experimental Results

This section describes the dataset used to train and test the Random Forest classifier and its achieved classification performance. The achieved latency performance of the distributed and streaming architecture described in Section 7.2.3 has already been discussed in previous work [10].

7.3.1 Dataset Description

The dataset that was used was provided by MarineTraffic and contains AIS messages during a two-month period, from January 1^{st}, 2018 to February 28^{th}, 2018. The dataset covers the seas of Northern Europe, specifically the Irish sea, the North sea and the Baltic sea and contains high quality AIS information without gaps of information. The AIS transmits two kinds of messages, positional and static[38]. Positional messages transmit information about the vessel's location, speed, heading and navigational status. The navigational status, which is manually inserted by the vessel's crew is an identifier that indicates the vessel's activity (e.g., 1 indicates that the vessel was anchored when the message was received). Static messages transmitted include information regarding the vessel's name, its dimensions, the location of its on board positioning system antenna and its destination. The destination denotes the port or area that the vessel is headed but it may also denote the activity the vessel is currently engaged in. Again, this kind of information is set manually by the crew. The AIS messages used for our ground truth dataset contain fishing activities that have been extracted from vessels which have set their navigational status to 7, which indicates "fishing activity", and by vessels which have set their destination to *Trawling* or *Longlining* for the corresponding activity. These fishing trajectories are then segmented to fishing or non-fishing segments. The total number of AIS messages sums up to $61,050$. Table 9 shows the number of AIS messages per activity. Although, the number of messages per activity varies, the number of events per activ-

[38] https://help.marinetraffic.com/hc/en-us/articles/205426887-What-kind-of-information-is-AIS-transmitted-

Activity	# AIS messages
Trawling	16, 110
Longline	8, 484
Underway	36, 456

Table 9: Number of AIS messages per activity.

number of trees	maximum depth	Accuracy	Precision	Recall	F1-score
10	2	72 %	64.02 %	68.05 %	65.97 %
50	2	74.66 %	62.07 %	65.01 %	63.5 %
100	2	77 %	64.5 %	66.22 %	65.34 %
10	5	89.66 %	84.76 %	88.24 %	86.47 %
50	5	90.66 %	88.13 %	89.81 %	88.96 %
100	5	91.33 %	85.64 %	88.22 %	86.91 %
10	10	92.66 %	89.02 %	88.75 %	88.88 %
50	10	93.33 %	87.71 %	91.91 %	89.76 %
100	10	95.33 %	93.91 %	93.15 %	93.52 %

Table 10: Classification results per RF hyperparameter.

ity remains the same. This is due to the transmission rate of the AIS protocol. When vessels travel at high speeds, the frequency can get as high as one message every two seconds, while the lowest frequency can get as low as one message every 3 minutes when the vessels are not moving. Since the underway vessels travel at much higher speeds, the number of AIS messages is also increased. Another factor affecting the AIS transmision rate is the vessel's turn frequency. Since trawling activity is characterised by frequent turns, the number of AIS messages will be higher compared to the longlining activity but not higher than the messages of the underway activity since the speed during trawling is much lower.

7.3.2 Experimental Evaluation

In this section we provide the evaluation results of the proposed classification scheme for the identification of trawling and longlining activities. Firstly we evaluate how various hyperparameters of Random Forests influence the performance and accuracy of the proposed classifier providing well-known metrics (e.g., f1-score, accuracy etc.) for multiple Random Forest configurations. Then, we evaluate the robustness of the classifier when applied in various temporal window lengths. Finally, we compare the classification performance of Random Forests against other well-known classifiers. In our first series of experiments we used the complete trajectories[39] of the dataset from which features are extracted following the methodology described in Section 7.2.2. Then, these features are fed to a Random Forest classifier, tuning in each experiment a different hyperparameter of the forest. The first hyperparameter selected is the number of trees of the forest. We use three distinct configurations for the number of trees, $numTrees = 10, 50, 100$, and run 5-fold cross validation per setting. The idea behind the Random Forest algorithm is to optimise the prediction of an instance by averaging the predictions of multiple decision trees. Moreover, for each configuration of *numTrees*, we tested three distinct configurations of another hyperparameter of Random Forest, the maximum depth, setting its value to 2, 5 and 10 respectively. This parameter controls the size of the trees, specifically defining the maximum number of levels each tree in the forest can have. Table 10 shows the results of each experimental setting. From the results, we observe that the configuration with the highest classification performance is $numTrees = 100$ and $maxDepth = 10$ that

[39]The length of the trajectories can last from few hours to several days.

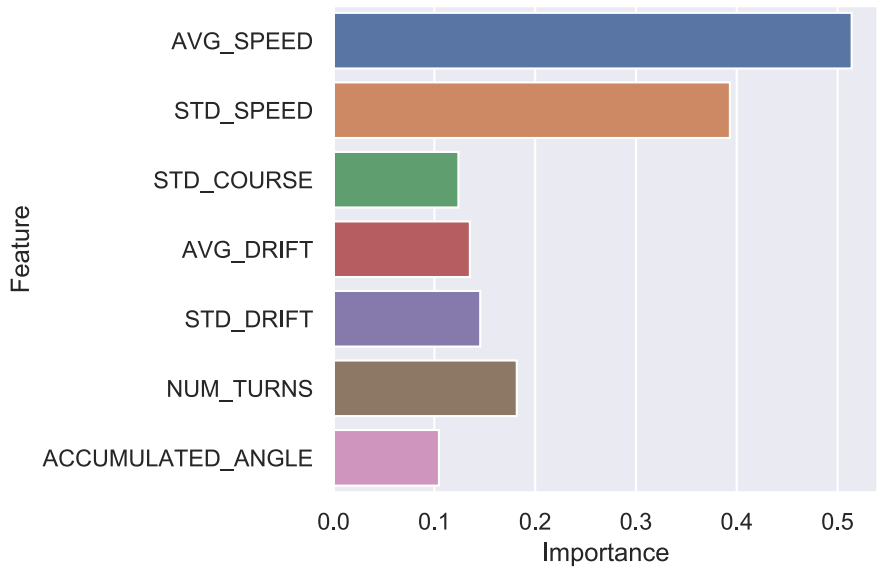


Figure 34: Feature importances.

Hours	Accuracy	Precision	Recall	F1-score
1	82.48 %	82.47 %	82.79 %	82.63 %
2	83.33 %	82.84 %	83.06 %	82.85 %
4	82.53 %	82.83 %	82.55 %	82.69 %
8	84.68 %	84.55 %	84.36 %	84.46 %
12	86.94 %	85.88 %	86.4 %	86.14 %
24	89.54 %	85.06 %	87.38 %	86.2 %

Table 11: Macro average results per temporal size.

achieves f1-score of 93.52% and accuracy of 95.33%. Therefore, for the rest of the experiments we used the best setting from Table 10.

Next, we measured the importance of each feature for our classifier and visualised the results in Figure 34. It can be seen that all the features are approximately of equal importance with the exception of the average speed and the standard deviation of speed. This means that all of the features contribute equally to the decision of the classifier, but the features that contain the speed factor are more decisive.

Afterwards, we evaluated the performance of our classifier when applied in time windows of different length. More specifically, we segmented each trajectory into temporal segments of 1, 2, 4, 8, 12 and 24 hours. This means that a fishing activity, e.g. trawling is now segmented into more parts, according to the temporal size.

For each temporal size we performed 5-fold cross validation, keeping 80% of the trajectories as a training set and the rest 20% as a test set. The macro average results after the cross validation are presented in Table 11 and Figure 35. From the results, we can observe that the accuracy is increased as the time window length is increased. Furthermore, it can be seen that from the one-hour window length to the four-hour, the accuracy does not show any significant change, while from the four-hour window length to the twenty-four-hour, the accuracy increases abruptly, from 82.53% to 89.54% (i.e., a 7.01% increase). This can be explained due to the fact that a fishing pattern may take hours to form. The features of each trajectory start to become distinguishable from one another after several hours have passed. From our experiments, we can see that the time threshold with which a pattern in the fishing trajectory is formed needs to be at least four hours which explains the increase in the accuracy.

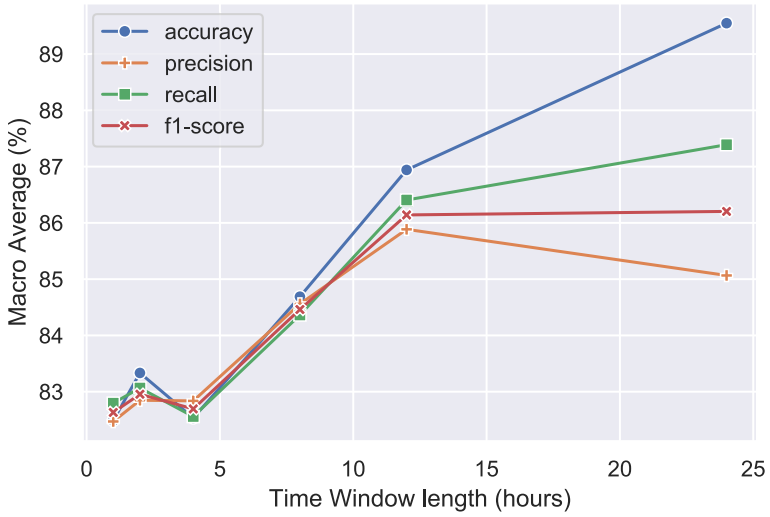


Figure 35: Macro average results per temporal size.

Classifier	Accuracy	Precision	Recall	F1-score
Random Forests	89.54 %	85.06 %	87.38 %	86.2 %
Gradient Boosted Trees	90.98 %	86.77 %	87.83 %	87.29 %
Linear Discriminant Analysis	81.4 %	69.13 %	77.71 %	73.17 %
Logistic Regression	79.71 %	76.19 %	75.59 %	75.89 %

Table 12: Macro average results per classifier.

Finally, we compared the Random Forests classifier against three other well-known classifiers, namely Gradient Boosted Trees (GBT), Linear Discriminant Analysis (LDA), and Logistic Regression, on the same set of features. To this end, we performed 5-fold cross validation, similar to the previous series of experiments of Table 11 and set the temporal size of the trajectories to a fixed length of 24 hours, the maximum length of the trajectories that yields the best classification performance. For the Gradient Boosted Trees we used the same hyperparameters to the Random Forests, since both of these classifiers use multiple decision trees to make a prediction, thus making a more direct comparison between the two. For the rest of the classifiers, we used the default hyperparameters that are provided by the Smile library. Table 12 presents the macro average results for each classifier. According to Table 12, Gradient Boosted Trees slightly outperform the Random Forests, yielding the best classification performance (a F1-score of 87.29 %), while both the LDA and the Logistic Regression perform weakly, achieving a F1-score of 73.17 % and 75.89 %, respectively. Despite the fact that Gradient Boosted Trees present a higher classification performance, they are computationally more expensive and are harder to fine-tune [133], making them the least favorable option for a real-time streaming system. A more in-depth comparison between the classifiers by fine-tuning the classifiers' hyperparameters will be conducted in the future for a more thorough evaluation.

Three studies in the literature are comparable to our own methodology in terms of classification performance [67, 68, 63]. Authors in [63] present a set of features that is used from XGBoost classifiers achieving an accuracy of approximately 97%. Despite their high classification accuracy, several features are best suited for the region of China and are not applicable to other regions, making it unsuitable for global fishing detection. Moreover, their methodology extracts features

based on entire trajectories of nine fishing vessel types, thus their goal is to identify the vessel type and not the vessel activity. Souza et al. [67] create three different classifiers for the detection of trawlers, longliners and purse seiners respectively, achieving a median accuracy of 83%, 84% and 97% for each activity correspondingly. The main drawback of their approach is the need of three separate classifiers, each one performing a binary classification task of fishing and non-fishing activity for each vessel type, compared to our methodology where it acts as a universal classifier of fishing activity. Finally, authors in [68], use autoencoders to detect whether a longliner vessel is engaged in fishing activity, achieving an accuracy of 85%. Similarly to [67], they perform a binary classification task instead of the multi-class classification task of our approach.

7.4 Summary

In this work we presented a methodology for the classification of fishing activities in a streaming fashion. Specifically, the fishing behaviour was analysed which led to the selection of a specific set of features able to describe and characterise two fishing activities, trawling and longlining. Furthermore, an approach of online feature extraction and classification was presented which eliminates the need of storing streaming window batches in memory. Experimental evaluation showed the analysis of the hyperparameter tuning of our classifier along with the importance of the selected features. Moreover, the evaluation of our approach demonstrated the increased classification performance and the way the temporal size of the trajectory affects the classification accuracy. Finally, a comparison between four classifiers was made on the same set of features, demonstrating that, in our case, decision-tree based classifiers yield a better classification performance.

8 A computer vision approach for trajectory classification

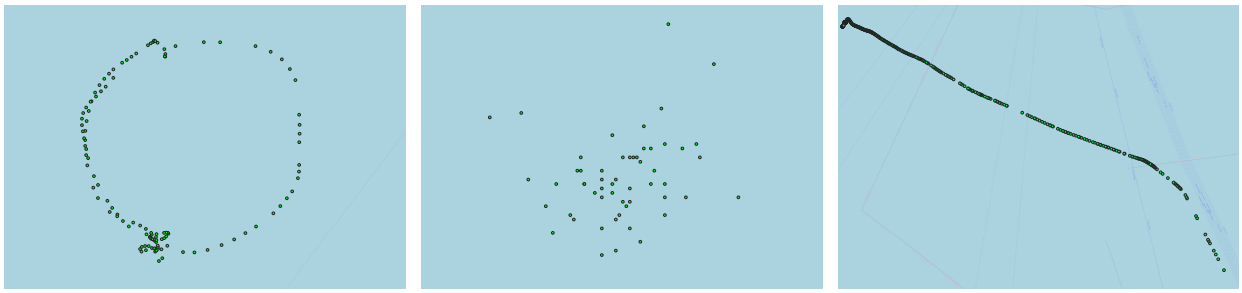
8.1 Introduction

In recent years, the large volume of mobility data generated from thousands of tracking devices has attracted researchers' interest in data-driven, knowledge discovery techniques. Such techniques, that often include anomaly detection and trajectory classification, are able to detect patterns from the generated mobility data. Mobility data consist of trajectories of moving objects and can provide spatio-temporal characteristics that indicate anomalous or suspicious behavior. Today, although all larger vessels are obliged to carry an Automatic Identification System (AIS) transponder on board, smaller vessels have also adopted this technology. AIS is a vessel tracking protocol that allows vessels to broadcast information about their whereabouts and identity and to receive information by nearby vessels as well.

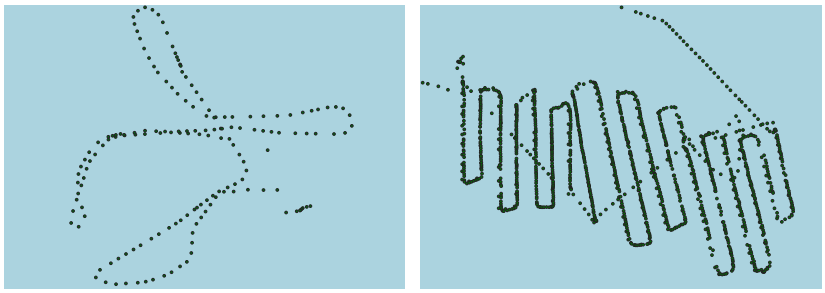
Characterizing parts of the trajectories can help the authorities in decision-making, thus improving the overall Maritime Situational Awareness (MSA). To promote the MSA, several supervised or unsupervised learning techniques were developed that take advantage of kinematic and spatio-temporal features of trajectories. In the literature, the field of trajectory classification has been extensively studied, but only in recent years the focus has shifted towards the maritime domain [6, 124, 64, 125, 67, 126, 14]. In all those studies, the context of the analysis is typically the physical world and the geography. Latitude and longitude are the basic features in a possibly multi-dimensional space (others can be the speed, direction, etc). However, experts rely heavily in the visualization of trajectories to manually identify parts of the trajectory that are of some importance. This provides an intuition to move the analysis in a different domain, leveraging computer vision techniques on classification. In computer vision, the most commonly used techniques include convolutional neural networks [134, 135]. Before the widespread usage of neural networks, other techniques were also devised where researchers tried to extract features from an image or to predict textual information from pixels [136, 137]. One of the most common goals of computer vision and specifically image recognition is to classify a set of images to a predefined set of labels which are of interest.

Similarly, the main concept of our proposed methodology is to classify the mobility patterns of vessels to a set of predefined (possibly illegal) activities by visually representing them as images for the purposes of promoting the MSA at sea. The novelty of our approach lies in the usage of computer vision techniques for the classification of trajectories and the detection of activities in mobility data. The use of computer vision for the problem at hand and the main contributions that it offers are: a) The distinct visual difference most mobility patterns in the maritime domain have, allows for the increase in the classification performance of trajectories, b) The classification of trajectories even when the positional data are not given at fixed intervals (e.g. hourly) such as the AIS messages[40], a problem faced by time-series classification methodologies [126], c) A computer vision approach for trajectory classification skips entirely the pre-processing step regarding the understanding and analysis of data and feature extraction. The reason for this is that the same technique for classifying an image (e.g. Convolutional Neural Networks) can be applied for the classification of all of the mobility patterns since all mobility patterns are converted to images. Therefore, a computer vision approach for trajectory classification yields a promising universal approach for the classification of mobility patterns and d) Clustering constitutes often a preliminary step when dealing with trajectory classification techniques. Almost all of the well-known clustering algorithms require input parameters which are hard to determine [138, 48, 13]. As our method skips entirely this step, as stated above, we have eliminated the need of arbitrary user-defined parameters, making our approach scalable and robust.

[40] https://help.marinetraffic.com/hc/en-us/articles/217631867-How-often-do-the-positions-of-the-vessels-get-updated-on-MarineTraffic-



(a) Anchored vessel. (b) Moored vessel. (c) Vessel underway.



(d) Vessel trawling. (e) Vessel longlining.

Figure 36: The movement patterns of 5 distinct vessel activities during an 8-hour window.

8.2 Methodology

8.2.1 Maritime Patterns

In total, five different vessels' mobility patterns have been studied in our work:

Anchored: During this type of activity, vessels are anchored offshore in an anchorage area. When anchored, the vessel tends to move around the anchor and forms circular or semi-circular patterns (Figure 36(a)) due to the effects of the wind, the tide or sea currents.

Moored: During this type of activity, vessels are anchored inside a port. In general, mooring refers to lassoing, tethering or tying to any permanent structure. During this type of activity, the vessel is stopped and the vessel is constrained by the mooring buoys. Its motion is more limited compared to an anchored vessel (Figure 36(b)).

Underway: A vessel is considered underway when it is not aground, anchored or has not been made fast to a dock, the shore, or some other stationary object (Figure 36(c)).

Trawling: There are different kinds of fishing activities such as trawling and longlining. Trawling vessels typically keep their speed steady in order to stabilize the fishing net which is dragged by the boat. Moreover, trawling vessels do not travel at a straight line, but they tend to frequently change their course around the fishing area of interest (Figure 36(d)). The trawling activity can last from several hours to several days.

Longlining: Vessels engaged in longlining activity set fishing lines with baited hooks attached to them. While setting the lines the vessels travel at their steaming speed and they maintain a constant speed. When all lines are set, they are left in the water and the vessels drift slowly with them. Although the two fishing activities have some similarities such as frequent turns and similar speeds, their mobility pattern can differ visually (Figure 36(e)).

8.2.2 Image Representation

This section describes the approach that creates an image representation of the trajectories. To visualize and efficiently classify the movement patterns of the vessels, we need to capture two key features that characterize the trajectory patterns in the maritime domain: i) the shape of the

trajectory and ii) the speed.

Shape of the trajectory. Although trajectories might form similar patterns, the distance each vessel travels through space is different. Therefore, the bounding box or the surveillance area in which the vessel moves needs to be normalized. To efficiently capture and place the shape of the trajectory inside a normalized bounding box we first need to define the total distance of both the x and the y axis in which the vessel moves. To do so, we measure the total horizontal distance (Eq. (24)) and the total vertical distance (Eq. (25)) the vessel travels based on the minimum and maximum longitudes and latitudes respectively. The total horizontal and vertical distance is defined as:

$$d_x = lon_{max} - lon_{min} \quad (24) \qquad d_y = lat_{max} - lat_{min} \quad (25)$$

Then, the distance each AIS position m has travelled from the minimum longitude and latitude can be calculated from the equations 26 and 27 respectively as follows:

$$d(m_x) = lon_m - lon_{min} \quad (26) \qquad d(m_y) = lat_m - lat_{min} \quad (27)$$

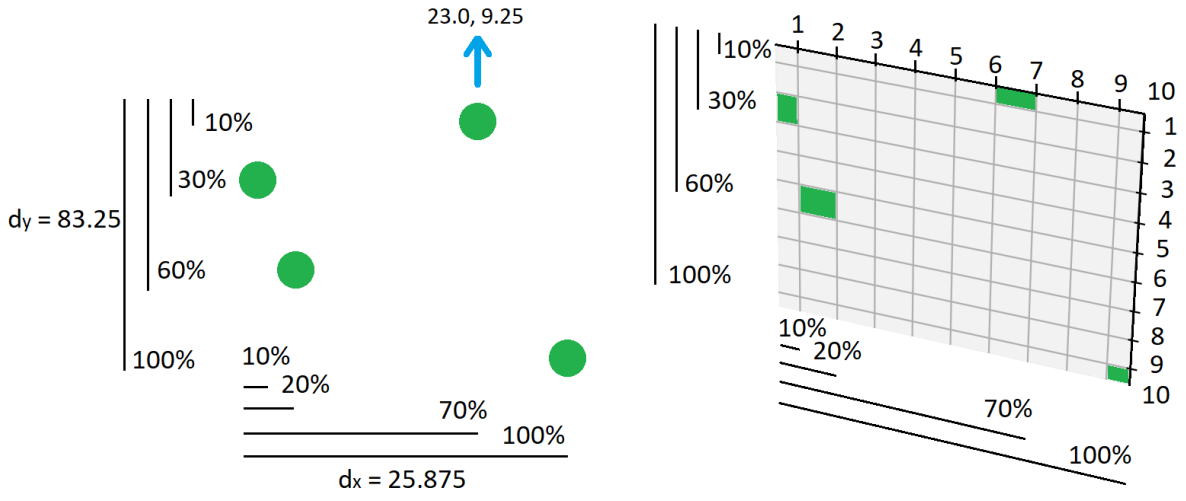
From equations 24,26 and 25,27, we can calculate the percentage of the total distance each AIS position m has travelled so far from the minimum coordinate in both x and y axes:

$$norm(m_x) = d(m_x) \div d_x \quad (28) \qquad norm(m_y) = d(m_y) \div d_y \quad (29)$$

Given a predefined image size of $N \times N$, the exact position of m inside an image can be calculated as follows:

$$p_x = norm(m_x) \times N \quad (30) \qquad p_y = norm(m_y) \times N \quad (31)$$

Therefore, each AIS position is placed inside a normalized bounding box or a surveillance space of size $N \times N$ that is essentially an image representation. An example of the surveillance space normalization can be seen in Figure 37 where each green circle corresponds to an AIS position and $N = 10$. For each AIS position the corresponding longitude and latitude is denoted by the blue arrows.



(a) Positions of the trajectory.

(b) Positions placed in a 10×10 raster.

Figure 37: Example of space normalization.

Finally, in order to make the pattern created by each trajectory more distinctive, a straight line between each temporally consecutive $pixel(pixel_x, pixel_y)$ or AIS position m is drawn using the Bresenham's line algorithm [139]. The Bresenham's line algorithm is a line-drawing algorithm

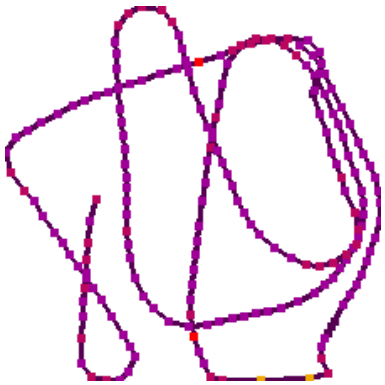


Figure 38: Example of a trawling trajectory that has been transformed into an image.

that generates points that form a close approximation to a straight line between two points of an N-dimensional raster.

Speed. Most common vessel types such as passenger, cargo or fishing vessels report a speed value between the range of 0 to 22 knots, $R = [0, 22]$. To represent the speed values of each AIS position m, the range R was segmented to 2-knot increments with each increment corresponding to a different RGB color value in the final image. Therefore, an AIS position with a speed value of $0 \leq s < 2$ has a different color value compared to an AIS position with a speed value of $2 \leq s < 4$. The 2-knot increment was chosen because we wanted a reasonable amount of color values (11 distinct color values against 22 color values with 1-knot increments) while maintaining a relatively high number of increments. Furthermore, fishing vessels typically report a speed between 2 to 4 knots while fishing [67, 14] which corresponds to a 2-knot increment. Since the speed can sometimes exceed 22 knots, speed values greater than 22 have the same color with the last increment ($20 \leq s < 22$). Moreover, pixels that do not contain an AIS position or a line drawn by the Bresenham's line algorithm are colored white and pixels that contain the lines between the AIS positions use the color of the first increment. Figure 38 illustrates a trawling trajectory that has been converted to a 224×224 image.

8.2.3 Image Features

The next step of the approach is the selection of a proper methodology able to classify the trajectory pattern images. There are many methodologies in the literature that use Convolutional Neural Networks (CNNs) for the problem of image recognition that require a huge amount of data and a considerable amount of time for training. Therefore, to perform image classification, a methodology is required through which features can be extracted from the images that can then be fed to a common classifier such as Random Forests (RFs) [140, 141, 142]. For such a methodology, features that can capture the two key characteristics of a trajectory pattern – the shape and the speed – are required. To this end, we employed 2 types of features that are widely used in the literature, namely Hu invariant moments and Color histogram which describe the image in terms of shape and color respectively:

Hu invariant moments. They were first established by Ming-Kuei Hu and were used for visual pattern recognition [143]. The main idea behind these features is that any geometrical pattern or shape can be represented by a density distribution function with respect to a pair of axes fixed in the visual field [143]. Therefore, the Hu invariant moments are seven features suitable for identifying shapes such as trajectory patterns or handwritings and letters. The Hu invariant moments have been successfully used since then for tasks such as plant classification [144], biometrics-based identification [145] and object recognition [146].

Color histogram. The color histogram is a representation of the distribution of colors in an

Classifier	Temporal Window	Cross Validation			Hold-out		
	h	**Precision**	**Recall**	**F1-score**	**Precision**	**Recall**	**F1-score**
Random Forests	1	0.9843	0.9842	**0.9837**	0.9607	0.96	0.9601
	2	0.9483	0.9543	0.9494	0.9669	0.9666	0.9667
	4	0.9661	0.9674	0.9649	0.9743	0.9766	**0.9767**
	8	0.9755	0.9705	0.9723	0.9735	0.9733	0.9731
Classifier	**Temporal Window**	**Cross Validation**			**Hold-out**		
	h	**Precision**	**Recall**	**F1-score**	**Precision**	**Recall**	**F1-score**
Support Vector Machines	1	0.9344	0.9396	**0.9311**	0.9487	0.9466	**0.9468**
	2	0.9063	0.9012	0.8988	0.9171	0.9133	0.9109
	4	0.8651	0.8567	0.8528	0.9366	0.9333	0.9337
	8	0.8341	0.8088	0.8015	0.8623	0.86	0.8583
Classifier	**Temporal Window**	**Cross Validation**			**Hold-out**		
	h	**Precision**	**Recall**	**F1-score**	**Precision**	**Recall**	**F1-score**
Logistic Regression	1	0.9504	0.9487	**0.9486**	0.9601	0.96	0.9599
	2	0.9306	0.9336	0.9296	0.9343	0.9333	0.9325
	4	0.9118	0.9176	0.9109	0.9665	0.9666	**0.9665**
	8	0.8942	0.8984	0.8927	0.8533	0.8533	0.8533

Table 13: Image classification results.

image. Specifically, the histogram represents the number of pixels that have a specific color out of a predefined list of colors. The list or the range of the colors is given by the number of bins that is used for the histogram. In our case, the number of the bins is equal to the number of the color values as described in Section 8.2.2. Color histograms are a common practice and have been used for content-based image retrieval [147, 148] and vehicle color recognition [149].

8.3 Experimental Results

8.3.1 Dataset Description

The first dataset used contains AIS messages collected from a Terrestrial AIS receiver (T-AIS) that covers the Saronic Gulf (Greece) and contains high quality AIS information without gaps of information. The AIS messages used for our ground truth dataset contain activities that have been extracted from vessels engaged in the following activities: underway, anchored and moored – hereafter **patterns A**. The vessels have been monitored for almost one and a half month period starting at February 18$^{\text{th}}$, 2020 and ending at March 31$^{\text{th}}$, 2020. The dataset provides information for 1229 unique vessels and contains $11,769,237$ AIS records in total. A small sample of the dataset can be found here [150].

The second dataset that was used was provided by MarineTraffic and contains AIS messages from January 1^{st}, 2018 to February 28^{th}, 2018 in the seas of Northern Europe. The AIS messages used for our ground truth dataset contain the following activities: trawling, longlining, moored and underway – hereafter **patterns B**. The total number of AIS messages of this dataset sums up to $61,050$.

In order to provide good-quality representations of vessel activities to the classifiers, 200 representative images from each class from both datasets were selected. The resolution of the images in both datasets is set to 224×224. This resolution is also used by CNNs trained with the ImageNet[41] dataset and allows for a straightforward comparison between our methodology and CNNs in the future.

8.3.2 Experimental Evaluation

To evaluate the image features, we selected three well-known classifiers, namely Random Forests (RFs), Support Vector Machines (SVMs) and Logistic Regression (LR). For each classifier we performed a 10-fold cross validation on the first dataset (**patterns A**), keeping at each fold 90% of the data for training and 10% of the data for validation and reported the macro-average results. Furthermore, at the end of each cross validation, we selected the model with the best classification performance out of the 10 generated models (one for each fold), and we tested it against previously unseen data of vessel activities. We repeated each experiment four times for four different temporal segments or windows of the trajectories: $h = 1, 2, 4, 8$, where h is the number of hours. Specifically, the trajectories of each vessel activity were segmented into equally-sized temporal-window trajectories of length h. Table 13 presents the results of the 10-fold cross validation for each experiment and the classification performance achieved in previously unseen data (Hold-out).

It can be observed from the results that all of the classifiers achieve a f1-score of over 90% most of the times with the Random Forests yielding the best classification performance on the unseen data for each temporal window with a maximum f1-score of 97.67%. The rest of the classifiers do not fall further behind, with the SVMs achieving a maximum f1-score of 94.68% and the LR achieving a maximum f1-score of 96.65%. Finally, on the one hand, we can observe that the Random Forests classifier yields a rather ascending f1-score with respect to the ascending temporal window h. This can be explained by the fact that the mobility patterns of the vessels require some time to form [14]. On the other hand, we can also observe that the rest of the classifiers do not follow the same "ascending f1-score pattern", but they tend to fluctuate and be more unstable. Our results fully agree with other studies of image classification in the literature [140, 141, 142] which consider Random Forests to be the most suitable classifier for the task.

Furthermore, we compared our results with other methodologies for trajectory classification that are illustrated in this recent survey of Wang et al. [151]. State-of-the-art methodologies for trajectory classification include the use of well-known classifiers such as RFs on features extracted from the data points of the trajectories. To this end, we extracted three features from the trajectories during each type of activity (Anchored, Moored and Underway): the average speed of the vessel, standard deviation of its speed and the haversine distance between the first and the last vessel position. These features were selected because they demonstrate distinct differences between these specific activities. A moored vessel typically has a zero speed while an anchored vessel moves with slightly greater speeds on average due to the effects of the wind. A vessel underway has typically much higher speeds and the distance between the first and the last position is greater compared to the distance of anchored or moored vessels. These features were then fed to two classifiers, Random Forests and Support Vector Machines. The methodologies used in the comparison are shown below:

- M1: the proposed methodology of this chapter (image classification with Random Forests).
- M2: the Random Forests classifier with the features extracted from the AIS messages of the trajectories.
- M3: the Support Vector Machines classifier with the features extracted from the AIS messages of the trajectories.

After the feature extraction, we performed 10-fold cross validation for each methodology similarly to the previous experiment. The temporal length (h) of the trajectories used in this comparison is equal to 8 hours. Table 14 illustrates the cross-validation macro average results of the comparison between the different approaches. Results show that our proposed methodology (M1) surpasses the

[41] http://www.image-net.org/

Methodology	Cross Validation		
	Precision	**Recall**	**F1-score**
M1	0.9755	0.9705	0.9723
M2	0.966	0.9603	0.9618
M3	0.924	0.9138	0.911

Table 14: Comparison of methodologies on patterns A.

Experiment	Precision	Recall	F1-score
Cross Validation	0.9686	0.9652	0.9653
Hold-out	0.9887	0.9885	0.9885

Table 15: Image classification results for fishing vessels.

other methodologies (M2,M3) in terms of classification performance. M1 achieved a f1-score of 97.23% and (M2,M3) achieved a f1-score of (96.18%, 91.1%) respectively.

To further evaluate our methodology on vessel activities of utmost importance to the maritime authorities, we performed 10-fold cross validation on the second dataset (**patterns B**) that contains fishing vessels engaged in trawling, longlining, moored or underway. A fishing vessel is typically moored at a port, then travels towards a fishing area and is finally engaged in the fishing activity. After the fishing ends, the vessel travels back to the port and moors. Similarly to the first set of experiments, at each fold 90% of the data is kept for training and the rest 10% of the data is kept for validation. Finally, the best model is used to test the classifier against unseen data. For this experiment, only the Random Forest classifier was used since the previous set of experiments demonstrated that it has the best classification performance. Furthermore, we only used the temporal window of 8 hours due to the fact that fishing activities can last from several hours to several days [14]. The results of this experiment are shown in Table 15 that demonstrates a high classification performance.

Finally, we compared the image classification methodology to the approach proposed in [14] for the fishing vessels (M4) where they use a set of features suitable for the fishing activities. Moreover, we also used the methodology of the previous experiment (M2) which uses features that are suitable for patterns A. Similarly to the previous set of experiments, we performed 10-fold cross-validation for each classifier in trajectories of 8-hour length. The macro average results are reported in Table 16. The results again validate that the proposed approach (M1) is more accurate than the set of features selected specifically for the classification of fishing activities (M4). Furthermore, results demonstrate that the methodology M2 does not perform well (f1-score of 76.2%). This is explained by the fact that the features used for patterns A are not able to discriminate between patterns B. On the other hand, the proposed methodology (M1) is able to perform well in both sets of patterns due to the fact that the patterns have distinct visual differences, thus eliminating the need for different feature selection for different mobility patterns.

Methodology	Cross Validation		
	Precision	**Recall**	**F1-score**
M1	0.9686	0.9652	0.9653
M2	0.7523	0.772	0.762
M4	0.8455	0.8436	0.8446

Table 16: Comparison of methodologies on patterns B.

8.4 Summary

In this work we presented a novel approach for trajectory classification which employs a computer vision technique. The aim of our approach is to provide a high-precision classification of different vessel mobility patterns and to create a universal approach for the classification of vessel activities. The classification performance of the proposed methodology was evaluated on two real-world datasets and demonstrated an f1-score of over 95%.

9 A deep learning streaming methodology for trajectory classification

9.1 Introduction

The sudden increase in mobility data volume and velocity has gained researchers' attention in event-based, distributed, and streaming knowledge extraction methodologies. The automatic identification of patterns in voluminous data is of uttermost importance. Nowadays, vessels over 300 gross tonnage worldwide are required to carry an Automatic Identification System (AIS) transponder on board, a vessel tracking system that allows vessels to report their position periodically. Although the AIS was initially developed to ensure safety at sea by aiding vessels in collision avoidance situations and assist officers on board, it did not take long for maritime authorities to realize that vessels' mobility data can provide useful information. Mobility patterns formed by vessels through the AIS can reveal behaviors able to explain suspicious or illegal activities at sea, making the early identification of such events a prominent way for Maritime Situational Awareness (MSA).

Several distinct use cases that demonstrate the need of an increased MSA include the detection of anomalous vessel behaviour due to damage of the ship, Search And Rescue operations (SAR) and collision detections, piracy and illegal fishing activities. Regarding piracy, several armed piracy attacks happen every year[42,43,44] that endanger the lives of passengers and crew members alike. For that reason, several services are dedicated to immediately report such illegal behaviors[45,46]. Regarding the fishing activities, the immediate identification of Illegal, Unreported and Unregulated (IUU) fishing activities can provide authorities a means of safer maritime surveillance. It is estimated that approximately 640,000 tonnes of ghost gear is left in the world's oceans each year, that ends up entangling and killing birds, and other sea animals[47]. This led to the creation of services such as Global Fishing Watch[48] which promote ocean sustainability and transparency. To promote the MSA, several supervised or unsupervised learning techniques were developed that exploit kinematic and spatiotemporal characteristics of vessels' trajectories.

Approaches that take advantage of such characteristics include trajectory clustering, classification, anomaly detection and event prediction. Specifically, trajectory classification is a widely used technique with which normality and behavioral models are created able to identify anomalous patterns or events of interest. On the other hand, trajectory clustering approaches are often employed to form groups of AIS positions with similar spatiotemporal behavior, uncovering behaviors that are harder to predefine. Although there is an abadance of studies in the literature regarding offline trajectory classification and clustering [6, 126, 64, 152, 151], fewer works have focused on steam processing of events in the maritime domain [153, 129, 9, 11, 14]. Event processing methodologies are faced with significant challenges when employed in streaming data where requirements for such applications demand low memory consumption and decreased latencies.

In all those studies, the context of the analysis is typically the physical world and the geography. Latitude and longitude are the basic features in a multi-dimensional space (speed, direction, etc). However, experts rely heavily in the visualization of trajectories to manually identify parts of the trajectory that are of some importance. This provides an intuition to move the analysis in a different domain, leveraging computer vision techniques on classification. In computer vision, the most commonly used techniques include convolutional neural networks (CNNs) [134, 154, 135]. Each layer of a CNN identifies a different feature of the image, including but not limited to shape and

[42] https://www.statista.com/topics/1290/pirate-attacks/
[43] https://www.bbc.com/news/business-53426890
[44] https://www.maritime-executive.com/piracy-news
[45] https://www.icc-ccs.org/
[46] https://safety4sea.com/tag/piracy-attack/
[47] https://www.worldanimalprotection.org/illegal-fishing-threatens-wildlife
[48] https://globalfishingwatch.org/

color. In order to increase the performance of CNNs, researchers employed deep learning [154, 135] which increases the complexity of the networks in terms of number of hidden layers and nodes. One of the most common goals of such networks is to classify a set of images to a predefined set of labels which are of interest.

Similarly, the main concept of our proposed methodology is to classify in near real-time the mobility patterns of vessels to a set of predefined (possibly illegal) activities for the purposes of promoting the MSA at sea. The novelty of our approach lies in the usage of computer vision techniques for the classification of trajectories and the detection of activities in mobility data. Specifically, the proposed approach leverages image classification techniques by visually representing vessel trajectories as images. The use of image classification for the problem at hand and its main contributions are:

- Mobility patterns in the maritime domain are visually distinct. This visual distinctiveness allows the increase of trajectory classification performance,
- image classification allows for the trajectory classification even when data are not transmitted at fixed intervals (e.g. hourly) such as the AIS protocol [49]. This is in contrast to time-series methodologies [126] that are inherently unsuitable for such tasks and require data points at fixed time points,
- trajectory classification approaches found in the literature [67, 68, 14, 6, 152] require a preprocessing step such as the understanding and analysis of data and the selection of features suitable only for the mobility patterns to be classified. This means that features selected for a certain mobility pattern cannot be applied to other patterns as well [67]. An image classification approach for trajectory classification skips entirely the aforementioned preprocessing step; the same technique for classifying an image (e.g. CNNs) can be applied for the classification of all of the mobility patterns since they are converted into images. Therefore, an image classification approach for trajectory classification yields a promising universal approach for the classification of mobility patterns,
- approximately $16,000$ AIS messages are generated each second from $200,000$ vessels worldwide, resulting in 46GB of data per day. In the maritime domain, only the recent years researchers have started tackling the problem of real-time stream processing with the use of AIS messages [153, 129, 9, 11, 14]. To the best of our knowledge, this is the first time in the maritime domain literature that computer vision techniques are used in real-time to classify trajectories. Deep learning algorithms such as VGG16 [155], InceptionV3 [156], NASNet-Large [157] and DenseNet201 [158] are used in a distributed and streaming fashion enabling low response times and early event detection of suspicious activities at sea,
- the clustering of trajectories often constitutes an initial step when dealing with trajectory classification. Most of the well-established clustering algorithms require input parameters which are hard to determine (Optics [138], Traclus [48], DBSCAN [13]) that eventually have a significant impact on the clustering results. As our method skips entirely this step, as stated above, we have eliminated the need of arbitrary or empirical user-defined parameters, making our approach scalable and robust,
- due to these unparalleled quantities of trajectory data which in turn can overwhelm human analysis approaches, several compression techniques are applied in order to minimize the size of trajectory data, while at the same time, minimizing the impact on the trajectory analysis

[49] https://help.marinetraffic.com/hc/en-us/articles/217631867-How-often-do-the-positions-of-the-vessels-get-updated-on-MarineTraffic-

methods. Thus, we performed some preliminary experiments in order to demonstrate the effect of trajectory compression over classification accuracy of mobility patterns.

The rest of the chapter is organized as follows. Section 9.2 describes in detail the proposed methodology of deep learning stream processing of mobility patterns while Section 9.3 evaluates the proposed approach in terms of classification and execution performance. Moreover, Section 9.4 discusses the experimental results and the benefits of the proposed approach for the problem at hand. Finally, Section 9.5 summarizes the merits of our work and highlights some perspectives that require further attention in the future.

9.2 Methodology

In this section, we present our proposed approach for the classification of vessel activities in near real-time. Specifically, we first demonstrate in Subsection 9.2.1 the mobility patterns or vessel activities which are used for the experimental evaluation. Despite the fact that four mobility patterns are used for demonstration purposes, our approach can be extended to use more patterns and classify more classes such as piracy if a ground truth is provided. Then, in Subsection 9.2.2 the transformation of the trajectories or mobility patterns into images is presented while in Subsection 9.2.3 the way these images are exploited for their classification through deep learning techniques are explained in detail. Furthermore, the classification of images in real-time is presented in Subsection 9.2.4. Finally, in Subsection 9.2.5 several trajectory compression algorithms are presented which will serve as a means of reducing the size of the AIS data while at the same time having a minimum effect on the classification accuracy.

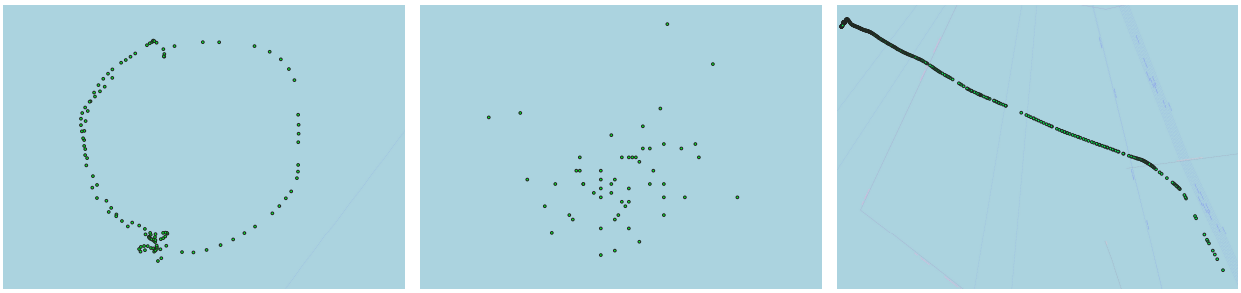
9.2.1 Maritime Patterns

In this research study, five different vessels' mobility patterns have been studied:

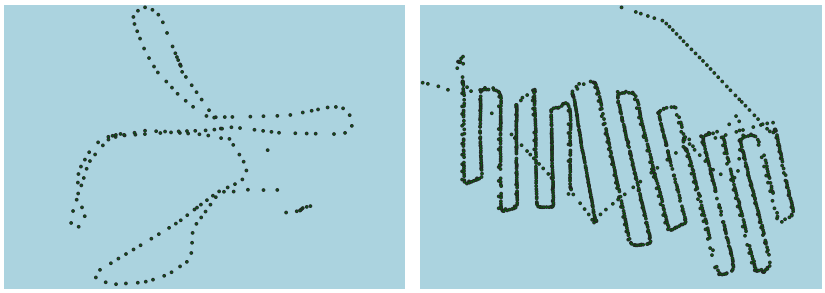
Anchored. Anchoring is a critical operation as it constitutes the key to avoiding accidents related to damage/loss of the vessel or other nearby vessels, by preventing drifting away from a desired position. Anchoring can be related to loading/unloading cargo, maintenance, waiting a berth and refueling as referred in [159]. When anchoring, factors such as sea and wind conditions (direction and strength), sea currents, prohibited areas, underwater pipelines, shallow waters and other vessels in the nearby area must be taken into account. During this type of activity, vessels are anchored offshore in an anchorage area. The vessel tends to move around the anchor and forms circular or semi-circular patterns in different orientations as shown in Figure 39(a) in which the anchor is located approximately in the middle of the circle. The circular movement of the vessel around the anchor can be caused due to the effects of the wind, the tide or sea currents and as a result the speed over ground is minimized to approximately 0.5-3.0 knots depending on the vessel type.

Moored. During this type of activity, vessels are anchored inside a port. Mooring can be related to lassoing, tethering, tying or any permanent structure to which a vessel may be secured such as quays, wharfs, jetties, piers, anchor buoys and mooring buoys. Vessel's motion is more limited compared to an anchored vessel because the latter is constrained not only by the anchor but by the mooring buoys as well. As shown in Figure 39(b), vessel position appear scattered and "close" to the mooring. The slight movement is due to the effects of the wind and the current of the sea and as a result the speed over ground does not exceed 1 or 2 knots.

Underway. During this type of activity, a vessel is travelling from a departure point to a destination point. A vessel is considered underway when it is not aground, anchored or has not been made fast to a dock, the shore, or some other stationary object. An underway vessel is not necessarily propelled or pushed by an instrument or a device, but it may be underway because of the wind or the sea current [58, 56]. Figure 39(c) shows the trajectory of an underway vessel. The



(a) Anchored vessel. (b) Moored vessel. (c) Vessel underway.



(d) Vessel trawling. (e) Vessel longlining.

Figure 39: The different vessels' mobility patterns.

movement patterns of an underway vessel typically form straight, curved or zigzag lines which can occur when a vessel avoids islands in the middle of the sea.

Trawling. There are different kinds of fishing activities such as trawling and longlining. Trawling vessels typically keep their speed steady at approximately 2.5 knots in order to stabilize the fishing net which is dragged by the boat. Moreover, trawling vessels do not travel at a straight line, but they tend to frequently change their course around the fishing area of interest (Figure 39(d)). The trawling activity can last from several hours to several days.

Longlining. On the other hand, vessels engaged in longlining activity set fishing lines with baited hooks attached to them. While setting the lines the vessels travel at their steaming speed and they maintain a constant speed. When all lines are set, they are left in the water and the vessels drift slowly with them. While drifting, the speed of the vessel can be at approximately 2.5 knots due to the wind or the current of the water. Finally, the longlining activity has a similar duration to the trawling activity and can last several days. Although the two fishing activities have some similarities such as frequent turns and similar speeds, their mobility pattern can differ visually (Figure 39(e)). It can be observed that long straight-line sub-trajectories are formed which correspond to the part of the trajectory during which longlining vessels set the lines with the baited hooks.

9.2.2 Image Representation

This section describes the image representation approach of the trajectories. These trajectories are the movement patterns indicative of the vessel activities as defined in the previous sub-section. In order to visualize and efficiently classify the movement patterns of the vessels, two key features are captured that characterize the trajectory patterns in the maritime domain: i) the shape of the trajectory which indicates the way the vessel moves in space taking into account the rate of change of Course Over Ground (COG) or heading and ii) the speed which indicates how fast the vessel moves in space.

Shape of the trajectory. Trajectories of the same vessel activity form similar patterns. However, as the distance each vessel travels through space is different (e.g. a fishing vessel in the Atlantic ocean travels greater distances compared to a fishing vessel in the Irish sea), the bounding

box or the surveillance area in which the vessel moves needs to be normalized. Therefore, to efficiently capture and place the shape of the trajectory inside a normalized bounding box, the total distance of both the x and the y axis in which the vessel moves must be defined first. For this reason, we calculate the total horizontal distance (x – Eq. (32)) and the total vertical distance (y – Eq. (33)) the vessel travels based on the minimum and maximum longitudes and latitudes respectively. The total horizontal distance is defined as:

$$distance_x = longitude_{max} - longitude_{min} \tag{32}$$

Similarly, the total vertical distance the vessel has travelled is defined as:

$$distance_y = latitude_{max} - latitude_{min} \tag{33}$$

Then, the distance each AIS position m has travelled from the minimum longitude and latitude can be calculated from the equations 34 and 35 respectively as follows:

$$distance(m_x) = longitude_m - longitude_{min} \tag{34}$$

and

$$distance(m_y) = latitude_m - latitude_{min} \tag{35}$$

From equations 32,34 and 33,35, we can calculate the percentage of the total distance each AIS position m has travelled so far from the minimum coordinate in both x and y axes:

$$normalized(m_x) = distance(m_x) \div distance_x \tag{36}$$

and

$$normalized(m_y) = distance(m_y) \div distance_y \tag{37}$$

Given a predefined image size of $N \times N$, the exact position of m inside an image can be calculated as follows:

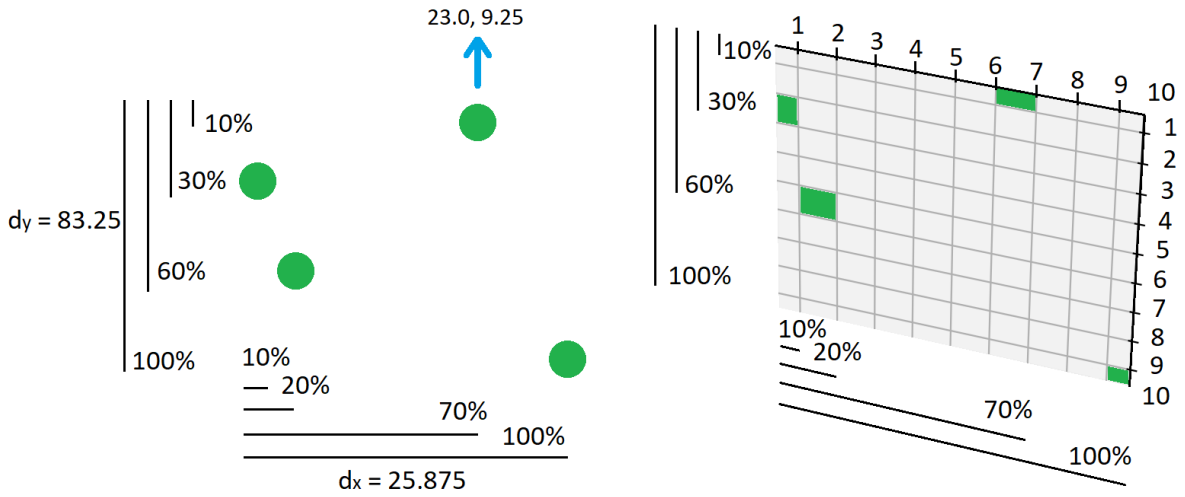
$$pixel_x = normalized(m_x) \times N \tag{38}$$

and

$$pixel_y = normalized(m_y) \times N \tag{39}$$

Therefore, each AIS position is placed inside a normalized bounding box or a surveillance space of size $N \times N$ that is essentially an image representation. Figure 40 demonstrates an example of the surveillance space normalization for $N = 10$. Each green circle corresponds to an AIS position where the corresponding longitude and latitude is denoted by the blue arrows. The total horizontal distance is $distance_x = 28.75 - 2.875 = 25.875$ while the total vertical distance is $distance_y = 92.5 - 9.25 = 83.25$. Based on equation (34) and (35), $distance(m_x)$ of the upper right AIS message is equal to 20.125 and $distance(m_y)$ is equal to 0. According to equations (36) and (37), $normalized(m_x)$ is equal to 0.7 and $normalized(m_y)$ is equal to 0. Multiplying each normalized value by $N = 10$ results in $pixel_x = 7$ and $pixel_y = 0$. In order to fit boundary AIS positions into the normalized surveillance space, pixel values smaller than N are transformed into 1 and pixel values larger than N are transformed into N Therefore, $pixel_y$ is transformed into 1 and the final pixel position inside the 10×10 image is $x = 7$ and $y = 1$ as shown in Figure 40.

Finally, a straight line between each temporally consecutive $pixel(pixel_x, pixel_y)$ or AIS position m is drawn using the Bresenham's line algorithm, [139] in order to make the pattern created by each trajectory more distinctive. The Bresenham's line algorithm is a line-drawing algorithm



(a) Positions of the trajectory.

(b) Positions placed in a 10×10 raster.

Figure 40: Example of space normalization.

that generates points that form a close approximation to a straight line between two points of an N-dimensional raster.

Speed. The range of speed regarding the most common vessel types such as passenger, cargo or fishing varies between 0 to 22 knots, $R = [0, 22]$. In order to represent the speed values of each AIS position m, the range R was segmented to 2-knot increments with each increment corresponding to a different RGB color value in the final image. The 2-knot increment was chosen because we wanted a reasonable amount of color values (11 distinct color values against 22 color values with 1-knot increments) while maintaining a relatively high number of increments. Therefore, the color values are different for dissimilar speed values of AIS positions. Furthermore, the speed of the fishing vessels varies between 2 to 4 knots [67, 14] which corresponds to a 2-knot increment. Sometimes speeds exceed R range, thus speed values greater than 22 knots have the same color with the last increment ($20 \leq s < 22$). Moreover, pixels that do not contain an AIS position or a line drawn by the Bresenham's line algorithm are colored white and pixels that contain the lines between the AIS positions use the color of the first increment. As a result, there are 12 color values in a trajectory image. An example of a trawling trajectory is shown in Figure 41. Finally, it is worth noting that the transformation of a trajectory into an image adds a negligible overhead.

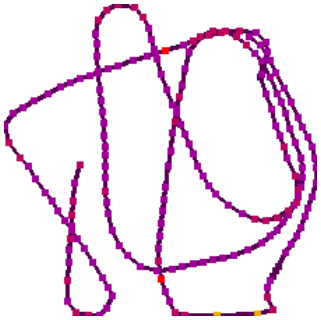


Figure 41: Example of a trawling trajectory that has been transformed into an image.

9.2.3 Deep Learning for Vessel Pattern Classification

This section describes the deep learning approach for vessel mobility pattern classification from trajectory images, based on Convolutional Neural Networks (CNN). CNNs do not require hand-

created feature extraction and deep features of trajectories such as shape and color (e.g. speed) can be learned by the CNN automatically. As stated in [76], neurons in CNN receive signals from other neurons in local area in the preceding layer, thus the CNN is able to capture more local spatial correlation. Furthermore, the weights-sharing characteristic in the connection of adjacent layers can significantly reduce the number of variables. The main disadvantage of deep learning approaches regarding image classification is that they require a large amount of data in order to perform accurate feature extraction and classification. In order to overcome this limitation, transfer learning was adopted. Transfer learning is a common and effective method which aims at training a network with fewer samples, as the knowledge extracted by a pre-trained model is then reused and applied to the given task of interest. The intuition behind transfer learning is that generic features learned on a general large dataset can be shared among seemingly disparate datasets. The learned features can be used to solve a different but related task [160].

In general, there are two ways for transfer learning utilization in the context of deep learning [161]: a) feature extraction [162] and b) fine-tuning [163, 164]. In feature extraction, representations learned from a pre-trained model are treated as an arbitrary feature extractor and employed in order to extract meaningful features from new samples. As the base convolutional network already contains generically useful features for classification, there is no need for retraining the entire model. In fine-tuning, the fully connected layers of the pre-trained model are replaced with a new set of fully connected layers. These new layers are trained on a given dataset and the weights of the top layers of the pre-trained model along with the newly-added layers are "fine-tuned" by means of backpropagation. Thus, the weights are tuned from generic feature maps to features associated specifically with the provided dataset. With fine-tuning, specialized features are adapted to a given task. Fine-tuned learning experiments are presented to be much faster and more accurate in comparison with models trained from scratch [165].

In this research work, fine-tuning is employed as the standard method for transfer learning. The weights are pre-trained on the ImageNet [166] dataset for all the deep CNNs. Figure 42 illustrates the fine-tuning process on the VGG16 network. The same process is applied in all the examined deep learning models. VGG16 model contains 13 convolutional ($CONV$) and 3 fully-connected (FC) layers. The final set of layers which contain the FC layers along with the $softmax$ activation function is called "head". The network is instantiated with weights pre-trained on ImageNet as shown on top of the figure. Afterwards, the FC layers are truncated and the final $POOL$ layer is treated as a feature extractor as depicted in the middle of the figure. Finally, the truncated layers are replaced by a new FC head layer which is randomly initialized and placed on top of the original architecture (bottom of the figure). Then the model is trained through a backpropagation process. The body of the network i.e. the weights of the $CONV$ layers of the pre-trained network have been freezed such that only the FC head layer is trained. This is because the $CONV$ layers have already learned discriminative filters and capture universal features like curves and edges, thus these weights have to remain intact. On the other hand FC head layer is randomly initialized from scratch and focused on learning dataset-specific features, thus random values are able to destroy the learned features. Early-layer features appear more generic whereas later features progressively become more specific to a specific task [167].

Historical trajectory data are transformed into images and labeled based on concrete mobility patterns annotation into four classes ($anchored$, $moored$, $underway$, $fishing$). These images are fed as input into the different deep learning models for training. The labels are encoded into one-hot vectors and then each model is trained through a backpropagation operation until the optimization process of the objective function converges. All the examined CNNs share some common hyper-parameters. Input images were scaled to the fixed size of 224×224 pixels. Training was conducted for 25 epochs for all pre-trained models with a learning rate of $1e-3$ and a batch size of 8. The output of the convolution layers are activated by the non-linear activation function called Rectified Linear Unit (ReLU) which computes the function:

Figure 42: Fine-tuning on the VGG16 network architecture.

$$ReLU(x) = \begin{cases} x, & x > 0 \\ 0, & x \leq 0 \end{cases} \tag{40}$$

ReLU was chosen as the activation function due its reduced likelihood of vanishing gradient and its efficient computation. After each convolution layer, the pooling layer is introduced to carry out downsampling operations which reduces the in-plane dimensionality of the feature maps. Downsample operations after convolutional layers introduce a translation invariance to small shifts and distortions and decrease the number of subsequent learnable parameters. Average pooling [168] is employed as the pooling strategy which performs an extreme type of dimensionality reduction, where a tensor with dimensions $h \times w \times d$ is downsampled into a $1 \times 1 \times d$ array by simply taking the average of all the elements in each $h \times w$ feature map, whereas the depth of feature maps is retained. The number of learnable parameters is reduced, preventing overfitting. The output feature maps of the final layer are flattened and connected to the fully connected layers, in which every input is connected to every output by a learnable weight. The output layer generates probability distribution over the classification labels by resorting the softmax function:

$$softmax(x)_i = \frac{exp(x_i)}{\sum_{j=1}^{I} exp(x_j))} \tag{41}$$

where x is the output vector of the network. Softmax function calculates the probabilities of each class $i = 1, 2, ..., I$ over all possible target classes.

The error is calculated between the actual output per class t_i and the estimated output derived from $f(s)_i$ using the "categorical_crossentropy" as the loss function:

$$CE = -\sum_{i}^{C} t_i \log(f(s)_i) \tag{42}$$

where $f(s)$ is referred to Eq. 41. "Categorical_crossentropy" compares the distribution of the

predictions with the true distribution. True class is represented as an one-hot encoded vector, and the closer the model's outputs are to that vector, the lower the loss.

Examined CNNs were compiled utilizing the optimization method called Adam [169] which finds the minimum of the objective (error) function making use of its gradient. Furthermore, a dropout layer [170] of 0.5 is applied which means that 50% of neurons will randomly set to zero during each training epoch thus avoiding overfitting on the training dataset. Dropout is one of the most popular regularization technique which forces the weights in the network to receive only small values making the distribution of weight values more regular. As a result this technique can reduce overfitting on small training examples [171]. Another efficient way to prevent model overfitting is data augmentation which increases the amount of training data [172]. Thus, data augmentation is performed during training leveraging several multi-processing techniques. Specifically, the transformations employed include random rotation of the images (maximum rotation angle was 30 degrees), horizontal flips, shearing, zooming, cropping and small random noise perturbation. Data augmentation improves the generalization and enhance the learning capability of the model. Table 17 summarizes the parameters employed by the different deep CNNs.

Parameter	Value
Epochs	25
Learning rate	$1e-3$
Batch size	8
Activation function	$ReLU$
Pooling	$Average2D$
Optimizer	$Adam$
Output activation function	$Softmax$
Loss function	$categorical\_crossentropy$
Dropout probability	0.5

Table 17: Configuration of CNNs architectures.

9.2.4 Streaming vessel classification

Low latency and high throughput are two key characteristics of streaming systems that support fast decision-making. In such systems events must be processed in real-time e.g. after the consumption of an event the system must output a result as soon as possible. In a real-world scenario, approximately $200,000$ vessels globally transmit more than $16,000$ AIS messages per second totalling in 46GB a day with each AIS receiver being flooded with 5 to 8 AIS messages per second. Therefore, a streaming system that balances latency and throughput needs to be developed for the classification of vessel activities in near real-time. To this end, we propose a deep learning streaming methodology for the mobility patterns identification which consists of two phases, the offline model training and the real-time vessel activity classification.

Offline model training. In this phase, the deep learning model is created as described in Section 9.2.3. For the training of the model, representative trajectories of the mobility patterns are required to be used as ground truth. Thus, already labeled trajectories from historical AIS data which have been annotated as "anchored", "moored", "underway" and "fishing" are used. These trajectories are converted into images which in turn are used as training instances of the deep learning model. For the implementation, the Keras[50] library with a Tensorflow[51] backend was used, which consists of not only APIs to create neural networks but pre-trained CNN models as well.

[50] https://keras.io/
[51] https://www.tensorflow.org/

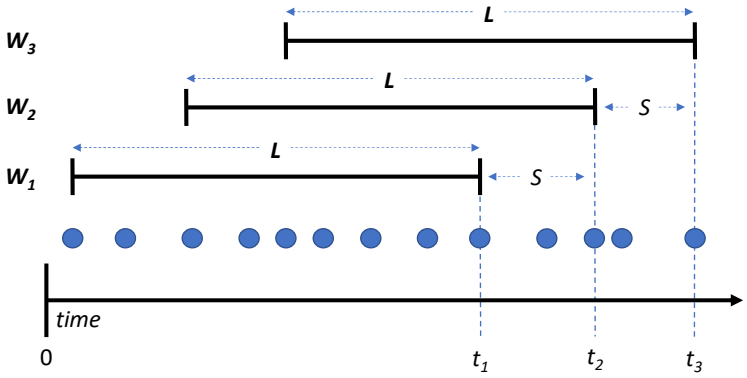


Figure 43: Example of sliding windows with a sliding step of $S = 2$ events.

These pre-trained models are employed and fine-tuned to classify images of mobility patterns in the next phase.

Real-time vessel activity classification. There are several frameworks for distributed stream processing such as Apache Spark[52], Apache Flink[53] and Kafka streams[54], out of which only Apache Spark has support of the python programming language which is needed for the implementation of the neural networks and the creation of the images. Apache Spark is not preferred since it performs micro-batching over streams of events and a system is needed which can handle event-processing in real-time. Therefore, to balance event-processing with low latency and high throughput, the Apache Kafka[55] framework was used in this phase, a distributed publish-subscribe and message-exchange platform similar to a message queue able to process streams of events as they occur. Three major concepts exist in the Apache Kafka ecosystem, namely topics, producers and consumers. A kafka topic is a category/feed name to which messages are stored and published. A producer is an application that continuously publishes or stores messages in a topic. A consumer is an application that is subscribed to a topic and continuously reads or consumes messages. A kafka topic can be divided into n partitions with each partition storing different messages. Specifically, messages with the same key will be stored in the same partition. n consumers can be subscribed to the partitioned topic with each consumer consuming from a different partition thus enabling high throughput. A producer can store messages to the partitioned topic and Apache Kafka will handle the load balancing of the messages among the partitions internally. In our use case, the vessel identifier can be considered as the message key, the AIS receiver as the producer and the prediction modules as the consumers. An even distribution of load within the nodes of the system reduces the probability that a node turns to a hotspot and his property also acts as a safeguard to the system reliability [173, 174].

The prediction modules are the main components of our methodology. Each prediction module is responsible for consuming AIS messages from a set of vessels and classifying parts of their trajectories based on the deep learning model created in the previous phase. To classify parts of the vessels' trajectories, the module uses a temporal sliding window W of user-defined length L and step S. Every S AIS messages the module takes into account all of the AIS messages of the corresponding vessel that belong to the current window W and converts them to an image. Next, the deep learning model reads the image and outputs for each of the predefined vessel activities a probability. The vessel activity with the highest probability is the final prediction of the module. Figure 43 illustrates the sliding windows of the prediction modules. A window of length L and a step of $S = 2$ events is illustrated that slides from left to right (W_1 to W_3). The upper temporal

[52] https://spark.apache.org/streaming/
[53] https://flink.apache.org/
[54] https://kafka.apache.org/documentation/streams/
[55] https://kafka.apache.org/

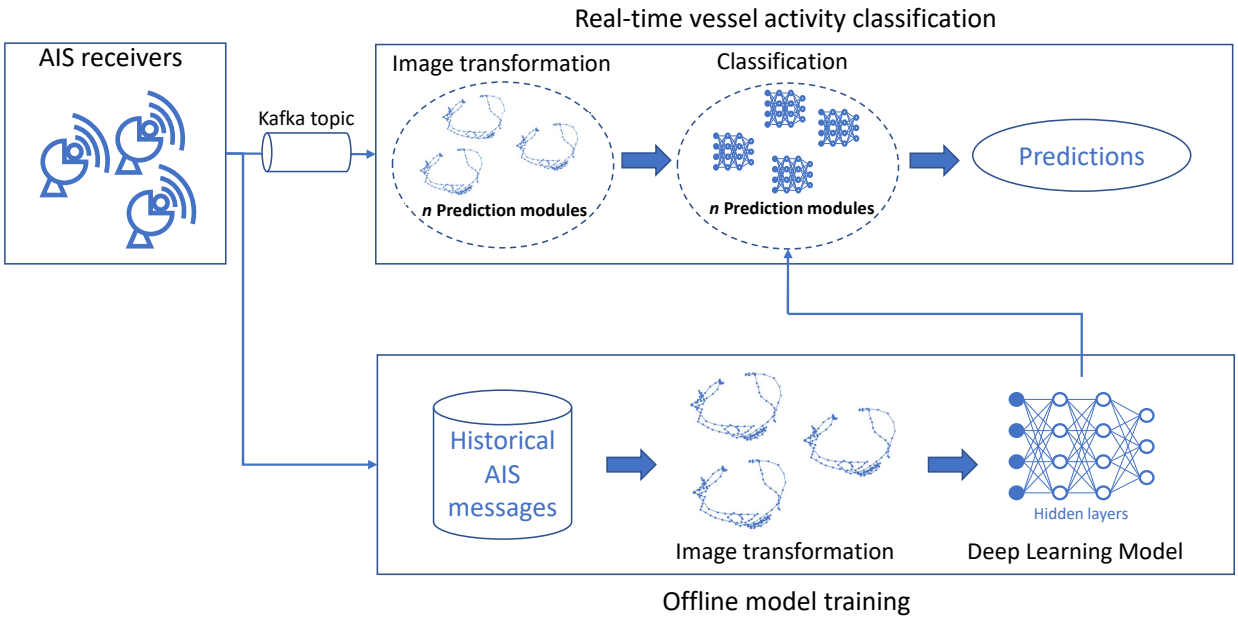


Figure 44: System architecture.

limit of the window is the time t the incoming message was consumed and the lower temporal limit of the window is L hours/minutes/seconds before t, $t - L$. Messages that fall within the window W are used for the prediction.

Finally, Figure 44 visualizes the entire system architecture of our proposed methodology. Initially, historical AIS messages that have been collected from AIS receivers are transformed into images. These images are then used to train a deep learning model offline. For the real-time classification, messages received through the AIS receivers are stored into a kafka topic. These messages are then consumed by the prediction modules and transformed into images which are fed to the already trained deep learning model for the prediction of the vessel activity. During the real-time vessel activity classification more than one prediction module can be employed to increase the throughput of the system.

9.2.5 Trajectory Compression

The increasing amount of spatio-temporal data being constantly generated by modern-day systems such as GPS devices, AIS receiving stations, location-based services and satellites have met staggering growth. Managing and analyzing these data is becoming increasingly important, enabling applications that may transform science and society [175], [176]. Spatio-temporal database management systems constitute the basic components in dealing with the challenges posed by spatio-temporal applications [177]. While the more points collected, the more accurate a trajectory representation becomes, the enormous volumes of trajectory data can quickly overwhelm available data storage systems and the redundant information often contained in these data can overwhelm human analysis.

A typical approach towards above challenges is the reduction of trajectory data by employing compression techniques. Trajectory compression aims at substantial reductions in the amount of data while minimizing the loss of information (error) and at the same time preserving the quality of trajectory. Thus, it is become clear that there is a trade-off between the compression rate and the quality of trajectory achieved after compression. The quality is strongly related to the information loss and measured though metrics that calculate the applied error over a trajectory if a certain point is discarded. The motivation behind trajectory compression in terms of data points representing a trajectory, is that only a small portion of carefully selected points are needed in order to conduct

the analysis task without significant loss in its performance metrics.

Several studies exists in the literature regarding compression which aim to balance the trade-off between the achieved compression rate and the acceptable error. Meratnia et al. [79] was among the first research studies considering three dimensional mobility data, thus trajectories of mobility objects, where the temporal factor was taken into consideration in the compression techniques. Leichsenring et al. [80] present an evaluation of seven lossy compression algorithms with a view to identify the most important aspects in selecting the appropriate compression algorithm while Muckell et al. [81] also present a performance comparison of seven compression algorithms when utilizing two different errors, the synchronized Euclidean distance (SED) and the median difference in speed. Potamias et al. [178] propose two online compression techniques called STTrace and Thresholds. STTrace exploits spatiotemporal features that characterize movement and detects changes in speed and orientation, thus minimizing the SED in each step while in Thresholds the choice of appending a point to the sample, is based on a velocity threshold. In SQUISH [179] the most important points of a trajectory are prioritized and new incoming points require the removal of another point that is already stored in a buffer. The removed point is that with the minimum estimated SED error. Dead Reckoning [180] estimates the successor point through the object's current position and velocity. Finally, an alternative compression technique is presented in [181], which is based on dual transformation of moving objects. Each raw trajectory is approximated by a discrete number of linear sub-trajectories which are subjected to dual transformation. The duality transformation of line segments is based on Hough-X and Hough-Y. In essence, Hough-X constitutes a linear transformation on spatial coordinates. As the results suggest, Hough space represents better mobility patterns than the original trajectory spatial data, thus leading to more homogeneous clusters and provides storage efficiency as well.

Our interest is focused on offline lossy compression algorithms which remove the less significant data in an attempt to preserve the major characteristics of the original trajectory while maintaining an acceptable degree of error. In this research work, five lossy offline top-down compression algorithms are evaluated. In top-down algorithms, a trajectory is recursively splitted until a halting condition is met [182]. The algorithms are as follows:

- Douglas-Peucker (DP)
- Time Ratio (TR)
- Speed Based (SP)
- Heading Based (HD)
- Speed-Heading-Based (SP_HD)

Douglas-Peucker [42] is a line generalization algorithm that recursively selects points from the original set. The process of the algorithm is illustrated in Figure 45. Initially, the first (P_a) and the last point (P_b) are selected as the anchor and float point respectively and these two points form a line segment $LS\ a \rightarrow b$ as shown in Figure 45a. The starting curve is a set of points and a threshold (distance dimension) $\varepsilon > 0$. Subsequently, the point with the maximum perpendicular distance ($\perp$) from $LS\ a \rightarrow b$ is selected (P_1). If this point is closer than ε to the $LS\ a \rightarrow b$, all the points between P_a and P_b are discarded. Otherwise, the newly added point is included in the resulting set and becomes the new float point for the first segment, and the anchor point for the second segment as shown in 45b. This process is repeated using recursion on each line segment (Figure 45c). When the recursion is completed, a new simplified trajectory is generated of only those points that have been marked as kept (Figure 45d).

Despite the popularity of line generalization algorithms in fields of cartography and computer graphics, the main drawback when dealing with trajectory data of moving objects is that they do

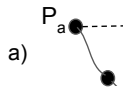


Figure 45: Douglas-Peucker algorithm.

Figure 46: Time Ratio algorithm.

not consider the temporal aspect. The spatial error finds the closest point on the compressed representation of a trajectory to each point on the original trajectory. Each trajectory is treated as a line in two-dimensional space. Although, a trajectory incorporates an important extra dimension, time. Meratnia and By [182] proposed the Time Ratio algorithm which computes the distances between pairs of estimated temporally synchronized positions, one on the original and one on the corresponding approximated trajectory as illustrated in Figure 46.

For each point on the original trajectory such as P_i the temporally synchronized point $P_{i'}$ is located on the approximated trajectory Tr_{Pa-Pb} and the coordinates (x'_i,y'_i) of Pi' can be calculated using linear interpolation as:

$$\begin{aligned} x'_i &= x_a + \frac{t_i - t_a}{t_b - t_a}(x_b - x_a) \\ y'_i &= y_a + \frac{t_i - t_a}{t_b - t_a}(y_b - y_a) \end{aligned} \tag{43}$$

After the temporally synchronized points are determined, the synchronized Euclidean distance is calculated as $SED(P_i, P'_i) = \sqrt{(x_i - x'_i)^2 + (y_i - y'_i)^2}$. The (x_i, y_i) and (x'_i, y'_i) represent the coordinates of a moving object at time t_i and synchronized time t'_i in the uncompressed and compressed traces respectively. Finally, if the distance between $P_{i'}$ and P_i is greater than a user-defined threshold, the particular point is included in the resulting set otherwise discarded. By including the temporal factor, the algorithm is able to provide more accurate results.

Speed Based algorithm exploits the speeds from subsequent segments of a trajectory. If the absolute value of speed difference between two subsequent segments of a trajectory, for example $|v_{i+1} - v_i|$, is greater than a user-defined threshold, the point in the middle is retained otherwise discarded. The algorithm is illustrated in Figure 47.

Heading Based algorithm exploits the angle formed by subsequent segments of a trajectory. Initially, the distances between continuous points are calculated. Then, for each triangle formed by three continuous points like $P_aP_bP_c$, the distance length of the opposite side of the examined angle is calculated e.g. P_aP_c and the law of cosines is used in order to find the angle e.g. A_1 as $c^2 = a^2 + b^2 - 2ab\cos\gamma$,

Figure 47: Speed Based algorithm.

Figure 48: Heading Based algorithm.

where γ denotes the angle contained between the sides of length a (P_aP_b) and b (P_bP_c) and opposite the side of length c (P_aP_c). If the angle of two subsequent segments is greater than a user-defined threshold, the point in the middle is retained otherwise discarded. The algorithm is illustrated in Figure 48.

By combining the Speed Based with the Heading Based algorithm a new algorithm can be derived that exploit the concept of both algorithms simultaneously. The algorithm, namely Speed-Heading-Based, retains or discards a point based on two thresholds. If speed or heading are greater than a speed and heading threshold respectively, the point is retained otherwise discarded.

One of the challenges is to define the thresholds to be employed by the compression algorithms. Setting the proper threshold is an application-dependent process and can significantly affect the compression results in terms of compression ratio and achieved quality. In general, the steps that each algorithm follow are: (i) group the points and create a trajectory of each object based on an identifier. This practically means that the number of trajectories in a dataset is as large as the number of objects; (ii) compress the whole trajectory of each identifier; (iii) write the points remaining after compression to a file grouped by identifiers. As we group the points for each identifier (object) in the dataset, we extract the trajectories (one trajectory for each object). In order to determine the threshold that each algorithm will use, a dynamic process is proposed: for each individual trajectory a different threshold is automatically defined based on an average. Specifically, we calculate the average epsilon (DP), average SED (TR), average speed (SP), average heading (HD) and average speed_heading (SP_HD) for each trajectory before any of the respective algorithms is applied. Then we use this average value (AVG) as a reference point to define a common rule for the threshold calculation. This practically means that in every trajectory of each dataset a different threshold is applied in the corresponding algorithm, which depends on the actual features and peculiarities of this trajectory. Thus, we have eliminated the need of arbitrary user-defined thresholds.

9.3 Experimental Evaluation

This section presents the experimental evaluation of the proposed methodology. To properly evaluate the approach presented in Section 9.2, we present experiments that:

- demonstrate the achieved classification accuracy and the overall classification performance of the deep learning methodology (Subsection 9.3.2),
- demonstrate the achieved latency and throughput and the overall execution performance (Subsection 9.3.3),

- demonstrate the effect the trajectory compression algorithms have on the classification performance of the task at hand (Subsection 9.3.4).

The real-world datasets used to evaluate the proposed trajectory classification approach are presented in the following Subsection(9.3.1).

9.3.1 Dataset Description

The first dataset used contains AIS messages collected from a Terrestrial AIS receiver (T-AIS) that covers the Saronic Gulf (Greece) including the port of Piraeus and contains high quality AIS information without gaps of information. The dataset provides information for 1229 unique vessels and contains 11, 769, 237 AIS records in total. The vessels have been monitored for almost one and a half month period starting at February 18th, 2020 and ending at March 31th, 2020. A small sample of the dataset can be found here [150].

AIS, besides the positional information, also transmits its navigational status[56]. Navigational status is manually inserted by the vessel's crew and constitutes an identifier regarding the vessel's activity (e.g. 5 indicates that the vessel was moored when the message was received). The AIS messages used for our ground truth dataset contain activities that have been extracted from vessels which have set their navigational status to 0,1 and 5 – hereafter patterns A:

- 0: underway
- 1: anchored
- 5: moored

Since the status is reported manually, it can often be observed that the vessel's crew may forget to change the status, resulting in false annotations. To tackle the problem of false annotations and to provide good-quality representations of vessel activities to the classifiers, 150 representative images from each class (450 in total) were selected.

The second dataset that was used was provided by MarineTraffic and contains AIS messages from January 1^{st}, 2018 to February 28^{th}, 2018 in the seas of Northern Europe. The total number of AIS messages of this dataset sums up to $61,050$. Similarly to the first dataset, the AIS messages used for our ground truth dataset contain fishing activities that have been extracted from fishing vessels which have set their navigational status to 7, which indicates "fishing activity", and by vessels which have set their destination to Trawling or Longlining for the corresponding activity. These fishing trajectories are then segmented to:

- fishing (trawling and longlining)
- moored
- underway

Vessels are typically moored to a port, then they travel to the fishing area and finally they are engaged in a fishing activity (trawling or longlining) – hereafter patterns B. Again, 150 representative images from each class (450 in total) were used.

In both datasets, each AIS record is consisted of 10 attributes as described in Table 18.

[56]https://help.marinetraffic.com/hc/en-us/articles/205426887-What-kind-of-information-is-AIS-transmitted-2020-04-06

Feature	Description
ship_id	unique identifier for each ship
lat, lon	the longitude and the latitude of the current ship position
ship_type	AIS reported ship-type
speed	Speed over ground in knots
course	Course over ground in degrees with 0 corresponding to north
heading	Ship's heading in degrees with 0 corresponding to north
destination	AIS reported destination
navigational_status	identifier regarding vessel's activity
timestamp	the time at which the message was received (UTC)

Table 18: Dataset Attributes.

9.3.2 Deep Learning Evaluation

In order to evaluate the classification performance, several commonly used classification metrics were adopted: Accuracy (ACC), Precision, Recall (Sensitivity) and F1-Score. $Accuracy$ indicates how well a classification algorithm can discriminate the classes of the trajectories in the test set. As shown in Eq. 44, ACC can be defined as the proportion of the predicted correct labels (TP + TN) to the total number of labels (N).

$$Accuracy(ACC) = \frac{|Correctly\ Labeled\ Examples|}{N} \tag{44}$$

$Precision$ is the proportion of predicted correct labels to the total number of actual labels as shown in Eq. 45 while $Recall$ is the proportion of predicted correct labels to the total number of predicted labels as shown in Eq. 46. Recall is also known as Sensitivity or true positive rate (TPR).

$$Precision(P) = \frac{TP}{TP + FP} \tag{45}$$

$$Recall(Sensitivity) = \frac{TP}{TP + FN} \tag{46}$$

F1-score consists on the harmonic mean of Precision and Recall, where Precision is the average precision in each class, and Recall is the average recall in each class as illustrated in Eq. 47.

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{47}$$

TP, TN, FP, FN refer to the true positive, true negative, false positive and false negative samples for each class (anchored, moored, underway, fishing).

The classification performance of several pre-trained CNNs namely, VGG16, InceptionV3, NASNetLarge and DenseNet201 were evaluated on the effectiveness of vessel pattern recognition. For each CNN we performed a 5-fold cross validation on the 800 samples (200 samples per class) for both datasets, keeping at each fold 80% of the data for training and 20% of the data for validation and reported the macro-average results. The trajectories were segmented into equally-sized temporal-window trajectories of 8 hours length. This particular temporal-window was selected because the mobility patterns of the vessels require some time to form [14]. During an one-hour window, for instance, the anchored pattern as described in Section 9.2.1 will most probably not have fully formed.

Model	Labels	Precision	Recall	F1-score	Overall Accuracy
	anchored	0.96	1.00	0.98	
VGG16	moored	1.00	1.00	1.00	0.9888
	underway	1.00	0.97	0.99	
	anchored	0.77	1.00	0.87	
InceptionV3	moored	1.00	1.00	1.00	0.9222
	underway	1.00	0.80	0.89	
	anchored	0.76	0.96	0.85	
NASNetLarge	moored	1.00	1.00	1.00	0.9158
	underway	0.97	0.80	0.88	
	anchored	0.68	1.00	0.81	
DenseNet201	moored	1.00	0.97	0.98	0.8777
	underway	1.00	0.71	0.83	

Table 19: Classification performance obtained from different pre-trained CNN models for patterns A.

Table 19 illustrates the classification results of each model of the first dataset (patterns A).

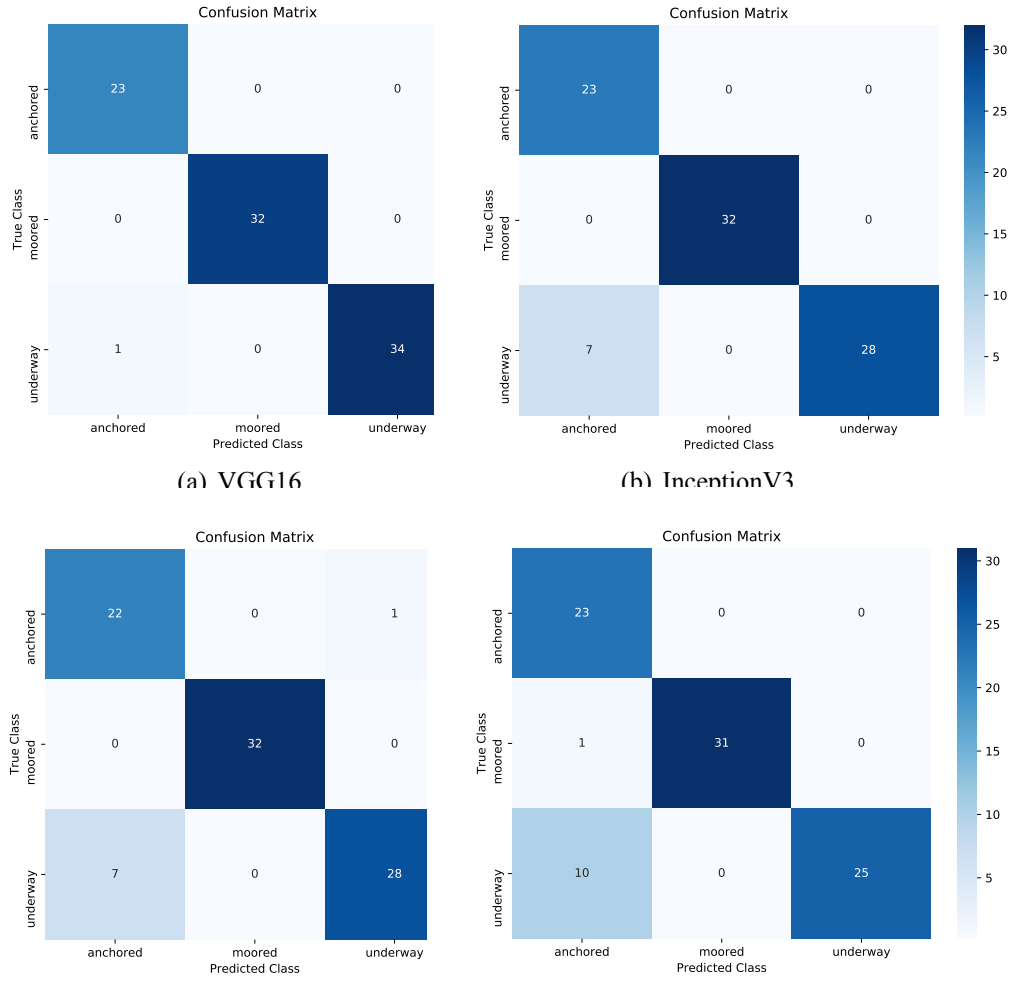
Keras package and a TensorFlow backend are employed along with the python programming language for training the deep learning models. Keras is a simple to use neural network library built on top of Theano or TensorFlow [183].

The results suggest that the VGG16 model achieves the best classification accuracy of almost 99% followed by InceptionV3 and NASNetLarge with almost the same accuracy of 92%. DenseNet201 presents the worst results with an accuracy of 88%. A sensitivity of 100% for anchored and moored and 97% for underway class can be observed for VGG16. This practically means that confirmed anchored or moored vessel mobility patterns would be accurately identified as such 100% of the time while confirmed underway patterns would be miss-classified only 3% of cases. Furthermore, VGG16 shows high precision values for all classes, specifically 100% for moored and underway and 96% for anchored. This implies that there were no classes incorrectly classified as moored or underway from another classes while only one anchored class was incorrectly classified as underway. This can be seen in the confusion matrix of the VGG16 model with the best classification performance out of the 5 folds (Figure 49(a)). Regarding F1-score, moored class achieved 100% in all performance metrics which means that FPs and FNs are minimized at zero.

InceptionV3 and NASNetLarge present very similar results. A precision of 100% in the moored class is observed, which means that there were no anchored or underway classes falsely miss-classified as moored. On the other hand the precision and F1-score regarding anchored class depicts low results. Indeed, these low values are justified by a large number of FPs. Figure 49(b) and 49(c) depict that the anchored class has 7 miss-classified cases as underway in both models. Furthermore the models present high sensitivity values for anchored and moored class while underway class depicts a moderate value of 80%. Again, moored class achieved 100% in all performance metrics.

Although DenseNet201 presents the worst results in terms of accuracy, moored class achieves again high values for all performance metrics. As the results suggest, the precision regarding anchored class is very low at about 68% and follows the same trend with InceptionV3 and NASNetLarge models for the particular class. However, the model is able to correctly identify confirmed anchored and moored classes as such, 100% and 97% of the time respectively while only 71% of confirmed underway cases would accurately identified correctly. Anchored class presents 11 FPs as illustrated in Figure 49(d) which is strongly associated with the moderate value of F1-score.

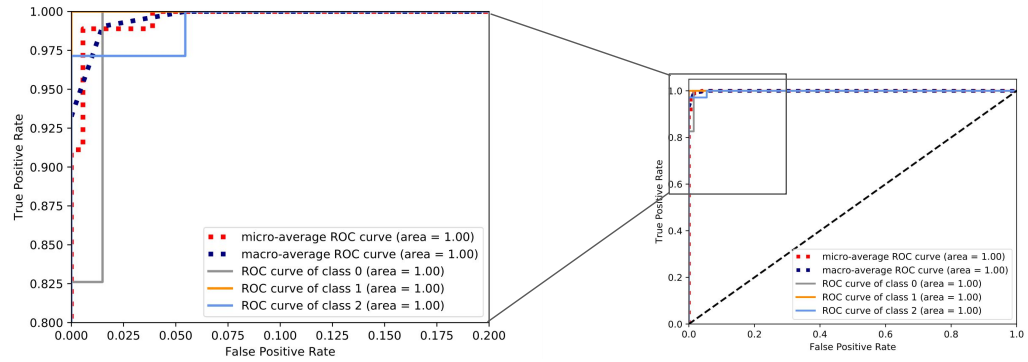
In order to visualize how well the examined models distinguish the classes in terms of the predicted probability, for each CNN we visualized the ROC curves of the model with the best



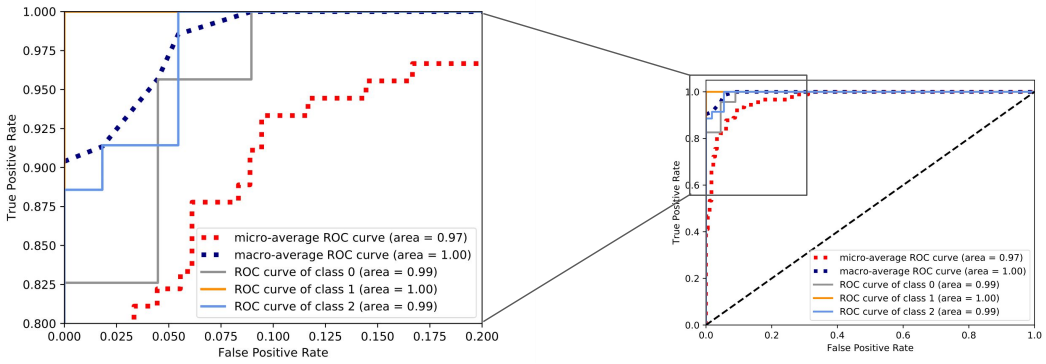
(a) VGG16 (b) InceptionV3

(c) NASNetLarge (d) DenseNet201

Figure 49: Confusion matrix of deep learning models for patterns A with the best classification performance out of the 5 folds.



(a) VGG16



(b) DenseNet201

Figure 50: ROC curve of deep learning models for patterns A

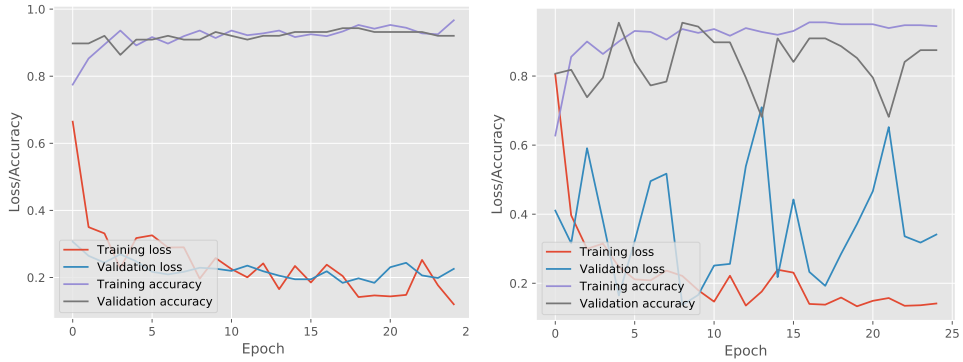
classification performance out of the 5 folds as illustrated in Figure 50. Specifically, Figures 50(a) and 50(b) demonstrate the ROC curves for the models with the highest and lowest accuracy, VGG16 and DenseNet201 respectively. As expected, the area covered (AUC) by VGG16 is bigger as the model is able to discriminate the classes with a much higher probability in comparison with DenseNet201.

Same patterns can be observed between the different models for particular performance metrics. The precision of the anchored class follow a downward trend as the overall accuracy decreased. The same behaviour is presented in the sensitivity of underway class. On the other hand, moored class achieves high values in all performance metrics even with a moderate accuracy in case of DenseNet201. Furthermore, all the CNN models present quite high precision values for moored and underway classes regardless accuracy.

Subsequently, for each CNN we visualized the loss and accuracy of the model with the best classification performance out of the 5 folds during their training in Figure 51. VGG16 demonstrate a smooth training process as shown in 51(a) during which the loss gradually decreases and the accuracy increases. Moreover, it can be observed that the accuracy of both training and validation do not deviate much from one another, a phenomenon that can also be observed for the training and validation loss, indicating that the model does not overfit. The same behaviour is presented in NASNetLarge model as shown in Figure 51(c). On the other hand, in case of InceptionV3 and DenseNet201 their validation loss is either increasing or fluctuating as shown in 51(b) and 51(c) respectively, which means that the models most probably overfit. An interesting fact is that the VGG16 model which contains the least number of layers achieves a better classification performance. This can be explained by the fact that neural networks with more hidden layers require more training data, thus an even larger number of mobility pattern samples is required as an input in these networks.

Additionally, we compared our results with other methodologies for trajectory classification

(a) VGG16 (b) InceptionV3



(c) NASNetLarge (d) DenseNet201

Figure 51: Accuracy and loss (train and test) of deep learning models for patterns A

which are illustrated in this recent survey of Wang et al. [151]. Well-known classifiers such as Random Forests (RFs) and Support Vector Machines (SVMs) are employed by state-of-the-art methodologies for trajectory classification, on features extracted from the AIS messages or data points of the trajectories. Thus, three features from the trajectories during each type of activity (Anchored, Moored and Underway) were extracted: i) the average speed of the vessel, ii) standard deviation of its speed and iii) the haversine distance between the first and the last vessel position. These features were selected because they demonstrate distinct differences between these specific activities. A moored vessel typically has a zero speed while an anchored vessel moves with slightly greater speeds on average due to the effects of the wind and the currents of the sea. A vessel underway presents much higher speeds and the distance between the first and the last position is greater compared to the distance of anchored or moored vessels. Subsequently, these aforementioned features were then fed to two classifiers, Random Forests and Support Vector Machines. The methodologies employed in the comparison are shown below:

- M1: the proposed deep learning methodology of this chapter (image classification by employing the VGG16 model, as it presents the best classification accuracy results).
- M2: the Random Forests classifier with the features extracted from the AIS messages of the trajectories.
- M3: the Support Vector Machines classifier with the features extracted from the AIS messages of the trajectories.

Table 20 illustrates the cross-validation macro average results of the comparison between the different approaches. Results show that our proposed methodology (M1) surpasses the other methodologies (M2, M3) in terms of classification performance. M1 achieved a f1-score of 99.32% and (M2,M3) achieved a f1-score of (96.18%, 91.1%) respectively.

Methodology	Cross Validation		
	Precision	Recall	F1-score
M1	0.9905	0.9912	0.9932
M2	0.966	0.9603	0.9618
M3	0.924	0.9138	0.911

Table 20: Comparison of methodologies on patterns A.

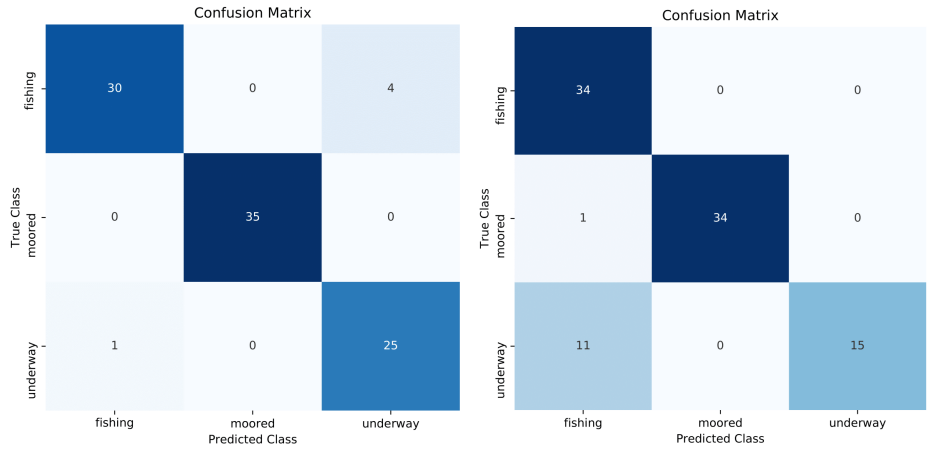
Model	Labels	Precision	Recall	F1-score	Overall Accuracy
	fishing	0.97	0.88	0.92	
VGG16	moored	1.00	1.00	1.00	0.9473
	underway	0.86	0.96	0.91	
	fishing	0.74	1.00	0.85	
InceptionV3	moored	1.00	0.97	0.99	0.8736
	underway	1.00	0.58	0.73	
	fishing	0.6	0.97	0.74	
NASNetLarge	moored	0.97	0.94	0.96	0.7578
	underway	1.00	0.23	0.38	
	fishing	0.63	1.00	0.77	
DenseNet201	moored	1.00	0.94	0.97	0.7894
	underway	1.00	0.31	0.47	

Table 21: Classification performance obtained from different pre-trained CNN models for patterns B.

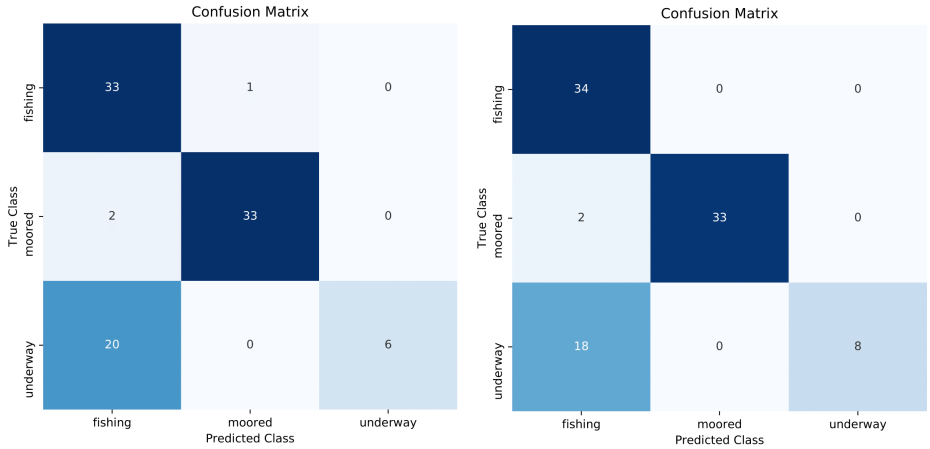
To further evaluate our methodology on vessel activities of utmost importance to the maritime authorities, we performed the same deep learning classifiers employed on patterns A on the second dataset (patterns B) that contains fishing vessels engaged in trawling, longlining, moored at the port or traveling towards a fishing area (underway). A fishing vessel is typically moored at a port, then travels towards a fishing area and is finally engaged in the fishing activity. After the fishing ends, the vessel travels back to the port and moors. Table 21 illustrates the classification results of each model for patterns B.

As the results suggest, VGG16 model achieves again the best classification accuracy of almost 95% followed by InceptionV3 with an accuracy of 87%. NASNetLarge exhibits the worst results with an accuracy of 76% while DenseNet201 presents slightly better results with an overall accuracy of 79%. A recall (sensitivity) of 100% and 96% for moored and underway class respectively can be observed for VGG16. This practically means that confirmed moored vessel mobility patterns would be accurately identified as such 100% of the time while confirmed underway patterns would be miss-classified only 4% of cases. In contrast, fishing presents a moderate recall of 88%. Furthermore, VGG16 presents high precision values for fishing and moored and a moderate precision for underway. This implies that there were no classes incorrectly classified as moored from another classes while only one fishing class is incorrectly classified as underway. The moderate precision of underway is justified by a moderate number of FPs. Indeed, four underway classes were incorrectly classified as fishing as shown in Figure 52(a), which represents the confusion matrix of the VGG16 model with the best classification performance out of the 5 folds. Moored class achieved F1-score of 100% while the remaining classes exhibit a high percentage of about 92% for fishing and 91% for underway.

In InceptionV3 model, moored class presents high values for all performance metrics. Also, a precision of 100% in both underway and moored is observed. On the other hand, the precision of fishing class presents a low value of 74% and justified by the large number of FPs (12 in total) as shown in Figure 52(b). Furthermore, the recall of underway class is low, 58%, as well as the



(a) VGG16 (b) InceptionV3



(c) NASNetLarge (d) DenseNet201

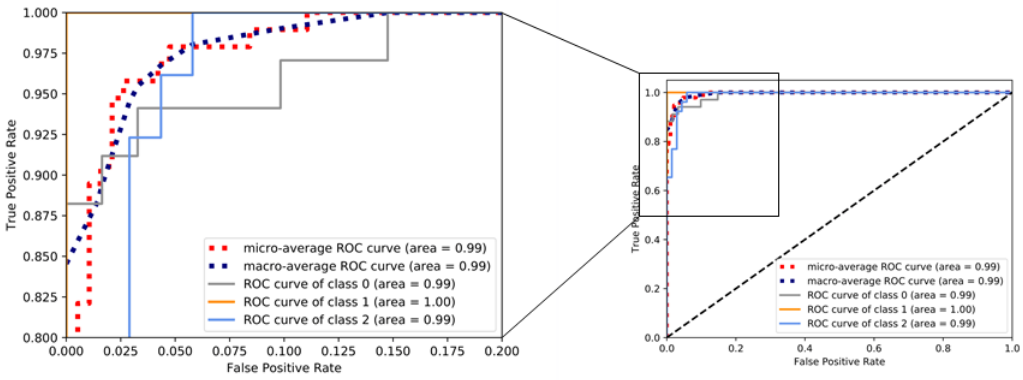
Figure 52: Confusion matrix of deep learning models for patterns B with the best classification performance out of the 5 folds.

F1-score which in turn justifies the large number of FNs (11 in total). The other classes exhibit large values regarding recall, 100% for fishing and 97% for moored.

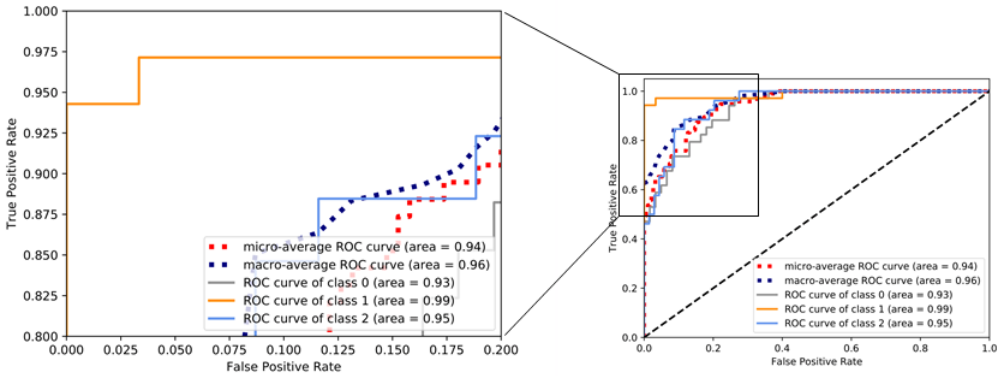
NASNetLarge and DenseNet201 present several similarities. Fishing class has a low value of precision, almost 60% in both models. Figure 52(c) and Figure 52(d) depicts that fishing class has 22 and 20 miss-classified cases respectively. On the other hand, high precision values is observed for moored and underway classes in both models. Moored class presents yet again high values in all performance metrics. Furthermore, recall regarding underway class is very low at about 23% in NASNetLarge and 31% in DenseNet201 and follows the same trend with InceptionV3 model for the particular class. In combination with the low values of F1-score, only a small number of confirmed underway classes are accurately identified correctly.

Figure 53 demonstrates the ROC curves for the models with the highest and lowest accuracy for patterns B, VGG16 53(a) and NASNetLarge 53(b) respectively. For these two CNNs, we visualized the ROC curves of the model with the best classification performance out of the 5 folds.

Subsequently, for each CNN we visualized the loss and accuracy of the model with the best classification performance out of the 5 folds during their training for patterns B as shown in Figure 54. VGG16 demonstrates a smooth training process for patterns B as shown in 54(a) during which the loss gradually decreases and the accuracy increases. Almost, the same behaviour is presented in InceptionV3 model as shown in Figure 54(b) except for the last epochs where the loss increased. On the other hand, in case of NASNetLarge and DenseNet201 their validation loss is either increasing



(a) VGG16



(b) NasNetLarge

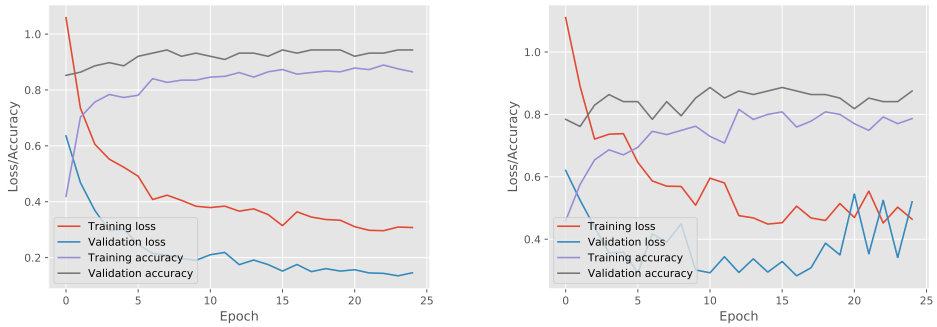
Figure 53: ROC curve of deep learning models for patterns B

Methodology	Cross Validation		
	Precision	Recall	F1-score
M1	0.9452	0.9528	0.9465
M2	0.7523	0.772	0.7620
M4	0.8455	0.8436	0.8446

Table 22: Comparison of methodologies on patterns B.

or fluctuating as shown in 54(c) and 54(d) respectively, which means that the models most probably overfit. Indeed the growth rate of validation loss is extremely high in case of NASNetLarge.

Finally, we compared our classification methodology results to the approach proposed in [14] for the fishing vessels (M4). Authors in [14] use a set of features suitable for the fishing activities which are then used by Random Forests to classify the vessel activities. Moreover, we also used the methodology of the previous experiment (M2) which uses features that are suitable for patterns A. Similarly to the previous set of experiments, we employed only the VGG16 model as it presents the best overall classification accuracy. The macro average results are reported in Table 22. The results again validate that the proposed approach (M1) is by far more accurate than the set of features selected specifically for the classification of fishing activities (M4). Furthermore, results demonstrate that the methodology M2 does not perform well (f1-score of 76.2%). This is explained by the fact that the features used for patterns A are not able to discriminate between patterns B. On the other hand, the proposed methodology (M1) is able to perform well in both sets of patterns due to the fact that the patterns have distinct visual differences.



(a) VGG16 (b) InceptionV3

(c) NASNetLarge (d) DenseNet201

Figure 54: Accuracy and loss (train and test) of deep learning models for patterns B

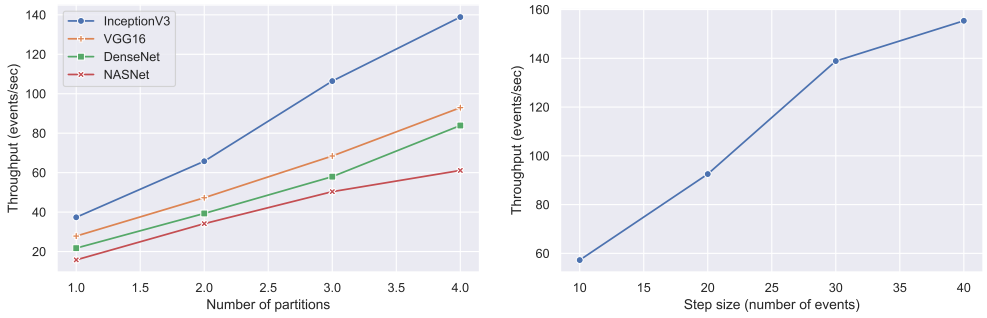
9.3.3 Streaming Evaluation

In this section we provide experimental results of the execution performance of the proposed streaming methodology. To this end, different sets of experiments were conducted in order to evaluate the streaming application's:

- scalability,
- throughput,
- latency,
- execution performance with a GPU.

The experimental evaluation presented below was performed on a cluster consisting of 4 machines with 4 cores (Intel(R) Xeon(R) CPU E5-2650 0 @ 2.00GHz) and 8 GB of RAM each, totalling in 16 cores and 32 GB of RAM, running Ubuntu 18.04 LTS with Python 3.6 and Apache Kafka 2.4.0.

To evaluate the scalability of our approach, we run experiments using 1, 2, 3 and 4 machines incrementally, each machine employed with a prediction module and kept the window size and window step constant equal to 4 hours and 30 events respectively. As mentioned in Section 9.2 the streaming methodology uses a sliding window over the streams of AIS messages to perform the classification. A window size of 8 hours is sufficient to have a fully formed pattern per vessel [14]. Since the highest transmission rate of the AIS protocol is one message every 2 seconds, a window step of 30 events or messages implies that a classification will be triggered by the prediction modules every one minute, a time period during which the pattern would not have changed



(a) Throughput of each CNN architecture with different number of partitions

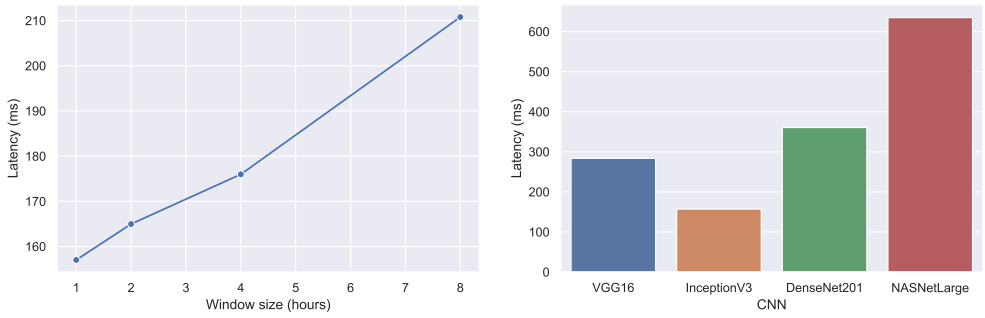
(b) Throughput of InceptionV3 with different window steps

Figure 55: Throughput experiments.

significantly. Furthermore, we repeated each experiment 4 times to evaluate the 4 different CNN architectures (VGG16, InceptionV3, DenseNet, NASNet). Figure 55(a) illustrates the throughput of each CNN architecture with a different number of machines or partitions. It can be observed that the proposed streaming methodology can benefit when more machines and prediction modules are used regardless of the employed CNN architecture. Moreover, it is apparent that InceptionV3 is the fastest CNN architecture achieving a maximum throughput of 140 events per second, outperforming all of its contestants. The next CNN architecture is VGG16 with a throughput of 90 events per second which is similar to DenseNet's throughput of 80 events per second. The slowest CNN architecture is NASNet which achieves a maximum throughput of 60 events per second. The performance of the two last architectures which presented the worse results can be explained by the fact that these architectures are more complex than the former two consisting of more hidden layers and pooling layes. To further evaluate the throughput of the proposed methodology, we selected InceptionV3 which is the fastest CNN architecture and performed experiments using 4 machines and a varying number of window steps. Figure 55(b) visualizes the results of the aforementioned experiment using window steps that range from 10 events to 40 events. The value of window step denotes how often the classification will be triggered, thus the smaller the number of the step the larger the amount of classifications that are triggered resulting in a lower throughput. Indeed this is confirmed by the experiments in Figure 55(b) where the throughput is 155 events per second when using a step of 40 and 57 events per second when using a step of 10.

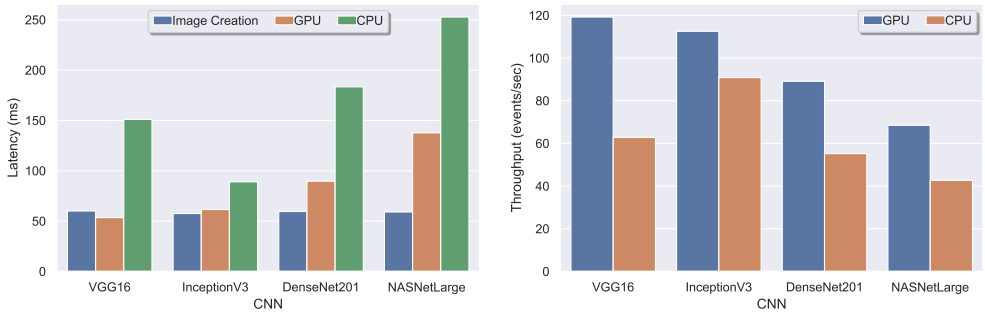
The first experiment which yielded the results of Figure 55(a) has also provided the average processing latency of each CNN architecture in Figure 56(b). Processing latency is defined as the amount of time required by a CNN to perform the image classification. Again, it can be observed that the InceptionV3 is the fastest CNN architecture with a processing latency of 157 milliseconds. Next, is the VGG16 network with a latency of 283 milliseconds followed by DenseNet (360 milliseconds) and NASNet (634 milliseconds). To evaluate the image latency which is defined as the amount of time required to create an image from the AIS messages, we kept the same configurations of the first experiment (4 machines and a window step of 30 events) but changed the window size. Figure 56(a) illustrates the average image latency with a window size of 1, 2, 4 and 8 hours. Since a larger time window contains more AIS messages it is expected that the creation of the image will be greater in larger window settings. As shown and confirmed in Figure 56(a), the average image latency in an one-hour window is 157 milliseconds and in an eight-hour window the latency can reach up to 210 milliseconds.

To further evaluate our approach, we demonstrate experimental results with and without the use of a GPU. The following experiments presented below were performed on a commodity machine with 8 logical cores (Intel(R) Core(TM) i7-7700HQ CPU @ 2.8GHz), 32 GB of RAM and a GPU (NVIDIA 1050 with 4GB of VRAM), running Windows 10. To demonstrate the performance boost



(a) Image latency with different window sizes (b) Latency of each CNN architecture

Figure 56: Latency experiments.



(a) Latency results (b) Throughput results

Figure 57: Execution performance experiments with and without GPU.

by GPU we ran an experiment with a window size and step of 4 hours and 30 events respectively twice for each CNN architecture, with and without the use of the GPU. The experimental results are presented in Figure 57. Specifically, Figure 57(a) illustrates the image latency and the latency of each CNN architecture. The image latency at each experiment is approximately 60 milliseconds (at least two times faster than the same experiment with a lower-frequency CPU in Figure 56(a)). On the other hand, a tremendous performance gain can be observed in the processing latency of each CNN. The processing latency of VGG16, InceptionV3, DenseNet and NASNet is 150 ms, 80 ms, 180 ms and 250 ms respectively without the use of GPU, while when a GPU is employed the processing latency reduces to 50 ms, 60 ms, 80 ms and 140 ms correspondingly. The same pattern can also be observed in the throughput of our approach in Figure 57(b) where the througput of each CNN architecture is increased. An indicative example is VGG16 where the throughput from 60 events per second is increased to 120 events per second.

Finally, it is worth noting that all of the experiments presented in this section achieved a throughput higher than the transmission rate of AIS messages per receiver (5 to 8 messages per second) as stated in Section 9.2.4 making the prediction module suitable to be employed on a terrestrial receiver basis.

9.3.4 Compression Evaluation

In order to evaluate the trajectory compression algorithms, some validation criteria were employed. The main validation solutions include performance and accuracy metrics. For performance the following metrics were considered: i) compression ratio and ii) compression time. Compression ratio refers to the ratio of the points remaining after compression (a) compared to the original points (n), being defined as $\frac{a}{n}$ while compression time refers to the total time required for the compression

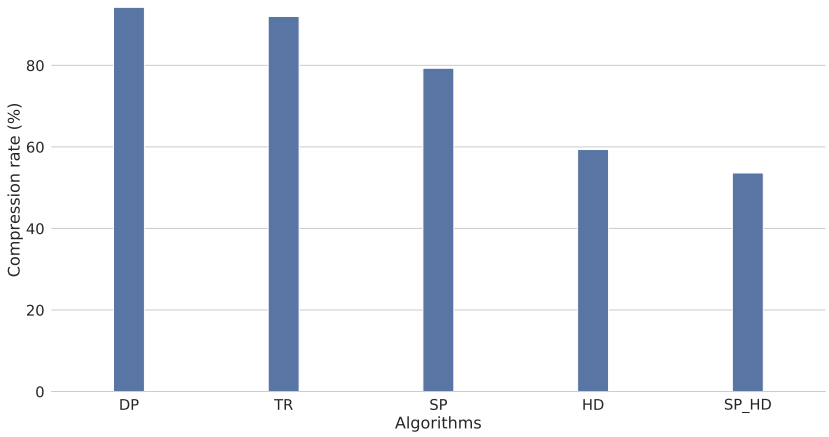


Figure 58: Compression ratio achieved from different algorithms.

execution. On the other hand, accuracy metrics are used to measure information loss and therefore the accuracy of the compression algorithms.

At this point, it should be mentioned that in order to evaluate the compression results we employ a subset of the first dataset which is referred in Section 9.3.1. Specifically, this sub-dataset contains information for 472 unique vessels which have been monitoring for 3 days period starting at March 1st, 2020 and ending at March 3th, 2020. Our goal is to perform some preliminary experiments in order to demonstrate the effect of trajectory compression over the classification accuracy of mobility patterns.

Figure 58 illustrates the compression ratio achieved by the different algorithms. The highest compression achieved by DP with a ratio of 94%. Compression is slightly lower in TR but nevertheless remains extremely high at about 92%. Algorithms that consider time information to decide whether or not to discard a point, tend to preserve more points in a trajectory. DP possess the highest compression rates as it treats a trajectory as a line in two-dimensional space and does not consider the temporal aspect. Compression ratio is decreased in case of SP but maintained at a high level of about 80%. HB and SP_HD hold the lowest compression rates with 59% and 54% respectively. SP_HD incorporates two thresholds simultaneously and it is expected to present the lowest compression ratio. Generally, the less the compression ratio the more points are included in the resulting set.

Besides the compression ratio, it is important to assess the execution time of each algorithm as constitutes an important index to measure efficiency. Figure 59 presents the execution times of all algorithms. DP and TR present the lowest execution among all algorithms. However, the compression presents the highest rates for these algorithms. In fact, DP is executed in only 180 seconds, almost 4 times faster than TR. SP spent considerable more time than DP and TR. Specifically, SP is 27 and 7 times slower than DP and TR respectively. The execution time of HD and SP_HD is quite high, more than two hours in both algorithms. As expected, SP_HD have the worst performance due to its exponential complexity.

As stated in 9.2.5, the examined algorithms are lossy which means that they present some information loss as compressed data cannot be recovered or reconstructed exactly. Thus, it is important to measure the compression performance by evaluating the quality of the compressed data. Existing techniques evaluate the trajectory compression algorithms using various error metrics such as spatial distance, synchronized Euclidean distance, time-distance ratio, speed, heading, acceleration and enclosed area. Muckell et al. [179] employ the average SED and average speed error while in [81] the same metrics with the addition of heading error are applied. Furthermore, three assessment

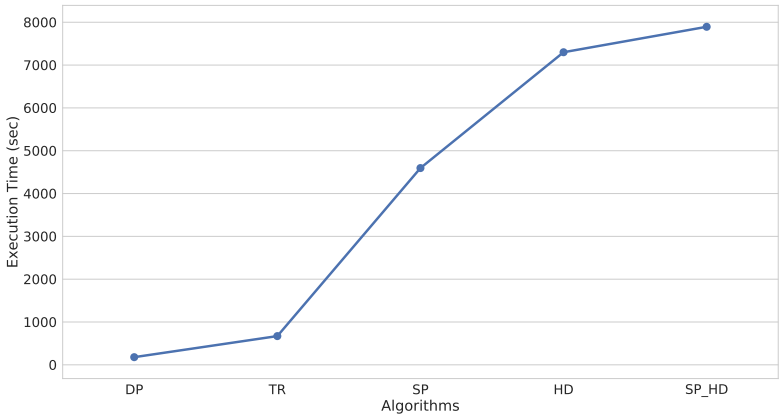


Figure 59: Execution times of compression algorithms.

Compression algorithms				
DP	TR	SP	HD	SP_HD
65%	69%	74%	93%	86%

Table 23: Percentage of correctly classified mobility patterns for each compression algorithm.

criteria used in [184], average SED, spatial and speed error while Liu et al. [185] evaluated several compression algorithms through the average PD, SED and enclosed area. Additionally, in [182] the error is provided by a formula which measures the mean distance error between the original and compressed trajectory.

As there is a trade-off between the achieved compression ratio and the acceptable error, the effectiveness of the trajectory compression algorithms is evaluated in terms of "information" quality obtained after compression using deep learning. The deep learning is applied over the compressed trajectory data in order to evaluate their accuracy. To the best of our knowledge, this is the first research work that investigates and evaluates the impact of trajectory compression techniques over vessel mobility pattern classification through neural networks.

Each unique vessel inside the examined sub-dataset consists of a different number of initial points which ranges from 1 to $25,000$. In order to evaluate the compression results, 47 unique vessels (10%) were selected from the total 472 out of a normal distribution of the points per vessel. Then, the sub-trajectories of these different vessels were extracted and visualized into trajectory images as described in Section 9.2.2 for both the original (uncompressed) and compressed data. Finally, we performed vessel pattern classification on both trajectory data (compressed & uncompressed), by employing the VGG16 model as it achieves the highest accuracy. Table 23 illustrates the percentage $\frac{m}{n}$ of correctly classified mobility patterns extracted from the sub-trajectories images for each compression algorithm (m) compared to the total number of examined mobility patterns (n). In fact, the higher the percentage, the better the quality of the data obtained after compression.

The results demonstrate that accuracy is strongly related to the compression ratio. DP presents the highest compression but exhibits the worst accuracy, about 65%. Similarly, TR achieves extremely high compression but the accuracy is low (69%), slightly better than DP. SP is neither in the worst nor the best results and presents a moderate accuracy of 74%. Nevertheless, the compression obtained in case of SP is 12% lower than TR while the accuracy between these algorithms differs only 5%. HD achieves the highest accuracy as it presents the most correctly classified vessel patterns from the sub-trajectory images with a ratio of 93%. Accuracy is slightly lower in SP_HD but nevertheless remains high at about 86%. The compression achieved in case of HD and SP_HD is

significantly lower in comparison with the other algorithms, but it is accompanied with extremely high accuracy, especially in case of HD.

9.4 Discussion

The proposed work provides a novel approach for the classification of trajectories in near real-time, a task that is of utmost importance in the surveillance of vessels in the maritime domain. The novelty behind our approach lies in the fact that trajectories are converted into a space normalization format that resembles an image which are then classified by a Convolutional Neural Network, a methodology that to the best of our knowledge has received limited study in the literature. This fusion of the fields of trajectory classification and computer vision can deliver results comparable to or even better than the state-of-the-art (see Section 9.3.2).

Classification techniques may involve a number of pre-processing steps in order to develop a classifier that yields optimal results. Initially, a thorough data analysis and cleaning is typically performed. Then, a clustering technique may be employed that often requires some form of user-defined parameters to group similar trajectories together. Afterwards, a careful selection and analysis of optimal features for the label or labels to be identified, follows. Finally, the analysis and selection of several classifiers is performed to achieve high-precision results [151]. Therefore, the overall process of feature selection and classification is not an easy task. This difficulty of finding the best classifier can be observed in the fishing patterns where vessels pose a similar behavior, thus different features and classifiers need to be employed for each pattern [67, 68]. Although mobility patterns can be similar (e.g. fishing patterns have similar behavior in terms of speed and frequency of turns [67, 14]), they differ visually (see Figures 39(d) and 39(e)). This distinct visual difference of mobility patterns is by computer vision techniques for their classification. The main advantage of image classification for mobility patterns over other techniques is 1) the decrease of the complexity of the aforementioned steps, 2) the removal of user-defined parameters, and 3) the distinct visual difference of mobility patterns. Although, the data cleaning process cannot be avoided, the parameter selection, feature selection and analysis and the search of an optimal classifier can be entirely skipped. For the classification of an image, only one classifier is needed (e.g. CNNs) whether this image illustrates a fishing pattern or a Search and Rescue (SAR) pattern.

Due to the similarity of fishing patterns (trawling, longlining, purse-seiners), Souza et al. [67] proposed a different classification scheme for each pattern (3 classifiers with 3 different sets of features) achieving an f1-score of 97%. Each classifier was trained to identify trajectories between fishing or underway, where fishing corresponds to the fishing activity of the respective vessel type. This classification performance is comparable to the performance of the proposed methodology ($\approx$ 95%) which uses a Convolutional Neural Network for the classification of four patterns in total (patterns A and B). Authors in [14] presented a set of features that can be extracted in real-time and classify the same fishing activities that this work does achieving a lower classification performance (87%). The main disadvantage is that these features cannot be used to classify other mobility patterns and they are only suitable for the fishing patterns in the study. Kapadais et al. [126] proposed a time-series shapelet classification technique to identify SAR patterns. The main drawback in such techniques is that they require data to be available at fixed intervals (e.g. hourly, weekly, monthly, yearly), a characteristic that is not present in AIS messages (see Section 9.3.1). Therefore, techniques need to be devised that can fill in the gaps or remove data points in order to create a time-series suitable for classification, a problem that can be avoided by our proposed approach.

Compared to Chen et al. [76] in which authors convert AIS trajectories into a mobility-based trajectory structure (MB) that resembles a low-resolution image, our approach can create high-resolution images able to capture a trajectory's behavior in its entirety. Since the surveillance space in [76] is segmented into large cells, cases where the vessels perform micro-movements that

eventually form a different mobility pattern such as fishing activities will be misclassified. Authors in [76] aim at the classification of the transportation mode (e.g. Carrying Cargo, Tankers, Towing or Fishing) of vessel trips when travelling from an origin port to a destination port and achieve an f1-score of 84.7%. A similar approach has been employed in [78] for the classification of the transportation mode of civilians (e.g. walking, bike, bus, taxi) achieving a maximum accuracy of 83.2%.

Despite the fact that several classification methodologies have been proposed in the literature, each one has its limitations and possible drawbacks. Similarly, although the proposed methodology has tackled most of the issues found in previous literature, it can be observed that there is a cold start problem. This means that the proposed approach can accurately classify vessel activities only when enough AIS messages are accumulated for a predefined period of time (e.g., four-hour time window) and not when the vessel first appears in the surveillance area. Furthermore, deep neural networks are computationally expensive not only during the training process but in the classification process as well. From the experimental results it can be seen that the maximum throughput for data originating from a single AIS receiver is approximately 155 events per second without a GPU. At a global scale where the frequency can reach up to 16,000 events per second, a huge cluster is required to handle the volume and the velocity of the data. Therefore, the solution to address the high-frequency and high-volume streams of AIS data is to deploy one prediction module (see Section 9.2.4) per AIS receiver, thus creating a global distributed network of prediction modules.

9.5 Summary

In this work, we presented a novel and high-accuracy trajectory classification approach over streams of AIS messages. The goal of our methodology is to provide an efficient and alternative way to treat the problem of streaming vessel activity classification problem as an image classification task using state-of-the-art deep learning algorithms and to create a universal approach for the classification of vessel activities. This universal approach can be achieved through the distinct visual difference that most of the mobility patterns, if not all, have in the maritime domain. Therefore, a common classifier can be used for all patterns after they have been converted to images.

To demonstrate the applicability of our work in real-world conditions, two real-world datasets of AIS messages was used, one collected from a terrestrial, vessel-tracking receiver installed on our premises and another one extracted from the MarineTraffic database. As deep learning approaches require a large amount of data in order to perform an accurate classification, transfer learning is employed. Experimental evaluation showed that the proposed approach achieves a state-of-the-art classification accuracy. The proposed approach has been evaluated on five mobility patterns of the maritime domain in total, namely anchored, moored, underway, trawling and longlining. The achieved classification performance exceeded 89% for all CNN models in the first dataset (patterns A) with the VGG16 yielding the best accuracy of 99%. In order to evaluate the methodology in a real-world use case of interest, we demonstrated its classification performance on four mobility patterns of fishing vessels: trawling, longlining, moored and underway. VGG16 model presented again the best classification results with an accuracy of 95%.

Scalability experiments demonstrated that the proposed methodology can deal with the event rate of real-world streams of AIS messages and can scale up to multiple machines allowing a near real-time execution performance. InceptionV3 achieved a maximum throughput of 140 events per second outperforming all of its contestants followed by VGG16 with a throughput of 90 events per second. Regarding latency, InceptionV3 is again the fastest CNN architecture with a processing latency of 157 milliseconds followed by VGG16 latency of 283 milliseconds. In addition, the superiority of GPU versus CPU was verified through experiments regarding throughput and latency.

Preliminary experiments with compression algorithms demonstrated that streams of AIS messages can be compressed more than 50% of the original number of messages with a relatively low

impact on the classification performance. The Heading-Based compression algorithm achieved the best compression ratio while the accuracy remained at a fairly high level, only 7% of cases were miss-classified.

Part V
Trajectory compression

10 A comparison of trajectory compression algorithms over AIS data

10.1 Introduction

In recent years, the number of vessel tracking data has drastically increased, following an impressive exponential trend. This is due to the fact that not only all larger vessels are obliged to be equipped with an Automatic Identification System (AIS) transponder, but smaller vessels are also adopting this technology voluntarily. AIS is a global tracking system that allows vessels to transmit data about their whereabouts. Through the AIS, vessels are able to be aware of vessel traffic in their vicinity and avoid potential collisions. Despite the fact that the initial purpose of the AIS was safety, it did not take long for the maritime authorities to exploit it for the identification of illegal vessel activities and the monitoring of vessels' behavior.

The exploitation of AIS data from the maritime authorities has shifted the focus of researchers' attention towards the development of trajectory mining techniques. Such techniques allow the authorities to further take advantage of the trajectories formed from the AIS messages either in real-time [9, 11, 14] or on historical data [43, 122, 13]. The main challenge to be tackled in the development of trajectory mining techniques is the data's huge volume generated globally and the associated high frequency. The increasing amount of data poses new challenges related to storing, transmitting, processing, and analyzing these data [186, 177, 175, 176, 178]. For instance, the transmission rate of the AIS protocol can reach a frequency of one message every two seconds per vessel. Such a frequency in combination with the approximately $200,000$ vessels using the AIS worldwide, yields a total amount of $16,000$ messages per second and 46GB of data per day. Thus, the enormous volumes of trajectory data can quickly overwhelm available data storage systems and the redundant information often contained in these data can overwhelm human analysis. In general, the more points collected, the more accurate a trajectory representation becomes. Nevertheless, most vessel movements on the water are relatively steady in space, thus the shape of a vessel trajectory can be represented by only a small portion of carefully selected points.

A typical approach towards these challenges is to reduce the size of trajectory data by employing compression techniques. A compression algorithm is required to not only compress redundant position information but also retain the vessel's behaviour information included in the trajectory. In other words, trajectory compression aims at substantial reductions in the amount of data while minimizing the loss of information and at the same time preserving the quality of trajectory. Quality is strongly related to the information loss and measured through metrics that calculate the applied error over a trajectory if a certain point is discarded. Therefore, it is become clear that there is a trade-off between the compression rate and the quality of trajectory achieved after compression [97].

The objective of this work is to evaluate some of the most representative lossy compression algorithms over AIS data. Specifically, the algorithms evaluated are: Douglas-Peucker (DP), Time-Ratio (TR), Speed-based (SP), Heading-based (HD), Time-Speed-Heading-based (TSH), STTrace (STT), SQUISH (SQ), Dear-Reckoning (DR) and Open-Window Time-Ratio (OWT). The comparison is based on the compression ratio and the execution time achieved across a real-world dataset. Their performance characteristics were compared against different dynamically defined thresholds. Each trajectory is different from others, hence it is beneficial to select the appropriate threshold for each trajectory according to its characteristics. Threshold refers to the discarding criterion employed by each algorithm. Thus, we suggest an automatic technique, in which a different threshold is applied in the corresponding algorithm, which depends on the actual features and peculiarities of each trajectory. In this way, we have eliminated the need of arbitrary user-defined thresholds.

The examined algorithms are lossy which means they present information loss and therefore it is

important to measure the quality of the compressed AIS data. Different error metrics are used in the literature to evaluate compression. For instance, the error in [182] is provided by a formula which measures the mean error of the compressed representation in terms of distance from the original trajectory. In [179] the average and median synchronized Euclidean distance (SED) as well as the speed error are employed in reference to compression ratio while in [81] the same metrics are used with the extra addition of heading error. Furthermore, in [184] the average SED, spatial and speed error are utilized. In this research work, the quality of the compression is evaluated by employing similarity measures over both the original and the compressed trajectory data, in order to evaluate their accuracy. The purpose of a similarity measure is to obtain a quantitative measure between any two trajectories, thus to identify to what extent two objects are similar. Most of the well-established similarity measures selected in this work, consider the shape of the trajectory which consist a characteristic of utmost importance in the maritime domain and can provide insights about vessels' behavior [14, 16].

The main contributions of this study are as follows:

- nine trajectory compression algorithms are compared and evaluated. The examined algorithms are suitable either for historical data or real-time data streams compression;
- the compression evaluation is based on six different trajectory similarity measures which consider the spatio-temporal aspect of the trajectory;
- different dynamically defined thresholds are applied by each compression algorithm based on the actual features and peculiarities of each trajectory, thus eliminating the need for arbitrary user-defined thresholds;
- Dead-Reckoning is considered the best algorithm when dealing with AIS data, followed by Douglas-Peucker and Open-Window Time-Ratio. These algorithms presented the most suitable performance in terms of compression ratio, execution time and information loss.

The rest of the chapter is structured as follows. Section 10.2 describes the compression algorithms while Section 10.3 presents the similarity measures. Section 10.4 details the compression evaluation and presents the results, while Section 10.5 summarizes the merits of our work.

10.2 Trajectory Compression Algorithms

Compression algorithms can be classified into two broad categories depending on the moment the compression procedure is performed [187]: offline and online.

More specific, offline compression is performed after a trajectory has been fully generated by discarding redundant points from the original trajectory. Online algorithms compress the trajectory during the point collection process which has the advantage of supporting online applications [188].

Offline algorithms, also known as batch algorithms, are able to obtain more accurate results and present smaller errors than online algorithms as the algorithm has a global view of the entire trajectory. On the other hand, online algorithms remove redundant data from trajectories as they occur, avoiding the unnecessary transfer of data over the network, improving data storage and reducing the memory space. With the exponential growth of AIS data, online algorithms seem to be a more appropriate approach for data compression. AIS data will eventually become prohibitively "big" for storage and transmission, thus offline algorithms are not suitable, as only collected historical data can be compressed.

In this research work, the examined algorithms are lossy in the sense that attempt to preserve the major characteristics of the original trajectory by removing the less significant data when compared to the original data. The main advantage of lossy compression techniques is that they can reduce storage size while maintaining an acceptable degree of error [179].

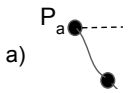


Figure 60: Douglas–Peucker algorithm. (a) Line segment formed by anchor and float points. (b) The point (P_1) with the maximum perpendicular distance from line segment is selected. (c) The process is repeated using recursion on each line segment. (d) New simplified trajectory.

10.2.1 Offline compression algorithms

Douglas-Peucker (DP). Douglas-Peucker algorithm was proposed in 1973 [42] as an approximate simplification method with error bound guarantee. The process of the algorithm is illustrated in Figure 60. An AIS trajectory is represented as a point set S = $\{P_a, ..., P_b\}$, as shown in Figure 60a. Initially, the first (anchor) and last (float) point from the trajectory are selected and these two points form a line segment $LS_{P_a \to P_b}$. The starting curve is a set of points and a threshold (distance dimension) $\varepsilon > 0$. After, the algorithm calculates the perpendicular distance PD, of each point (P_i) between the trajectory and its projection on the line segment $LS_{P_a \to P_b}$. Subsequently, the point with the maximum perpendicular distance d_{P_1} from $LS_{P_a \to P_b}$ is selected (P_1). If $d_{P_1} > \varepsilon$, the point is retained in the resulting set and becomes the new float point for the first segment, and the anchor point for the second segment as shown in Figure 60b. Thus, the original trajectory is splitted into two sub-trajectories $STr_{P_a - P_1}$ and $STr_{P_1 - P_b}$. Otherwise, if $d_{P_1} < \varepsilon$, all the points between P_a and P_b are discarded. This process is repeated using recursion on each line segment (Figure 60c). The algorithm halts when the maximum distance between the original trajectory and the line segments is below ε. When the recursion is completed, a new simplified trajectory is generated of only those points that have been marked as kept (Figure 60d). In the worst case scenario its running time is $\mathcal{O}(n^2)$ where n is the amount of points in the trajectory. An improved version which includes convex hulls (DPhull) is presented in [189] with running time $\mathcal{O}(n \log n)$.

Top-down Time-Ratio (TR).

The main drawback of line generalization algorithms like Douglas-Peucker, is that when dealing with trajectory data of moving objects, they do not consider the temporal aspect. They treat each trajectory as a line in two-dimensional space. But a trajectory has an important extra dimension, time. Time-Ratio algorithm computes the distances between pairs of estimated temporally synchronized positions, one on the original and one on the corresponding approximated trajectory as illustrated in Figure 61.

For each point on the original trajectory such as P_i, the temporally synchronized point $P_{i'}$ is located on the approximated trajectory Tr_{Pa-Pb} and the coordinates (x'_i,y'_i) of Pi' are calculated using linear interpolation as:

$$\begin{aligned} x'_i &= x_a + \frac{t_i - t_a}{t_b - t_a}(x_b - x_a) \\ y'_i &= y_a + \frac{t_i - t_a}{t_b - t_a}(y_b - y_a) \end{aligned} \tag{48}$$

After the temporally synchronized points are determined, the next step is to calculate the distance between Pi' and Pi. If this distance is greater than a user-defined threshold, the particular

Figure 61: Time Ratio algorithm.

Figure 62: Speed Based algorithm.

point is included in the resulting set otherwise it is discarded. By including the temporal information, as well as spatial data in its compression heuristic, the algorithm provides more accurate results. The worst-case running time of TR is $\mathcal{O}(n^2)$ since it extends the original Douglas-Peucker algorithm.

Speed-based (SP).

Speed-based algorithm exploits the speeds from subsequent segments of a trajectory. If the absolute value of speed difference of two subsequent trajectory segments, for example $|V_{STr_{Pb-Pc}} - V_{STr_{Pa-Pb}}|$, is greater than a threshold, the point in the middle (P_b) is retained otherwise it is discarded. The process continues until all points in the trajectory are examined. The algorithm is illustrated in Figure 62.

Heading-based (HD).

Heading-based algorithm exploits the angle formed by subsequent segments of a trajectory. Initially, the distances of continuous segments of the trajectory are calculated; i.e. d_STr_{Pa-Pb} and d_STr_{Pb-Pc}. Then for each triangle formed by three continuous points like $P_aP_bP_c$, the distance (length) of the opposite side of the examined angle is calculated e.g. d_STr_{Pa-Pc}. Knowing the size of three sides and utilizing the law of cosines, the angle is calculated. For example we can calculate A_1 ($\angle\ PaPbPc$) as $c^2 = a^2 + b^2 - 2ab\cos\gamma$, where γ denotes the examined angle (A_1) contained between the sides of length a (d_STr_{Pa-Pb}) and b (d_STr_{Pb-Pc}) and located opposite the
s ıgth c (d_STr_{Pa-Pc}). If the angle of two subsequent segments is greater than a threshold,
t in the middle is retained otherwise it is discarded. The process continues until all points
i ɛctory are examined. The algorithm is illustrated in Figure 63.

Time-Speed-Heading-based (TSH). By integrating the concepts of time ratio distance, speed and heading, a new algorithmic approach can be obtained. The proposed algorithm, namely Time-Speed-Heading-based, considers ship behaviour characteristics in terms of motion features related to speed and course variations but also examines time, in the sense of synchronized euclidean dis-

Figure 63: Heading Based algorithm.

tance of temporally synchronized points. The algorithms sets an anchor point and then gradually "opens the window". In each recursion, three halting conditions are examined, the synchronous euclidean distance measure, the speed difference and the heading difference between trajectory subsegments. If SED, speed or heading are greater than a SED, speed and heading threshold respectively, the point is retained otherwise it is discarded. Of note, the integration of different compression algorithms is not a novel idea but was originally proposed by Meratnia and de By in [182].

10.2.2 Online compression algorithms

Spatiotemporal trace – STTrace (STT). STTrace was proposed by Potamias et al. [178] and introduces the use of SED as an error metric. The algorithm is designed to preserve spatiotemporal, heading and speed information in a trace. Initially, every incoming point along with its SED is inserted in the allocated memory. The SED is calculated after its successor is stored in the sample. As soon as the allocated memory gets exhausted and a new point is examined for possible insertion, the sample is searched for the item with the lowest SED, which represents the least possible loss of information. If SED of the inserted point is larger than the minimum one found already in the sample, the current processed point is inserted into the sample at the expense of the point with the lowest SED. When a point is excluded from the memory, SED attributes for the neighboring points of the removed one are recalculated along with a reassessment of the new minimum SED. The time complexity of the algorithm is $\mathcal{O}(\frac{1}{n} \log \frac{n}{m} \log m)$ where n refers to trajectory size (number of points) and m to the memory size.

Spatio Quality Simplification Heuristic – SQUISH (SQ). SQUISH is presented by Muckell et al. [179] and although it is a similar approach to STTrace, it differs on the criteria of adding a point in memory. The algorithm requires an input parameter, i.e. the size of a buffer that contains the number of points that will exist in the final data after compression. Initially, all incoming points are inserted in the buffer until is full and any new incoming point requires the removal of another point that is already stored inside the buffer. The selection of the point to be removed is based on the SED estimation if that point is removed. In other words, the removed point is that with the minimum estimated SED error, which will introduce the lowest amount of error in the compression. Subsequently, the algorithm estimates the upper bound SED error of the adjacent neighbors by adding the SED value of the deleted point. Since removal of any point would require reassigning priorities to every point in the buffer, the prioritization algorithm uses local optimization instead of a more accurate global approach. The time complexity of the algorithm is $\mathcal{O}(n \log(n'))$ where n refers to trajectory size (number of points) and n' to the compressed trajectory size.

Dead-Reckoning (DR). Dead-Reckoning algorithm, proposed by Trajcevski et al. [180], employs a prediction location strategy in order to estimate the localization of the next trajectory point in time, using the point's current position and velocity. The algorithms requires a single tolerance value as an input parameter, which specifies the maximum distance a future position can deviate from the estimated position. Then, the distance between the predicted location and the actual location of an incoming point is calculated. If this distance is greater than the defined tolerance, the point is included in the compressed trajectory, otherwise, the point is discarded. The algorithm has a Sliding Window approach with $\mathcal{O}(n)$ time complexity. This complexity is due to the fact that it takes only O(1) time to compare the actual and predicted locations of each point.

Opening-window Time-ratio (OWT). Opening Window algorithm proposed by Meratnia and de By [182] starts by defining a segment between the anchor (first data point) and the float (third data point) in a trajectory. Subsequently, the perpendicular distances of each intermediate point are calculated and if they are under a predefined error threshold, the float is moved to the next point. The process continues until the perpendicular distance of a point inside the window exceeds the error threshold. Then, two strategies can be employed depending on which version of the algorithm

Algo-rithm	Time Cost	Space Cost	Criterion	shape	time	sog	cog
DP	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	Spatial distance (PD)	✓	×	×	×
TR	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	Time-ratio distance (TRD)	✓	✓	×	×
SP	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	Speed difference	×	×	✓	×
HD	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	Heading difference	×	×	×	✓
TSH	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	SED, Speed diff, Heading diff	✓	✓	✓	✓
STT	$\mathcal{O}(\frac{1}{n}\log\frac{n}{m}\log m)$	$\mathcal{O}(m)$	SED	✓	✓	×	×
SQ	$\mathcal{O}(n\log(n'))$	$\mathcal{O}(m)$	SED	✓	✓	×	×
DR	$\mathcal{O}(n)$	$\mathcal{O}(1)$	SED distance, speed	✓	✓	✓	×
OWT	$\mathcal{O}(n^2)$	$\mathcal{O}(n)$	Time-ratio distance (TRD)	✓	✓	×	×

Table 24: Summary of trajectory compression algorithms

is executed: i) the data point causing the threshold violation is the one selected to remain (Normal OW) and ii) the predecessor of the actual float point is selected (Before OW). The selected point becomes the anchor and the float is set two points ahead. If no threshold is exceeded, the float is moved one up the data points and the method continues, until the entire series has been transformed into a piecewise linear approximation. Opening-Window Time-Ratio is an improved version of the OW algorithm. It employs the time-ratio distance measuring technique presented in 10.2.1, thus considering the temporal aspect of the trajectory in the compression process. The worst-case running time of Opening Window algorithms is $\mathcal{O}(n^2)$.

Table 24 depicts the basic characteristics of the compression algorithms compared in this study. Specifically, the compression algorithms are categorized into offline and online mode and the computational complexity is presented in terms of time complexity and memory (space cost). Furthermore the error criterion of each algorithm is presented. Each trajectory compression algorithm employs particular heuristic(s) in order to discard or retain points. Different error metrics such as perpendicular distance, time-ratio distance, synchronized Euclidean distance etc. are utilized as criteria to decide which points to discard. Finally, the table illustrates the different trajectory features each algorithm considers in compression. For example, DP only considers the geometrical shape of a trajectory while TR takes also time into consideration. TSH considers all the examined features followed by DR which considers all except heading. As the table suggests, online algorithms tend to consider more features in compression than offline.

10.2.3 Threshold definition

One of the challenges is to define the required thresholds to be employed by the compression algorithms. Setting the proper threshold can significantly affect the compression results in terms of compression ratio and achieved quality. Threshold determination is an application-dependent process and varies between different studies. For example, in [190, 88] the 0.8-times the ship length-threshold is utilized while in [191] the threshold distance is set experimentally to $500m$. Each trajectory is different, hence it is beneficial to select the appropriate threshold for each trajectory automatically.

In general, each of the examined algorithms follows the steps below:

- group the points and create a trajectory of each object based on an identifier. This practically means that the number of trajectories in a dataset is as large as the number of objects

- compress the whole trajectory of each identifier
- write the points remaining after compression to a file grouped by identifiers

As we group the points for each identifier (object) in the dataset, we extract the trajectories (one trajectory for each object). In order to determine the threshold that each algorithm will use, a dynamic process is proposed: for each individual trajectory a different threshold is automatically defined based on an average. Specifically, the average value refers to the discarding criterion of each algorithm. For example, in case of DP the average epsilon (ε) is calculated while for STTrace two thresholds are calculated, the average speed and the average orientation. Then we use this average value as a reference point to define a common rule for the threshold calculation. This practically means that in every trajectory of each dataset a different threshold is applied in the corresponding algorithm, which depends on the actual features and peculiarities of this trajectory. Thus, we have eliminated the need of arbitrary user-defined thresholds.

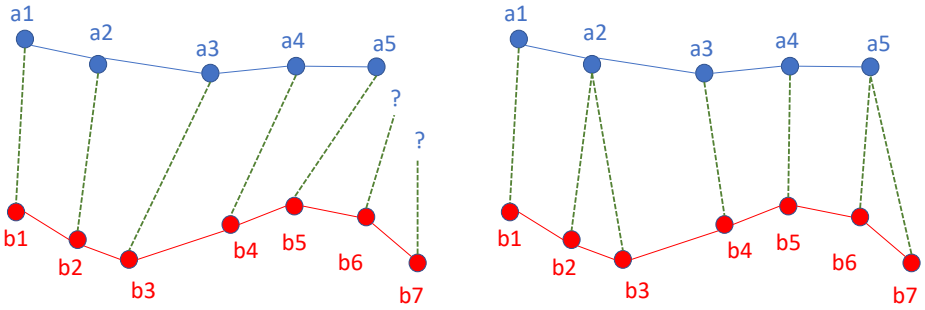
10.3 Trajectory Similarity Measures

In this section, we describe the similarity measures that were used in the experiments to evaluate the trajectory compression algorithms. Several trajectory similarity algorithms have been proposed and evaluated in the literature [192, 193, 194, 195, 196, 197], each with its own advantages and drawbacks. Similarity measures are employed for different purposes in information retrieval and data mining, such as top-K similarity queries, trajectory outlier detection and clustering techniques [198]. One of the most recent studies [199], has created a hybrid unsupervised learning method for trajectory similarity computation, by combining auto-encoders and convolutional neural networks. The original vessel trajectories are transformed into images which in turn are remapped into two-dimensional matrices. Then, low-dimensional representations of the aforementioned produced trajectory images were obtained using the proposed hybrid network. Finally, the trajectories similarities are measured by calculating the distances between the low-dimensional feature vectors learned from the corresponding trajectory images. The results show that the proposed method outperformed both Fréchet distance and DTW in terms of efficiency and effectiveness.

Most of the well-established similarity measures selected in this work consider both the spatial and the temporal aspect of the trajectory [200]. The spatial aspect or the shape of a trajectory is a key characteristic that is of utmost importance in the maritime domain and can provide insights about vessels' behavior [14]. The well-known Longest Common Sub-Sequence (LCSS) [201] and Edit Distance on Real sequence (EDR) [202] algorithms were not selected because they try to find matching pairs of points based on a user-defined parameter, ε. This parameter is a threshold that defines two points between two trajectories as similar if the distance between the points is less than ε. When comparing compressed and uncompressed trajectories all of the points of the compressed trajectory are contained in the uncompressed trajectory, therefore a parameter of $\varepsilon \geq 0$ would result in misleading results.

10.3.1 Dynamic Time Warping

The Dynamic Time Warping (DTW) is a method that calculates the distance between two temporal sequences or time-series of different length and a one-to-one match between these sequences is not possible. Let A and B be two temporal sequences of spatial or positional data points, $A = [a_1, a_2, ..., a_{n-1}, a_n]$ and $B = [b_1, b_2, ..., b_{m-1}, b_m]$, where n and m is the number of points of A and B respectively and $n \neq m$. These sequences can form a matrix $M = A \times B$, where each point $M_{i,j}$ is the distance between a_i and b_j. The Dynamic Time Warping algorithm then finds an optimal path W between $M_{0,0}$ and $M_{n,m}$. As a result the optimal path W is a set of edges $M_{i,j}$ that connect points of a_i and b_j and the DTW distance is the total distance of these edges:



(a) A one-to-one match between A and B is not possible

(b) DTW distance between A and B

Figure 64: Example of the Dynamic Time Warping distance

$$D_{dtw} = \sum_{x=0}^{k} W_x \tag{49}$$

where k is the number of elements in W. More details on the optimal path finding can be found in [203]. Figure 64 illustrates an example of the DTW algorithm. In Figure 64(a) we can see that a one-to-one match is not possible between sequences of different length. In Figure 64(b) we can see the edges connecting A and B that consist the optimal path W.

10.3.2 Edit Distance with Real Penalty

The Edit distance with Real Penalty (ERP) algorithm is a metric that computes the distance between two numeric sequences or time-series of different length. Similar to the DTW algorithm, ERP creates a distance matrix M that describes the mapping between two series A and B, $M = A \times B$ and tries to find an optimal path between $M_{0,0}$ and $M_{y,y}$, where $y = max(n, m)$ and n and m is the number of elements in A and B respectively. The key difference between ERP and DTW is that ERP allows for gaps or sequences of data points that cannot be matched with any other data point. These gaps are then penalized based on the distance of the unmatched points from a constant value g. The distance between a_i and b_i can be formulated as follows:

$$D_{erp}(a_i, b_i) = \left\{ \begin{array}{ll} |a_i - b_i|, & \text{if } a_i, b_i \text{ not gaps} \\ |a_i - g|, & \text{if } b_i \text{ is a gap} \\ |b_i - g|, & \text{if } a_i \text{ is a gap} \end{array} \right\} \tag{50}$$

More details on the optimal path finding and the computation of the ERP distance can be found in [202].

10.3.3 Fréchet

The Fréchet distance is a similarity measure between curves and is named after the mathematician Maurice Fréchet. This distance metric takes into account the location and the ordering of the points along the curves, thus reflecting the course of the curves. Let $f : [a, b] \rightarrow V$ and $g : [a', b'] \rightarrow V$ be two curves or continuous mappings, where $a, b \epsilon R$, $a \leq b$ and (V, d) is a metric space. Given these curves the Fréchet distance between them is defined as:

$$D_F(f, g) = \inf_{\alpha,\beta} \max_{t\epsilon[0,1]} d(f(\alpha(t)), g(\beta(t))) \tag{51}$$

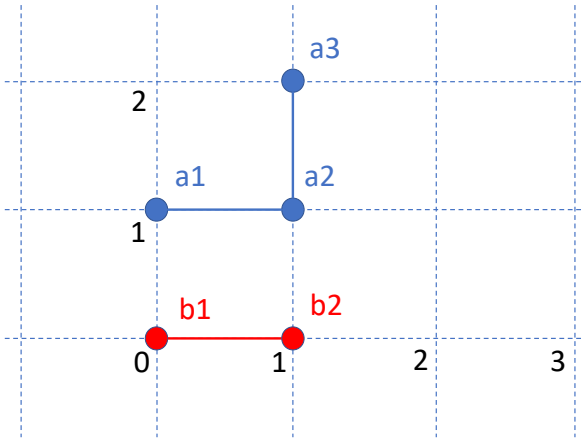


Figure 65: Example of the discrete Fréchet distance.

where α and β are two arbitrary continuous and non-decreasing functions from $[0, 1]$ onto $[a, b]$ and $[a', b']$ respectively. For instance, assuming that a person walks a dog on a leash, both traverse two separate finite curves with different speeds, but neither can go backwards. The Fréchet distance between these two curves is the length of the shortest leash long enough to traverse both separate paths from the beginning to the end of the curves. According to equation 51, $f(\alpha(t))$ would be the position of the dog and $g(\beta(t))$ would be the position of the person at time t or vice versa. More details on the Fréchet distance can found in [204].

10.3.4 Discrete Fréchet

The difference between the Fréchet distance and its discrete counterpart is that the latter is a restriction of the continuous case described in the example with the person and the dog. Considering the previous analogy one would imagine that in the case of the discrete Fréchet distance, snapshots of both the person's and the dog's curves would be taken at discrete time points, thus representing poly-line points or polygonal curves. Given two polygonal curves, $P = [u_1, u_2, ..., u_{p-1}, u_p]$ and $Q = [v_1, v_2, ..., v_{q-1}, v_q]$ where u_i and v_i are their discrete points respectively, a coupling L between these curves is defined as a sequence $(u_{a_1}, v_{b_1}), (u_{a_2}, v_{b_2}), ..., (u_{a_m}, v_{b_m})$ of distinct pairs where $a_1 = 1$, $b_1 = 1$, $a_m = p$, $b_m = q$ and for all $a = 1, ..., q$ we have $a_{i+1} = a_i$ or $a_{i+1} = a_i + 1$ and $b_{i+1} = b_i$ or $b_{i+1} = b_i + 1$, thus respecting the ordering of the points in P and Q. The length of the longest link in L is defined as:

$$||L|| = \max_{i=1,...,m} d(u_{a_i}, v_{b_i}) \tag{52}$$

and the discrete Fréchet distance between P and Q is defined as:

$$D_{dF}(P, Q) = min||L|| \tag{53}$$

where $D_{dF}(P, Q) = D_{dF}(Q, P)$ and $D_{dF}(P, P) = 0$. For a visual illustration, see Figure 65 where there are two poly-lines or polygonal curves: $[a_1, a_2, a_3]$ and $[b_1, b_2]$. The possible couplings between the two are $[b_1\ a_1, b_2\ a_2, b_2\ a_3]$ and $[b_1\ a_1, b_1\ a_2, b_2\ a_3]$ taking into account that the ends of both polygonal curves must match given that we respect the ordering of the points and we do not go backwards. The smallest of the maximum pairwise distances is the discrete Fréchet distance. In this example, the maxima $b_2\ a_3$ of both couplings is equal to two, therefore the minimum of both is also two.

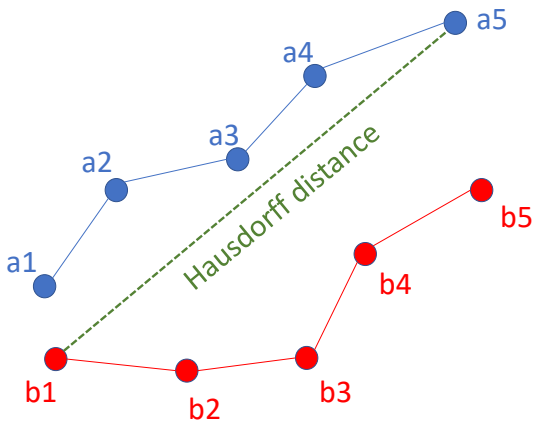


Figure 66: The Hausdorff distance $H(A, B)$ between A and B.

10.3.5 Hausdorff

The Hausdorff distance is defined as the maximum distance of a set to the nearest point in the other set and was named after Felix Hausdorff. Formally, let A and B be two sets of positional data points, $A = [a_1, a_2, ..., a_{n-1}, a_n]$ and $B = [b_1, b_2, ..., b_{m-1}, b_m]$, where n and m is the total number of points of A and B respectively. Then, the Hausdorff distance $h(A, B)$ is defined as:

$$h(A, B) = \max_{a \epsilon A}(\min_{b \epsilon B}(d(a, b))) \tag{54}$$

where d can be any distance metric between the points such as Euclidean or Haversine. Furthermore, the Hausdorff distance is directed which means that $h(A, B) \neq h(B, A)$. For instance, in Figure 66 we can observe that $h(A, B) = d(a_5, b_1)$ where b_1 is the nearest point of B to A and $d(a_5, b_1)$ is the maximum distance. Similarly, $h(B, A) = d(b_5, a_1)$ where a_1 is the nearest point of A to B and $d(b_5, a_1)$ is the maximum distance. Due to this asymmetry, a more general definition is given to the Hausdorff distance:

$$H(A, B) = max(h(A, B), h(B, A)) \tag{55}$$

where $H(A, B)$ is defined as the maximum distance between the two directed Hausdorff distances. Figure 66 illustrates the distance $H(A, B)$ between A and B. For the rest of the paper when referring to the Hausdorff distance, the distance of equation 55 is used.

10.3.6 Symmetrized Segment-Path Distance

The Symmetrized Segment-Path Distance (SSPD) is similar to the Hausdorff distance but the key difference is that instead of calculating the maximum distance of a set to the nearest point in the other set, the mean distance is used. Therefore, equation 54 can be rewritten as:

$$D_{SPD}(A, B) = mean(\min_{b \epsilon B}(d(a, b))) \tag{56}$$

where $D_{SPD}(A, B) \neq D_{SPD}(B, A)$. The SSPD is simply the mean of the two assymetric Segment-Path Distances (SPD) and is defined as:

$$D_{SSPD}(A, B) = mean(D_{SPD}(A, B), D_{SPD}(B, A)) \tag{57}$$

More details on the SSPD can found in [205].

Table 25 summarizes the features of the trajectory similarity measures used in this work.

Measure	Meaning	Time complexity	Parameter free
DTW	Raw representation	$O(nm)$	✓
ERP	Raw representation	$O(nm)$	×
Fréchet	Shape	$O(nm \log(nm))$, where n and m are number of edges in two curves	✓
Discrete Fréchet	Shape	$O(n^m)$	✓
Hausdorff	Shape	$O(k \log k)$, where k is the total number of vertices	✓
SSPD	Shape	$O(k \log k)$, where k is the total number of vertices	✓

Table 25: Trajectory similarity measures

10.4 Evaluation

This section presents the experimental evaluation of the examined compression algorithms. The performance evaluation is conducted in terms of compression ratio, execution speed and distance similarity. The underlying computing infrastructure has been a commodity machine with the following configuration: Ubuntu 18.04.4 LTS 64-bit; Intel Xeon E312xx @ 1.70GHz $\times$ 12; and 64 GiB RAM.

10.4.1 Dataset Description

The analysis of a global AIS dataset is challenging as it combines areas of very different message density due to the patterns that vessels follow, the system's technical characteristics and the means of collection (i.e. satellite or terrestrial network).

There is a lack of temporal and spatial uniformity in global AIS datasets affected by several factors; for example, in coastal areas the spatial and temporal distance between the collected positions is much smaller as opposed to open sea journeys where the lack of coverage can create much sparsely defined trajectories (e.g. a single position received in several hours). In this perspective it makes sense to focus on a representative dataset as the following which is a subset in the Mediterranean sea, rather than examining a very sparse dataset, e.g. in the Pacific or Atlantic ocean.

The dataset used in the following experiments contains AIS messages collected from a Terrestrial AIS receiver (T-AIS). The surveillance area covers the Saronic Gulf (Greece) and the vessels have been monitored for almost one and a half month period starting at February 18$^{\text{th}}$, 2020 and ending at March 31$^{\text{th}}$, 2020. The dataset provides information for 1229 unique vessels and contains $11,769,237$ AIS records in total, each comprising 8 attributes as described in Table 26. A sample of the dataset used can be found in [150].

10.4.2 Compression Evaluation

The compression ratio achieved by the different algorithms is presented in Figure 67. Top-down algorithms DP and TR depict the highest compression ratios. In top-down algorithms, a trajectory is recursively splitted until a halting condition is met. Specifically, DP presents a compression ratio of almost 96%. Compression is slightly lower in TR but nevertheless remains extremely high at

Feature	Description
timestamp	the time at which the message was received (UTC)
ship_id	unique identifier for each ship
lon	the longitude of the current ship position
lat	the latitude of the current ship position
ship_type	AIS reported ship-type
speed	Speed over ground in knots
course	Course over ground in degrees with 0 corresponding to north
heading	Ship's heading in degrees with 0 corresponding to north

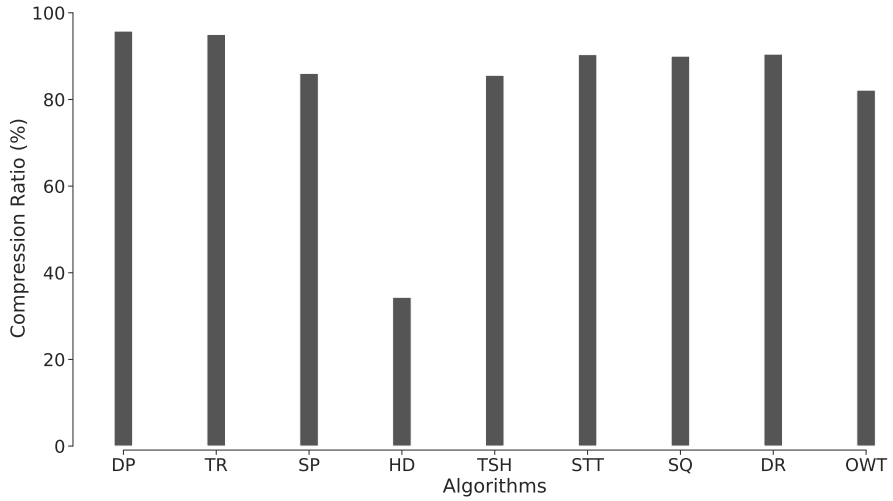
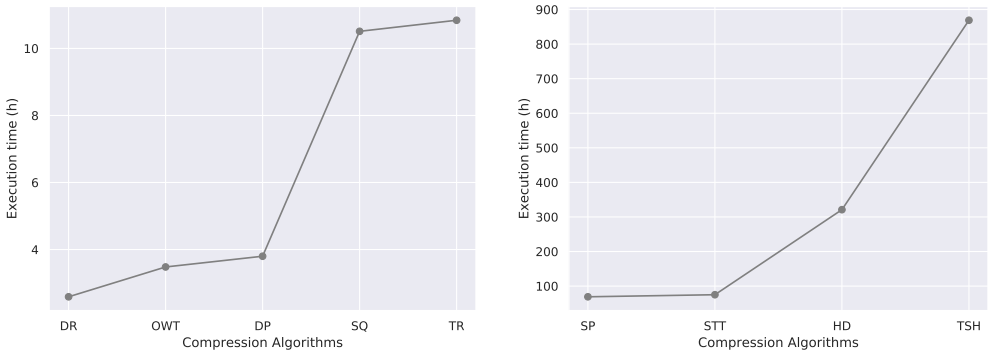


Figure 67: Compression ratio achieved by different algorithms.

about 95%. Algorithms that consider time information to decide whether or not to discard a point, tend to preserve more points in a trajectory. As DP treats a trajectory as a line in two-dimensional space and does not consider the temporal aspect, it possess the highest compression. On the other hand, opening-window algorithms SP and TSH preserve more points in the resulting set than the aforementioned algorithms, with a compression ratio of 10% less. Compression ratio is decreased in case of online algorithms STT, SQ and DR which employ SED as error criterion (DR also exploits speed), but maintained at a high level, about 90%. OWT which utilizes TRD is slightly behind by 8% from the other online algorithms with a compression of 82%. HD presents the lowest compression ratio of 34%, almost three times smaller than DP and TR. This practically means, the majority of trajectories present significant heading variations in some subsegments that exceed the average heading threshold variation of the trajectory as a whole. Thus, HD algorithm retains these "significant" points where the heading variations occur.

Besides the compression ratio, it is important to assess the execution time of each algorithm as it constitutes an important index to measure efficiency and can serve as a decision element for selecting an algorithm among others. As illustrated in Figure 68, the execution times (measured in hours) of the examined algorithms are presented in two groups, based on the exhibition time scales. Indeed, the first group's execution times are maintained low with a maximum of 11 hours, while in the second group the measurements range from 69 to over 800 hours. As shown in Figure 68(a), DR presents the lowest execution time among all algorithms followed by OWT and DP. The fairly low execution time of DR is justified by its low complexity. The execution times of OWT and DP are slightly higher, but the advantage of DP compared to DR is that the former reaches around 6% more compression than the latter. On the other hand, the compression ratio



(a) First group (b) Second group

Figure 68: Execution times of different groups.

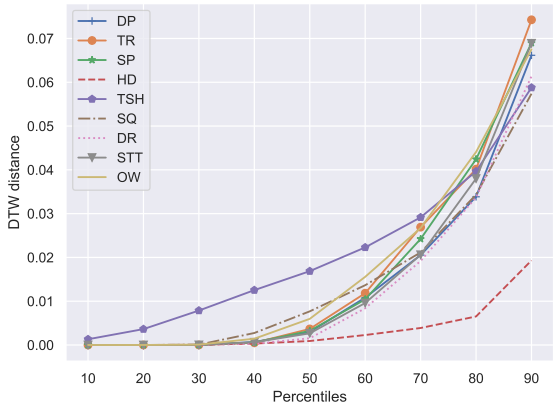
of OWT is 8% less in comparison with DR. SQ and TR spent considerable more time than the other algorithms belonging to the first group. Specifically, TR is 4 times slower than DR but reaches around 5% more compression. Additionally, when compared to OWT the compression ratio difference of TR is even higher, about 13% but accompanied with 3 times slower execution of the latter. SQ exhibits almost the same execution time with TR but presents almost the same compression ratio as DR. DP holds the highest compression ratio (96%) and is accompanied by a fairly low execution time, only 1.4 times slower than DR. Nevertheless, at this point it is worth noting that online algorithms have the advantage of executing in online mode.

In the second group, as shown in Figure 68(b), the execution times of SP and STT differ by 6 hours with the latter to present 4% more compression. The execution time of HD is 4 times larger than STT, with a compression ratio of only 34%. TSH exhibits an extremely high execution time due to its exponential complexity, 869 hours. This means that this algorithm is 2.7 times slower than HD and 335 times slower than DR which presents the lowest execution time. The compression ratio of TSH is 2.5 times better than HD, but lacks only 5% when compared to DR. Only STT from the online algorithms has a relatively high execution time. In fact, the execution times of DR, OWT and SQ remain low, except for the OWT which slightly increases. On the other hand, all the offline algorithms depict high execution times, extremely high in some cases (HD, TSH) except for the DP algorithm.

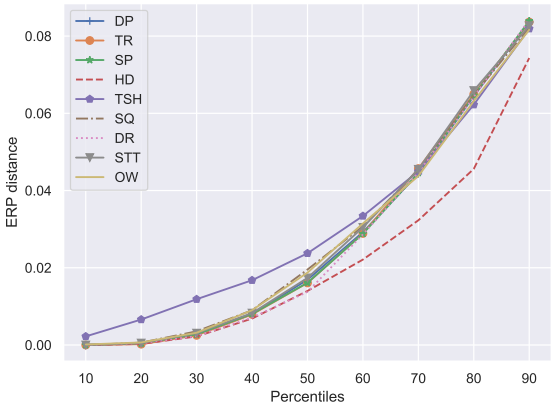
10.4.3 Distance Evaluation

To evaluate the amount of information loss that is posed by each compression algorithm, trajectory similarity measures were chosen (see Section 10.3) that are able to measure the distance between the original trajectory and the compressed one in terms of shape. The shape of the trajectory and the pattern that it forms can play an important role in the identification of illegal activities [14, 16]. Therefore, for each compression algorithm, we measured the distances D between each trajectory and its compressed counterpart and calculated the mean distance and the standard deviation of distance for the entirety of the dataset. Furthermore, the trajectory distance measures do not calculate the distance in the range of 0 to 1 and for that reason we normalized the results.

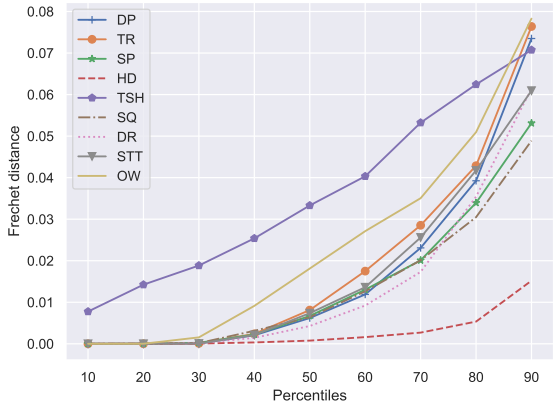
Table 27 illustrates the normalized distance results for each compression algorithm and for each trajectory distance measure. It can be seen that the HD compression algorithm presents the lowest distance in most of the cases and therefore the highest similarity with its uncompressed counterpart. Furthermore, the TSH compression algorithm which is a combination of TR, SP and HD yields the highest distance in most of the cases. This can be explained by the fact that it filters out data points based on both the speed and the heading, thus the amount of points kept is greatly reduced and the distance to the original trajectory is increased. The rest of the compression algorithms



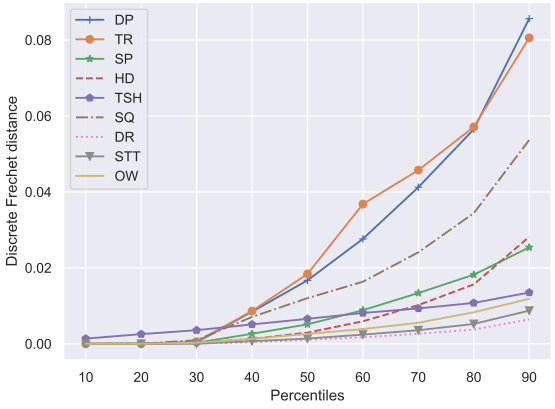
(a) DTW



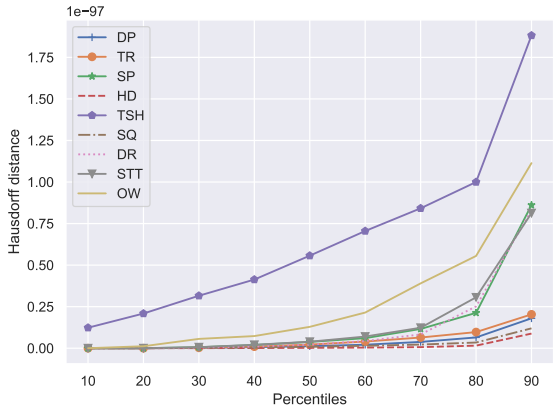
(b) ERP



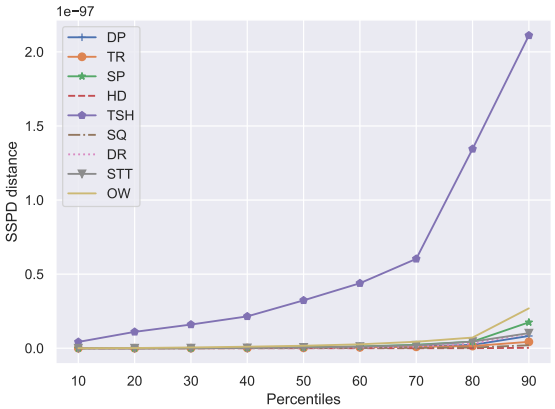
(c) Fréchet



(d) Discrete Fréchet



(e) Hausdorff



(f) SSPD

Figure 69: Distance percentiles of compression algorithms.

Compression Algorithms									
Similarity Measures		DP	TR	SP	HD	TSH	SQ	DR	STT	OWT
DTW	μ	0.0217	0.0231	0.0224	**0.0121**	0.0279	0.0222	0.021	0.0218	0.0237
	σ	0.0381	0.037	0.0377	0.0371	0.0376	0.0378	0.0385	0.0381	0.0369
ERP	μ	0.03	0.0299	0.03	**0.0268**	0.0343	0.0305	0.0294	0.0301	0.0302
	σ	0.032	0.0321	0.032	0.0348	0.0318	0.0315	0.0326	0.0319	0.0318
Frechet	μ	0.0232	0.0248	0.0203	**0.0074**	0.0382	0.0196	0.0205	0.0226	0.0283
	σ	0.0372	0.0362	0.0389	0.0332	0.027	0.0392	0.0388	0.0376	0.0335
discret Frechet	μ	0.0291	0.0299	0.0116	0.0117	0.0115	0.0221	**0.0054**	0.0055	0.0069
	σ	0.0328	0.0321	0.0423	0.0423	0.0453	0.0379	0.0436	0.0435	0.0433
Hausdorff	μ	0.0128	0.0128	0.0128	**0.0125**	0.0136	0.0128	0.0129	0.0128	0.0169
	σ	0.042	0.042	0.042	0.042	0.0447	0.042	0.0419	0.042	0.0405
SSPD	μ	0.0128	0.0128	0.0128	**0.0125**	0.0136	0.0128	0.0129	0.0128	0.0161
	σ	0.042	0.042	0.042	0.042	0.0447	0.042	0.0419	0.042	0.0408

Table 27: Distance results

present similar distances in the majority of the similarity measures. Moreover, it is worth noting that the Hausdorff distance and the SSPD yield similar results due to the similarity of these two distance metrics (see Section 10.3). These results can be further verified by the percentiles from 10 to 90 percent of each distance measure and compression algorithm that are illustrated in Figure 69. Each percentile denotes the percentage of the dataset that has a distance value less than or equal to the illustrated value. The x axis of all sub-figures of Figure 69 represents the percentile of the dataset, the y axis represents the respective distance value and each line represents a different compression algorithm. Similar to Table 27 it can be seen that the HD algorithm has the lowest distance in most of the percentiles and distance measures, while the TSH algorithm has the highest distances in most of the cases. Finally, despite the fact that all compression algorithms present an upward trend as the percentage of the dataset increases, it can be observed that all algorithms except HD present a steeper curve in most distance measures. This denotes that the HD compression algorithm is less volatile and it is an indicator that it is more suitable for datasets in the maritime domain. This is further confirmed by research conducted in [16] where the proposed trajectory classification algorithm achieves a better classification performance when the HD is used in trajectories compared to the rest of the algorithms.

10.4.4 Discussion

Table 28 presents a comparison of the different algorithms in terms of compression ratio, execution time and similarity score. The algorithms are represented in descending order regarding their performance in each aspect with the top algorithm demonstrating the best performance and the bottom algorithm representing the worst performance. The results demonstrate high compression ratios for DP, TR and DR. DR is considered the best algorithm, because it is one of the three best algorithms in all evaluating criteria. DP is also one of the best three algorithms in two of the three evaluating criteria. However, it presents the worst similarity score along with TR and TSH. The high compression ratio of TR is accompanied with a moderate execution time and a low similarity score. SQ and SP are neither in the best or worst results. They present a moderate performance in all evaluating criteria. STT presents a moderate value regarding compression ratio but achieves one of the best similarity scores. Nevertheless, the execution time is among the three worst. STT and SQ are the only two algorithms that use a memory size-based compression approach. OWT belongs to the best three algorithms regarding execution time but presents a moderate similarity score and one of the worst compression ratios. STT and OWT compete for the third place among

Position	Compression Ratio	Execution Time	Similarity Score
1^{st}	DP	DR	HD
2^{nd}	TR	OWT	DR
3^{rd}	DR	DP	STT
4^{th}	STT	SQ	SP
5^{th}	SQ	TR	SQ
6^{th}	SP	SP	OWT
7^{th}	TSH	STT	DP
8^{th}	OWT	HD	TR
9^{th}	HD	TSH	TSH

Table 28: Compression algorithms comparison

the best algorithms with STT to present slightly better results. In general terms, TSH exhibit the worst results in the sense that present the smaller accuracy for all the evaluation criteria. In fact, its execution time is extremely high in comparison with the other algorithms. HD presents the highest similarity score but on the other hand the compression ratio is significantly lower in comparison with the other algorithms and also the execution time is among the lowest.

Finally, it is worth noting that all of the compression algorithms require user defined parameters. In the set of experiments proposed in this work, the parameters were simply an average value of either speed or heading, depending on the algorithm. Despite the fact that this technique of parameter selection is suitable for compression algorithms comparison, the parameters of its compression algorithm should dynamically change for the same trajectory in real-world settings. The reason for this is that vessels exhibit different behavior such as speed and movement in different geographic areas, thus a global static parameter would yield erroneous results in large trajectories that pass through a variety of water areas. Therefore, a recommended solution for either offline or online compression algorithms is to segment the trajectory in temporal sliding windows and use a different parameter for each window.

10.5 Summary

In this work, we presented a set of well-established trajectory compression algorithms. In particular, nine compression algorithms were compared against different defined thresholds. The dynamic determination of threshold values eliminates the need of arbitrary user-defined thresholds. Instead, a threshold is applied in each trajectory, based on its actual features and its peculiarities. Furthermore, we presented the distance metrics that are suitable to best evaluate their performance and run experiments on a maritime dataset that contains AIS messages. In the experimental evaluation all compression algorithms were evaluated on three aspects: execution time, compression ratio and distance between the compressed trajectories and their uncompressed counterparts. Experiments demonstrated that the best performance of offline and online compression algorithms was achieved by DP, SP and DR, STT respectively. However, considering all assessed aspects, DR presented the most suitable performance in general. Undoubtedly, online algorithms have the advantage of executing in online mode.

The results suggest that there is a trade-off between the compression ratio and the quality achieved. Choosing a proper compression algorithm is not an easy task as the selection is application-dependent. Different application scenarios may require different trajectory characteristics preservation. Nevertheless, this research work can provide some insights of choosing and handling trajectory compression algorithms over AIS data.

Part VI
Outlook

11 Conlusions

In this chapter, we summarize the contributions of this thesis.

In Part II, we discussed the problem of real-time anomaly detection through a set of rules that can be applied over streams of AIS data to efficiently identify malicious behavior. In more detail, in Chapter 3, we presented a distributed architecture for real-time spoofing detection. We demonstrated its use in a real-world application and showed its increased performance in terms of accuracy, using a varying number of noise levels injected artificially in the stream. In chapter 4, we presented a distributed system architecture for real-time anomaly detection. The system can be employed in the AIS receivers and detect communication gaps online. We developed an algorithm that can efficiently detect vessels which switch-off their AIS transponder and demonstrated its increased performance in a real-world dataset and in real-world scenarios achieving an increased detection performance.

Recognizing the limitations of approaches that use rule-based techniques of the work proposed in Part II, in Part III, we developed approaches that model the normalcy of vessels' behavior to identify events of interest. Specifically, in Chapter 5, we proposed a clustering technique that can efficiently model the behavior of vessels using only free and openly transmitted AIS data. The modelling of the normal vessel behavior allows us to further distinguish outliers in the trajectories that are of interest to the maritime authorities. In this work, we showcased a few real world examples which our model managed to accurately detect. Identifying specific cases of anomalous behavior allows us to fine-tune, improve and exploit the proposed unsupervised technique as a basis for a supervised model for the detection of events of interest in the maritime sector. Furthermore, in Chapter 6, we presented a distributed framework that captures the vessels' behavior and extracts the patterns of their movement. Specifically, the findings of the proposed approach can be summarized in two major points: i) the resulting convex hulls manage to accurately represent the spatial boundaries of the vessel movement indicating that any future AIS position can be effectively predicted and placed in a certain geographic region ii) the convex hulls can effectively represent the behavior of the vessel movement in terms of speed and heading and detect possible anomalies and deviations.

In Part IV, we took into account the extended notion of anomaly detection which includes not only moving objects that deviate from the normal behavior, but objects whose mobility patterns can be classified into a set of predefined labels or categories. Therefore, we developed techniques that can efficiently classify vessels' trajectories in real-time. In more detail, in Chapter 7, we presented a methodology for the classification of fishing activities in a streaming fashion. The fishing behaviour was analysed which led to the selection of a specific set of features able to describe and characterise two fishing activities, trawling and longlining. Furthermore, an approach of online feature extraction and classification was presented which eliminates the need of storing streaming window batches in memory. Experimental evaluation showed the analysis of the hyperparameter tuning of our classifier along with the importance of the selected features. Moreover, the evaluation of our approach demonstrated the increased classification performance and the way the temporal size of the trajectory affects the classification accuracy. Finally, a comparison between four classifiers was made on the same set of features, demonstrating that, in our case, decision-tree based classifiers yield a better classification performance. In Chapter 8, we presented a novel approach for trajectory classification which employs a computer vision technique. The aim of our approach is to provide a high-precision classification of different vessel mobility patterns and to create a universal approach for the classification of vessel activities. Extending the work of Chapter 8, in Chapter 9, we presented a deep learning trajectory classification approach over streams of AIS messages. The goal of our methodology is to provide an efficient and alternative way to treat the problem of streaming vessel activity classification problem as an image classification task using state-of-

the-art deep learning algorithms and to create a universal approach for the classification of vessel activities. This universal approach can be achieved through the distinct visual difference that most of the mobility patterns, if not all, have in the maritime domain. Therefore, a common classifier can be used for all patterns after they have been converted to images. To demonstrate the applicability of our work in real-world conditions, two real-world datasets of AIS messages was used, one collected from a terrestrial, vessel-tracking receiver installed on the premises of Harokopio University and another one extracted from the MarineTraffic database. As deep learning approaches require a large amount of data in order to perform an accurate classification, transfer learning was employed. Experimental evaluation showed that the proposed approach achieves a state-of-the-art classification accuracy. Moreover, to handle the large amounts of data transmitted from the vessels every second, methodologies are required to address the high-velocity and voluminous streams of AIS data without impacting the overall system performance. Therefore, Preliminary experiments with compression algorithms were conducted which demonstrated that streams of AIS messages can be compressed more than 50% of the original number of messages with a relatively low impact on the classification performance. Finally, in Chapter 10, we presented a set of well-established trajectory compression algorithms. In particular, nine compression algorithms were compared against different defined thresholds. The dynamic determination of threshold values eliminates the need of arbitrary user-defined thresholds. Instead, a threshold is applied in each trajectory, based on its actual features and its peculiarities. Furthermore, we presented the distance metrics that are suitable to best evaluate their performance and run experiments on a maritime dataset that contains AIS messages. In the experimental evaluation all compression algorithms were evaluated on three aspects: execution time, compression ratio and distance between the compressed trajectories and their uncompressed counterparts. The results suggest that there is a trade-off between the compression ratio and the quality achieved. Choosing a proper compression algorithm is not an easy task as the selection is application-dependent. Different application scenarios may require different trajectory characteristics preservation. Nevertheless, this research work can provide some insights of choosing and handling trajectory compression algorithms over AIS data.

12 Ideas for Future Work

In this chapter, we present ideas and potential directions for future work.

Concerning the detection of malicious events in real-time through a set of rules, although our proposal in Chapter 3 is able to efficiently identify such events, it lacks the ability to handle more use cases of real-world noise in the streams of AIS messages. To this end, in the future, we plan to incorporate more rules in the system to extend its use cases. In Chapter 4, a first example of an AIS gap detection algorithm was presented as an extension of the proposed system of Chapter 3 that lacks the ability to take into account several aspects of the vessel's behavior. As a future work, we plan on improving the aforementioned algorithm by incorporating the change of speed and heading of previous positions of the vessel and by using dynamic time thresholds based on the AIS transmission rate of the vessel. Furthermore, we plan to enhance the system architecture with better fault tolerance techniques in case of node failures, add backpressure techniques in order to be able to handle even higher event rates and test the reliability of the system in more advanced ways.

In Chapter 5, a modified DBSCAN version was presented and incorporated in an extended framework for anomaly detection in Chapter 6. Although the system is able to efficiently model the maritime traffic through a set of generated convex hulls, many early alerts and false alarms may arise. Therefore, the use of richer pattern information (direction and speed and position) as well as windowed averages of the latest vessel status might be able to distinguish true from false alarms.It is among our future steps to further examine this problem. One interesting perspective that is to be studied further regarding the maritime traffic patterns is the experimental evaluation in large sparse geographic areas such as the Atlantic and the Pacific ocean. The resulting convex hulls in such sparse areas tend to be smaller compared to the convex hulls in dense areas such as ports due to the strict movement the vessels follow and the long linear trajectories. Our next steps include the expansion of the network with more node types, such as turn-points that may represent capes, or areas of slow speed, such as canals. The integration of the framework with weather services that can provide added value to the definition of convex hulls and edges is also among the future steps. Currently, we are working on the use of the extracted maritime traffic patterns as a basis for a real-time anomaly detection and route monitoring system.

In Chapter 7, a set of features that can be extracted online and in real-time from the vessel trajectories and a classifier was presented to identify fishing activities. As a future work, we intend to extend our methodology to cover more fishing activities, by incorporating more features to our classifier. Furthermore, to support more fishing activities, more classification schemes will be investigated and tested, in terms of their classification performance. Moreover, we aim at developing a methodology which will retrain the suggested classifier online with the use of newly received data.

In Chapters 8 and 9, a methodology was presented which converts vessel trajectories into images and classifies them into a set of predefined labels. Specifically, Chapter 8 extracts a set of features from the images and uses Random Forests to classify the images and Chapter 9 uses deep learning schemes to identify the labels of the images in real-time. As an extenstion of the methodology, it is within our future plans, to collect more images to further improve the performance of the deep neural networks and identify more vessel mobility patterns to enrich our deep learning models with more labels.

Συνοπτική παρουσίαση διδακτορικής διατριβής

Ανάλυση σε πραγματικό χρόνο Δεδομένων Κινητικότητας και ειδικά η ανάπτυξη Μεθοδολογιών για την αποτελεσματική ανίχνευση μη Αναμενόμενων Συμπεριφορών και την Κατηγοριοποίηση Τροχιών

Οι ανάγκες της ναυτιλιακής βιομηχανίας οδηγούν τα κίνητρα και τα σενάρια εφαρμογής πίσω από αυτήν την έρευνα, επομένως οι πάροχοι ναυτιλιακών υπηρεσιών όπως η MarineTraffic θα μπορούσαν να ωφεληθούν σημαντικά βελτιώνοντας τα υπάρχοντα προϊόντα και τις εφαρμογές τους. Υπάρχουν αρκετές εφαρμογές που προσπαθούν να αντιμετωπίσουν συγκεκριμένα προβλήματα στον ναυτιλιακό τομέα, όπως η αποθήκευση των τροχιών, αλλά αποτυγχάνουν να αποδείξουν την πρακτική τους εφαρμογή σε πραγματικές συνθήκες. Αντιθέτως, η συγκεκριμένη έρευνα αυτή δημιουργεί ένα πλαίσιο που περιέχει ένα σύνολο λύσεων που θα μπορούσαν να χρησιμοποιηθούν ως προϊόντα που ικανοποιούν τις απαιτήσεις της βιομηχανίας. Κάθε λύση αντιμετωπίζει μια διαφορετική πτυχή της ανίχνευσης μη αναμενόμενων συμπεριφορών στη ναυτιλία, αλλά όλες ικανοποιούν τις ανάγκες του κλάδου για Υπηρεσίες Βελτιωμένης Ποιότητας (Quality of Service - QoS): α) ανάλυση δεδομένων σε πραγματικό χρόνο, β) επεξεργασία μεγάλου όγκου δεδομένων τροχιάς και γ) αξιολόγηση σε πραγματικά δεδομένα που αποδεικνύουν την πρακτική τους εφαρμογή υπό πραγματικές συνθήκες. Οι λύσεις που περιέχονται σε αυτό το πλαίσιο προσφέρουν μια εναλλακτική, αλλά καινοτόμα προσέγγιση για κατηγοριοποίηση τροχιών και ομαδοποίηση, καθώς και ανίχνευση μη αναμενόμενων συμπεριφορών και συμβάντων που συμβάλλουν στα ερευνητικά πεδία της χωροχρονικής ανάλυσης και της ανάλυσης τροχιών. Επιπλέον, τα αποτελέσματα και τα ευρήματα προορίζονται τόσο για ερευνητές όσο και για επαγγελματίες στον τομέα της ευφυούς παρακολούθησης και επιτήρησης πλοίων. Για την έρευνα χρησιμοποιήθηκαν πραγματικά δεδομένα τροχιών πλοίων προερχόμενα από το αυτόματο σύστημα αναγνώρισης (Automatic Identification System - AIS), ένα σύστημα παρακολούθησης πλοίων που περιοδικά μεταδίδει πληροφορίες σχετικά με τη θέση τους. Οι κύριες συνεισφορές της διδακτορικής διατριβής και το σύνολο των λύσεων που περιέχονται σε αυτό το πλαίσιο συνοψίζονται παρακάτω.

Προτάθηκε μια μεθοδολογία που λειτουργεί σε κατανεμημένα περιβάλλοντα και σε πραγματικό χρόνο η οποία μπορεί να ανιχνεύει ψευδή μηνύματα AIS. Αυτά τα ψευδή μηνύματα προέρχονται από κακόβουλο πλήρωμα και προσπαθούν να αποκρύψουν τη θέση τους αλλάζοντας τις συντεταγμένες που μεταδίδονται. Για να μπορέσει η προτεινόμενη μεθοδολογία να λειτουργήσει σε πραγματικό χρόνο και κακτανεμημένα, χρησιμοποιήθηκε το εργαλείο Akka[57] το οποίο είναι κατάλληλο για τέτοιες εφαρμογές.

Σαν επέκταση της προηγούμενης μεθοδολογίας, αναπτύχθηκε ένας αλγόριθμος ο οποίος μπορεί να ανιχνεύει κενά στις τροχιές των πλοίων σε πραγματικό χρόνο. Τα κενά στις τροχιές μπορεί να προκύψουν είτε επειδή το πλήρωμα έκλεισε τον αναμεταδότη AIS, είτε γιατί στην περιοχή που βρίσκεται το πλοίο δεν υπάρχει κάλυψη από κεραίες AIS, είτε γιατί οι καιρικές συνθήκες εμποδίζουν τη μετάδοση. Το πλήρωμα συνήθως κλείνει τον αναμεταδότη όταν θέλει να κρύψει τη θέση του και να προβεί σε παράνομες δραστηριότητες. Η ιδιαιτερότητα του αλγορίθμου έγκειται στο γεγονός ότι αναγνωρίζει αν το κενό στην τροχιά δημιουργήθηκε επί σκοπού από το ίδιο το πλήρωμα ή όχι.

Προτάθηκε μια παραλλαγή του γνωστού αλγορίθμου ομαδοποίησης DBSCAN με σκοπό να ανιχνευθούν σημεία ενδιαφέροντος στις τροχιές των πλοίων. Στις περισσότερες φορές που χρησιμοποιείται ο DBSCAN, μόνο το χωρικό κομμάτι λαμβάνεται υπόψιν (π.χ. η απόσταση μεταξύ των θέσεων που μεταδίδονται μέσω του AIS). Αντιθέτως, η προτεινόμενη παραλλαγή εκμεταλλεύεται τόσο την ταχύτητα, όσο και την κατεύθυνση των πλοίων για να ανιχνεύσει ομάδες με παρόμοια συμπεριφορά.

Με βάση την παραλλαγή του DBSCAN, προτάθηκε μια μεθοδολογία ανίχνευσης μη αναμενόμενων συμπεριφορών. Μέσω αυτής της μεθοδολογίας, ναυτιλιακά μοτίβα κίνησης και διαδρομές

[57] https://akka.io/

δημιουργούνται τα οποία μπορούν να χρησιμοποιηθούν για την ανίχνευση πλοίων που αποκλίνουν από αυτά τα προκαθορισμένα μοτίβα σε πραγματικό χρόνο.

Ένα σύνολο από χαρακτηρικά (features) που μπορούν να εξαχθούν σε παργματικό χρόνο υλοποιήθηκε. Αυτά τα χαρακτηριστικά είναι κατάλληλα για την κατηγοριοποίηση τροχιών αλιευτικών δραστηριοτήτων. Μέσω αυτών των χαρακτηριστικών και της εκάστοτε δραστηριότητας που ανιχνεύθηκε, οι αρμόδιες αρχές μπορούν να αναγνωρίσουν όταν ένα πλοίο εμπλέκεται σε παράνομη αλιεία.

Χρησιμοπιήθηκε μια προσέγγιση υπολογιστικής όρασης (computer vision) για κατηγοριοποίηση τροχιών η οποία μετατρέπει τις τροχιές προερχόμενες από AIS σε εικόνες. Οι εικόνες προέρχονται από τροχιές που εμπλέκονται σε συγκεκριμένες δραστηριότητες, όπως το ψάρεμα. Αυτή η προσέγγιση χρησιμοποιεί ένα σύνολο από καθιερωμένα χαρακτηριστικά στον τομένα της υπολογιστικής όρασης εξαγόμενα από τις παραγόμενες εικόνες τα οποία μπορούν να αξιοποιηθούν από γνωστούς αλγορίθμους για κατηγοριοποίηση δεδομένων. Ως αποτέλεσμα, μπορεί να ανιχνευθεί η εκάστοτε δραστηριότητα του πλοίου.

Σαν επέκταση της προσέγγισης με τις εικόνες, μια μεθοδολογία με νευρωνικά (deep learning) αναπτύχθηκε η οποία κατηγοριοποιεί τις εικόνες τροχιών σε πραγματικό χρόνο. Η μεθοδολογία αυτή χρησιμοποιεί καθιερωμένα στον τομέα υπολογιστικής όρασης νευρωνικά δίκτυα συνέλιξης για την κατηγοριοποίηση των εικόνων.

Τέλος, μια συγκριτική μελέτη εκτελέστηκε διαφόρων αλγορίθμων συμπίεσης τροχιών σε δεδομένα προερχόμενα από AIS. Οι αλγόριθμοι αυτοί χρησιμοποιούνται για να μειώσουν το μέγεθος της τροχιάς, χωρίς όμως να χαθεί σημαντική πληροφορία. Σε αυτήν τη μελέτη αλγόριθμοι συμπίεσης που λειτουργούν είτε σε πραγματικό χρόνο είτε όχι, συγκρίθηκαν έτσι ώστε να αναγνωριστούν τα πλεονεκτήματα και τα μειονεκτήματα του κάθε αλγορίθμου στα δεδομένα τροχιών.

References

[1] Alan R. Hevner, Salvatore T. March, Jinsoo Park, and Sudha Ram. Design science in information systems research. *MIS Quarterly*, 28(1):75–105, 2004.

[2] European Maritime Safety Agency (EMSA). Annual overview of marine casualties and incidents. http://www.emsa.europa.eu/fc-default-view/tagged/85-annual-overview.html, 2018. Accessed: 02-Jun-2021.

[3] UNHCR. Mediterranean death toll soars, 2016 is deadliest year yet. https://www.unhcr.org/news/latest/2016/10/580f3e684/mediterranean-death-toll-soars-2016-deadliest-year.html., 2016. Accessed: 02-Jun-2021.

[4] Leonardo Maria Millefiori, Dimitrios Zissis, Luca Cazzanti, and Gianfranco Arcieri. A distributed approach to estimating sea port operational regions from lots of AIS data. In James Joshi, George Karypis, Ling Liu, Xiaohua Hu, Ronay Ak, Yinglong Xia, Weijia Xu, Aki-Hiro Sato, Sudarsan Rachuri, Lyle H. Ungar, Philip S. Yu, Rama Govindaraju, and Toyotaro Suzumura, editors, *2016 IEEE International Conference on Big Data, BigData 2016, Washington DC, USA, December 5-8, 2016*, pages 1627–1632. IEEE Computer Society, 2016.

[5] Giannis Spiliopoulos, Dimitrios Zissis, and Konstantinos Chatzikokolakis. A big data driven approach to extracting global trade patterns. In Christos Doulkeridis, George A. Vouros, Qiang Qu, and Shuhui Wang, editors, *Mobility Analytics for Spatio-Temporal and Social Data - First International Workshop, MATES 2017, Munich, Germany, September 1, 2017, Revised Selected Papers*, volume 10731 of *Lecture Notes in Computer Science*, pages 109–121. Springer, 2017.

[6] Jae-Gil Lee, Jiawei Han, Xiaolei Li, and Hector Gonzalez. *TraClass*: trajectory classification using hierarchical region-based and trajectory-based clustering. *Proc. VLDB Endow.*, 1(1):1081–1094, 2008.

[7] Rikard Laxhammar. Anomaly detection for sea surveillance. In *11th International Conference on Information Fusion, FUSION 2008, Cologne, Germany, June 30 - July 3, 2008*, pages 1–8. IEEE, 2008.

[8] Herbert A. Simon. The sciences of the artificial. *MIT press*, 1996.

[9] Ioannis Kontopoulos, Giannis Spiliopoulos, Dimitrios Zissis, Konstantinos Chatzikokolakis, and Alexander Artikis. Countering real-time stream poisoning: An architecture for detecting vessel spoofing in streams of AIS data. In *2018 IEEE 16th Intl Conf on Dependable, Autonomic and Secure Computing, 16th Intl Conf on Pervasive Intelligence and Computing, 4th Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress, DASC/PiCom/DataCom/CyberSciTech 2018, Athens, Greece, August 12-15, 2018*, pages 981–986. IEEE Computer Society, 2018.

[10] Konstantinos Chatzikokolakis, Dimitris Zissis, Marios Vodas, Giannis Spiliopoulos, and Ioannis Kontopoulos. A distributed lightning fast maritime anomaly detection service. In *OCEANS 2019 - Marseille*, pages 1–8. IEEE, 06 2019.

[11] Ioannis Kontopoulos, Konstantinos Chatzikokolakis, Dimitris Zissis, Konstantinos Tserpes, and Giannis Spiliopoulos. Real-time maritime anomaly detection: detecting intentional AIS switch-off. *Int. J. Big Data Intell.*, 7(2):85–96, 2020.

[12] Ioannis Kontopoulos, Iraklis Varlamis, and Konstantinos Tserpes. Uncovering hidden concepts from AIS data: A network abstraction of maritime traffic for anomaly detection. In Konstantinos Tserpes, Chiara Renso, and Stan Matwin, editors, *Multiple-Aspect Analysis of Semantic Trajectories - First International Workshop, MASTER 2019, Held in Conjunction with ECML-PKDD 2019, Würzburg, Germany, September 16, 2019, Proceedings*, volume 11889 of *Lecture Notes in Computer Science*, pages 6–20. Springer, 2019.

[13] Ioannis Kontopoulos, Iraklis Varlamis, and Konstantinos Tserpes. A distributed framework for extracting maritime traffic patterns. *Int. J. Geogr. Inf. Sci.*, 35(4):767–792, 2021.

[14] Ioannis Kontopoulos, Konstantinos Chatzikokolakis, Konstantinos Tserpes, and Dimitris Zissis. Classification of vessel activity in streaming data. In Julien Gascon-Samson, Kaiwen Zhang, Khuzaima Daudjee, and Bettina Kemme, editors, *DEBS '20: The 14th ACM International Conference on Distributed and Event-based Systems, Montreal, Quebec, Canada, July 13-17, 2020*, pages 153–164. ACM, 2020.

[15] Ioannis Kontopoulos, Antonios Makris, Dimitris Zissis, and Konstantinos Tserpes. A computer vision approach for trajectory classification. In *22nd IEEE International Conference on Mobile Data Management, MDM 2021, Toronto, ON, Canada, June 15-18, 2021*, pages 163–168. IEEE, 2021.

[16] Ioannis Kontopoulos, Antonios Makris, and Konstantinos Tserpes. A deep learning streaming methodology for trajectory classification. *ISPRS International Journal of GeoInformation*, 10(4):250, 2021.

[17] Antonios Makris, Ioannis Kontopoulos, Panagiotis Alimisis, and Konstantinos Tserpes. A comparison of trajectory compression algorithms over AIS data. *IEEE Access*, 9:92516–92530, 2021.

[18] C. Hewitt. Actor model of computation: Scalable robust information systems. *arXiv: Programming Languages*, 2010.

[19] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM Comput. Surv.*, 41(3):15:1–15:58, 2009.

[20] Ming Chen, Yuhua Li, Zhifeng Zhang, Ching-Hsien Hsu, and Shangguang Wang. Real-time, large-scale duplicate image detection method based on multi-feature fusion. *J. Real Time Image Process.*, 13(3):557–570, 2017.

[21] Wenzhong Guo, Jie Li, Guolong Chen, Yuzhen Niu, and Chengyu Chen. A pso-optimized real-time fault-tolerant task allocation algorithm in wireless sensor networks. *IEEE Trans. Parallel Distributed Syst.*, 26(12):3236–3249, 2015.

[22] Kevin Lee and Kiel Gilleade. Generic processing of real-time physiological data in the cloud. *Int. J. Big Data Intell.*, 3(4):215–227, 2016.

[23] Anders Holst and Jan Ekman. Anomaly detection in vessel motion. *Internal report Saab Systems*, 2003.

[24] Anders Holst, Björn Bjurling, Jan Ekman, Åsa Rudström, Klas Wallenius, Mattias Bjorkman, Farzad Fooladvandi, Rikard Laxhammar, and Johan Tronninger. A joint statistical and symbolic anomaly detection system: Increasing performance in maritime surveillance. In *15th International Conference on Information Fusion, FUSION 2012, Singapore, July 9-12, 2012*, pages 1919–1926. IEEE, 2012.

[25] Iraklis Varlamis, Konstantinos Tserpes, and Christos Sardianos. Detecting search and rescue missions from AIS data. In *34th IEEE International Conference on Data Engineering Workshops, ICDE Workshops 2018, Paris, France, April 16-20, 2018*, pages 60–65. IEEE Computer Society, 2018.

[26] Konstantinos Chatzikokolakis, Dimitrios Zissis, Giannis Spiliopoulos, and Konstantinos Tserpes. Mining vessel trajectory data for patterns of search and rescue. In Nikolaus Augsten, editor, *Proceedings of the Workshops of the EDBT/ICDT 2018 Joint Conference (EDBT/ICDT 2018), Vienna, Austria, March 26, 2018*, volume 2083 of *CEUR Workshop Proceedings*, pages 117–124. CEUR-WS.org, 2018.

[27] Marco Balduzzi. Ais exposed understanding vulnerabilities and attacks 2.0. https://www.blackhat.com/docs/asia-14/materials/Balduzzi/Asia-14-Balduzzi-AIS-Exposed-Understanding-Vulnerabilities-And-Attacks.pdf, 2014. Accessed: 07-Jun-2021.

[28] Cyril Ray, Clément Iphar, and Aldo Napoli. Methodology for real-time detection of ais falsification. In *Maritime Knowledge Discovery and Anomaly Detection Workshop*, 07 2016.

[29] PAPI Francesco, TARCHI Dario, VESPE Michele, OLIVERI Franco, BORGHESE Francesco, and AULICINO Giuseppeand VOLLERO Antonio. Radiolocation and tracking of automatic identification system signals for maritime situational awareness. *Publications Office of the European Union*, 2015.

[30] Fotios Katsilieris, Paolo Braca, and Stefano Coraluppi. Detection of malicious AIS position spoofing by exploiting radar information. In *Proceedings of the 16th International Conference on Information Fusion, FUSION 2013, Istanbul, Turkey, July 9-12, 2013*, pages 1196–1203. IEEE, 2013.

[31] Rikard Laxhammar, Göran Falkman, and Egils Sviestins. Anomaly detection in sea traffic - A comparison of the gaussian mixture model and the kernel density estimator. In *12th International Conference on Information Fusion, FUSION '09, Seattle, Washington, USA, July 6-9, 2009*, pages 756–763. IEEE, 2009.

[32] Dimitrios Zissis. Intelligent security on the edge of the cloud. In *International Conference on Engineering, Technology and Innovation, ICE/ITMC 2017, Madeira Island, Portugal, June 27-29, 2017*, pages 1066–1070. IEEE, 2017.

[33] Loïc Salmon and Cyril Ray. Design principles of a stream-based framework for mobility analysis. *GeoInformatica*, 21(2):237–261, 2017.

[34] Marco Guerriero, Peter Willett, Stefano Coraluppi, and Craig Carthel. Radar/ais data fusion and SAR tasking for maritime surveillance. In *11th International Conference on Information Fusion, FUSION 2008, Cologne, Germany, June 30 - July 3, 2008*, pages 1–5. IEEE, 2008.

[35] Marco Guerriero, Stefano Coraluppi, Craig Carthel, and Peter Willett. Analysis of AIS intermittency and vessel characterization using a hidden markov model. In Klaus-Peter Fähnrich and Bogdan Franczyk, editors, *40. Jahrestagung der Gesellschaft für Informatik, Service Science - Neue Perspektiven für die Informatik, INFORMATIK 2010, Leipzig, Germany, September 27 - October 1, 2010, Band 2*, volume P-176 of *LNI*. GI, 2010.

[36] Fabio Mazzarella, Michele Vespe, Alfredo Alessandrini, Dario Tarchi, Giuseppe Aulicino, and Antonio Vollero. A novel anomaly detection approach to identify intentional AIS on-off switching. *Expert Syst. Appl.*, 78:110–123, 2017.

[37] Kostas Patroumpas, Elias Alevizos, Alexander Artikis, Marios Vodas, Nikos Pelekis, and Yannis Theodoridis. Online event recognition from moving vessel trajectories. *GeoInformatica*, 21(2):389–427, 2017.

[38] Peter Yap. Grid-based path-finding. In *Proceedings of the 15th Conference of the Canadian Society for Computational Studies of Intelligence on Advances in Artificial Intelligence*, AI '02, pages 44–55, London, UK, UK, 2002. Springer-Verlag.

[39] mR-V: Line Simplification through Mnemonic Rasterization. https://www.researchgate.net/publication/316748027_mR-V_Line_Simplification_through_Mnemonic_Rasterization. Accessed: 08-Jun-2021.

[40] Virginia Fernandez Arguedas, Giuliana Pallotta, and Michele Vespe. Maritime traffic networks: From historical positioning data to unsupervised maritime traffic monitoring. *IEEE Trans. Intell. Transp. Syst.*, 19(3):722–732, 2018.

[41] Pasquale Coscia, Paolo Braca, Leonardo Maria Millefiori, Francesco A. N. Palmieri, and Peter Willett. Multiple ornstein-uhlenbeck processes for maritime traffic graph representation. *IEEE Trans. Aerosp. Electron. Syst.*, 54(5):2158–2170, 2018.

[42] David H Douglas and Thomas K Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: the international journal for geographic information and geovisualization*, 10(2):112–122, 1973.

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

[44] Omnia Ossama, Hoda M.O. Mokhtar, and Mohamed E. El-Sharkawi. An extended k-means technique for clustering moving objects. *Egyptian Informatics Journal*, 12(1):45 – 51, 2011.

[45] Wenting Liu, Zhijian Wang, and Jun Feng. Continuous clustering of moving objects in spatial networks. In Ignac Lovrek, Robert J. Howlett, and Lakhmi C. Jain, editors, *Knowledge-Based Intelligent Information and Engineering Systems*, pages 543–550, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[46] Jidong Chen and Xiaofeng Meng. Clustering analysis of moving objects. In *Moving Objects Management*, pages 151–171. Springer, 2010.

[47] Bo Liu, Erico N. de Souza, Stan Matwin, and Marcin Sydow. Knowledge-based clustering of ship trajectories using density-based approach. In Jimmy J. Lin, Jian Pei, Xiaohua Hu, Wo Chang, Raghunath Nambiar, Charu C. Aggarwal, Nick Cercone, Vasant G. Honavar, Jun Huan, Bamshad Mobasher, and Saumyadipta Pyne, editors, *2014 IEEE International Conference on Big Data, Big Data 2014, Washington, DC, USA, October 27-30, 2014*, pages 603–608. IEEE Computer Society, 2014.

[48] Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. Trajectory clustering: a partition-and-group framework. In Chee Yong Chan, Beng Chin Ooi, and Aoying Zhou, editors, *Proceedings of the ACM SIGMOD International Conference on Management of Data, Beijing, China, June 12-14, 2007*, pages 593–604. ACM, 2007.

[49] Yifan Li, Jiawei Han, and Jiong Yang. Clustering moving objects. In Won Kim, Ron Kohavi, Johannes Gehrke, and William DuMouchel, editors, *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Seattle, Washington, USA, August 22-25, 2004*, pages 617–622. ACM, 2004.

[50] Daniel Gyllstrom, Eugene Wu, Hee-Jin Chae, Yanlei Diao, Patrick Stahlberg, and Gordon Anderson. Sase: Complex event processing over streams. *CoRR*, abs/cs/0612128, 12 2006.

[51] Adrian Paschke and Alexander Kozlenkov. Rule-based event processing and reaction rules. In Guido Governatori, John Hall, and Adrian Paschke, editors, *Rule Interchange and Applications*, pages 53–66, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[52] Gianpaolo Cugola and Alessandro Margara. TESLA: a formally defined event specification language. In Jean Bacon, Peter R. Pietzuch, Joe Sventek, and Ugur Çetintemel, editors, *Proceedings of the Fourth ACM International Conference on Distributed Event-Based Systems, DEBS 2010, Cambridge, United Kingdom, July 12-15, 2010*, pages 50–61. ACM, 2010.

[53] Gianpaolo Cugola and Alessandro Margara. Processing flows of information: From data stream to complex event processing. *ACM Computing Surveys*, 44, 06 2012.

[54] Alexander Artikis, Marek Sergot, and Georgios Paliouras. An event calculus for event recognition. *IEEE Transactions on Knowledge and Data Engineering*, 27:895–908, 04 2015.

[55] Maria Riveiro, Göran Falkman, Tom Ziemke, and Håkan Warston. Visad: An interactive and visual analytical tool for the detection of behavioral anomalies in maritime traffic data. *Proc SPIE*, 7346, 05 2009.

[56] Manolis Pitsikalis, Alexander Artikis, Richard Dreo, Cyril Ray, Elena Camossi, and Anne-Laure Jousselme. Composite event recognition for maritime monitoring. In *Proceedings of the 13th ACM International Conference on Distributed and Event-Based Systems*, DEBS '19, page 163–174, New York, NY, USA, 2019. Association for Computing Machinery.

[57] Manolis Tsogas, Polyzois Parthymos, Marios Moutzouris, Nektarios Patlakas, George Karagiannis, Antonis Kostaridis, and Dimitris Diagourtas. A geospatial complex event processing engine for abnormal vessel behavior detection suitable for maritime surveillance. In *1st Maritime Situational Awareness Workshop MSAW2019*, 10 2019.

[58] Manolis Pitsikalis, Ioannis Kontopoulos, Alexander Artikis, Elias Alevizos, Paul Delaunay, Jules-Edouard Pouessel, Richard Dreo, Cyril Ray, Elena Camossi, Anne-Laure Jousselme, and et al. Composite event patterns for maritime monitoring. In *Proceedings of the 10th Hellenic Conference on Artificial Intelligence*, SETN '18, New York, NY, USA, 2018. Association for Computing Machinery.

[59] Juan Boubeta-Puig, Inmaculada Medina-Bulo, Guadalupe Ortiz, and Germán Fuentes-Landi. Complex event processing applied to early maritime threat detection. In *Proceedings of the 2nd International Workshop on Adaptive Services for the Future Internet and 6th International Workshop on Web APIs and Service Mashups*, WAS4FI-Mashups '12, page 1–4, New York, NY, USA, 2012. Association for Computing Machinery.

[60] Emily Walker and Nicolas Bez. A pioneer validation of a state-space model of vessel trajectories (vms) with observers' data. *Ecological Modelling*, 221:2008–2017, 08 2010.

[61] Youen Vermard, Etienne Rivot, Stéphanie Mahévas, Paul Marchal, and Didier Gascuel. Identifying fishing trip behaviour and estimating fishing effort from vms data using bayesian hidden markov models. *Ecological Modelling*, 221, 02 2010.

[62] Nicolas Bez, Emily Walker, Daniel Gaertner, Jacques Rivoirard, and Philippe Gaspar. Fishing activity of tuna purse seiners estimated from vessel monitoring system (vms) data. *Canadian Journal of Fisheries and Aquatic Sciences*, 68:1998–2010, 10 2011.

[63] Haiguang Huang, Feng Hong, Jing Liu, Chao Liu, Yuan Feng, and Zhongwen Guo. Fvid: Fishing vessel type identification based on vms trajectories. *Journal of Ocean University of China*, 18:403–412, 04 2019.

[64] Fabio Mazzarella, Michele Vespe, Dimitrios Damalas, and Giacomo Osio. Discovering vessel activities at sea using AIS data: Mapping of fishing footprints. In *17th International Conference on Information Fusion, FUSION 2014, Salamanca, Spain, July 7-10, 2014*, pages 1–7. IEEE, 2014.

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

[66] Jose Antonio M. R. Rocha, Valeria C. Times, Gabriel Oliveira, Luis O. Alvares, and Vania Bogorny. Db-smot: A direction-based spatio-temporal clustering method. In *2010 5th IEEE International Conference Intelligent Systems*, page 114–119. IEEE, 2010.

[67] Erico Souza, Kristina Boerder, and Boris Worm. Improving fishing pattern detection from satellite ais using data mining and machine learning. *PLOS ONE*, 11:e0158248, 07 2016.

[68] Xiang Jiang, Daniel Silver, Baifan Hu, and Erico Souza. Fishing activity detection from ais data using autoencoders. In *Proceedings of the 29th Canadian Conference on Artificial Intelligence on Advances in Artificial Intelligence*, pages 33–39. Springer, 05 2016.

[69] Buncha Chuaysi and Supaporn Kiattisin. Fishing vessels behavior identification for combating iuu fishing: Enable traceability at sea. In *Wireless Personal Communications*. Springer, 02 2020.

[70] Rajkumar Saini, Partha Roy, and Debi Dogra. A segmental hmm based trajectory classification using genetic algorithm. *Expert Systems with Applications*, 93, 10 2017.

[71] Xiaogang Wang, Keng Teck Ma, Gee Wah Ng, and W. Eric L. Grimson. Trajectory analysis and semantic region modeling using a nonparametric bayesian model. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008), 24-26 June 2008, Anchorage, Alaska, USA*. IEEE Computer Society, 2008.

[72] Weiming Hu, Xi Li, Guodong Tian, Stephen J. Maybank, and Zhongfei Zhang. An incremental dpmm-based method for trajectory clustering, modeling, and retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35:1051–1065, 2013.

[73] Xiang Jiang, Erico N. de Souza, Ahmad Pesaranghader, Baifan Hu, Daniel L. Silver, and Stan Matwin. Trajectorynet: an embedded GPS trajectory representation for point-based classification using recurrent neural networks. In Marcellus Mindel, Kelly A. Lyons, and Joe Wigglesworth, editors, *Proceedings of the 27th Annual International Conference on Computer Science and Software Engineering, CASCON 2017, Markham, Ontario, Canada, November 6-8, 2017*, pages 192–200. IBM / ACM, 2017.

[74] Rui Zhang, Peng Xie, Chen Wang, Gaoyang Liu, and Shaohua Wan. Classifying transportation mode and speed from trajectory data via deep multi-scale learning. *Comput. Networks*, 162, 2019.

[75] Xiang Jiang, Xuan Liu, Erico N. de Souza, Baifan Hu, Daniel L. Silver, and Stan Matwin. Improving point-based AIS trajectory classification with partition-wise gated recurrent units. In *2017 International Joint Conference on Neural Networks, IJCNN 2017, Anchorage, AK, USA, May 14-19, 2017*, pages 4044–4051. IEEE, 2017.

[76] Rui Chen, Mingjian Chen, Wanli Li, Jianguang Wang, and Xiang Yao. Mobility modes awareness from trajectories based on clustering and a convolutional neural network. *ISPRS International Journal of Geo-Information*, 8(5):208, 2019.

[77] Xiang Chen, Achuthan Kamalasudhan, and Xinyu Zhang. An application of convolutional neural network to derive vessel movement patterns. In *Proceedings of the 5th International Conference on Transportation Information and Safety (ICTIS)*, pages 939–944. IEEE, 2019.

[78] Yuki Endo, Hiroyuki Toda, Kyosuke Nishida, and Jotaro Ikedo. Classifying spatial trajectories using representation learning. *Int. J. Data Sci. Anal.*, 2(3-4):107–117, 2016.

[79] Nirvana Meratnia and Rolf A de By. A new perspective on trajectory compression techniques. In *Proc. ISPRS Commission II and IV, WG II/5, II/6, IV/1 and IV/2 Joint Workshop Spatial, Temporal and Multi-Dimensional Data Modelling and Analysis*, 2003.

[80] Yoran E. Leichsenring and Fabiano Baldo. An evaluation of compression algorithms applied to moving object trajectories. *Int. J. Geogr. Inf. Sci.*, 34(3):539–558, 2020.

[81] Jonathan Muckell, Jeong-Hyon Hwang, Catherine T. Lawson, and S. S. Ravi. Algorithms for compressing GPS trajectory data: an empirical evaluation. In Divyakant Agrawal, Pusheng Zhang, Amr El Abbadi, and Mohamed F. Mokbel, editors, *18th ACM SIGSPATIAL International Symposium on Advances in Geographic Information Systems, ACM-GIS 2010, November 3-5, 2010, San Jose, CA, USA, Proceedings*, pages 402–405. ACM, 2010.

[82] Minjie Chen, Mantao Xu, and Pasi Franti. Compression of gps trajectories. In *2012 Data Compression Conference*, pages 62–71. IEEE, 2012.

[83] Jeremy Birnbaum, Hsiang-Cheng Meng, Jeong-Hyon Hwang, and Catherine Lawson. Similarity-based compression of gps trajectory data. In *2013 Fourth International Conference on Computing for Geospatial Research and Application*, pages 92–95. IEEE, 2013.

[84] Philippe Cudre-Mauroux, Eugene Wu, and Samuel Madden. Trajstore: An adaptive storage system for very large trajectory data sets. In *2010 IEEE 26th International Conference on Data Engineering (ICDE 2010)*, pages 109–120. IEEE, 2010.

[85] Yan Zhao, Shuo Shang, Yu Wang, Bolong Zheng, Quoc Viet Hung Nguyen, and Kai Zheng. Rest: A reference-based framework for spatio-temporal trajectory compression. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2797–2806, 2018.

[86] Jingxian Liu, Huanhuan Li, Zaili Yang, Kefeng Wu, Yi Liu, and Ryan Wen Liu. Adaptive douglas-peucker algorithm with automatic thresholding for ais-based vessel trajectory compression. *IEEE Access*, 7:150677–150692, 2019.

[87] Zhaokun Wei, Xinlian Xie, and Xiaoju Zhang. Ais trajectory simplification algorithm considering ship behaviours. *Ocean Engineering*, 216:108086, 2020.

[88] Shu-kai Zhang, Zheng-jiang Liu, Yao Cai, Zhao-lin Wu, and Guo-you Shi. Ais trajectories simplification and threshold determination. *The Journal of Navigation*, 69(4):729–744, 2016.

[89] Liangbin Zhao and Guoyou Shi. A method for simplifying ship trajectory based on improved douglas–peucker algorithm. *Ocean Engineering*, 166:37–46, 2018.

[90] Liangbin Zhao and Guoyou Shi. A trajectory clustering method based on douglas-peucker compression and density for marine traffic pattern recognition. *Ocean Engineering*, 172:456–467, 2019.

[91] Laurent Etienne, Thomas Devogele, and Alain Bouju. Spatio-temporal trajectory analysis of mobile objects following the same itinerary. *Advances in Geo-Spatial Information Science*, 10:47–57, 2012.

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

[93] Yan Li, Ryan Wen Liu, Jingxian Liu, Yu Huang, Bin Hu, and Kai Wang. Trajectory compression-guided visualization of spatio-temporal ais vessel density. In *2016 8th International Conference on Wireless Communications & Signal Processing (WCSP)*, pages 1–5. IEEE, 2016.

[94] Yu Huang, Yan Li, Zhaofeng Zhang, and Ryan Wen Liu. Gpu-accelerated compression and visualization of large-scale vessel trajectories in maritime iot industries. *IEEE Internet of Things Journal*, 7(11):10794–10812, 2020.

[95] Giannis Fikioris, Kostas Patroumpas, Alexander Artikis, Georgios Paliouras, and Manolis Pitsikalis. Fine-tuned compressed representations of vessel trajectories. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 2429–2436, 2020.

[96] Shuang Sun, Yan Chen, Zaiji Piao, and Jinsong Zhang. Vessel ais trajectory online compression based on scan-pick-move algorithm added sliding window. *IEEE Access*, 8:109350–109359, 2020.

[97] Antonios Makris, Camila Leite da Silva, Vania Bogorny, Luis Otavio Alvares, Jose Antonio Macedo, and Konstantinos Tserpes. Evaluating the effect of compressing algorithms for trajectory similarity and classification problems. *GeoInformatica*, pages 1–33, 2021.

[98] Masiar Babazadeh, Andrea Gallidabino, and Cesare Pautasso. Decentralized stream processing over web-enabled devices. In Schahram Dustdar, Frank Leymann, and Massimo Villari, editors, *Service Oriented and Cloud Computing - 4th European Conference, ESOCC 2015, Taormina, Italy, September 15-17, 2015. Proceedings*, volume 9306 of *Lecture Notes in Computer Science*, pages 3–18. Springer, 2015.

[99] Benjamin Gaunitz, Martin Roth, and Bogdan Franczyk. Dynamic and scalable real-time analytics in logistics - combining apache storm with complex event processing for enabling new business models in logistics. In Joaquim Filipe and Leszek A. Maciaszek, editors, *ENASE 2015 - Proceedings of the 10th International Conference on Evaluation of Novel Approaches to Software Engineering, Barcelona, Spain, 29-30 April, 2015*, pages 289–294. SciTePress, 2015.

[100] Fan Zhang, Junwei Cao, Samee U. Khan, Keqin Li, and Kai Hwang. A task-level adaptive mapreduce framework for real-time streaming data in healthcare applications. *Future Gener. Comput. Syst.*, 43-44:149–160, 2015.

[101] James Warren and Nathan Marz. Big data: Principles and best practices of scalable real-time data systems. https://www.manning.com/books/big-data, 2015. Accessed: 08-Jun-2021.

[102] Carlos Santamaria, Virginia Fernandez Arguedas, Pietro Argentieri, Marlene Alvarez, Harm Greidanus, and Mattia Stasolla. Sentinel-1 maritime surveillance: Testing and experiences with long-term monitoring. *Publications Office of the European Union*, 2015.

[103] The Jerusalem Post. Avoiding detection: The team tracking iran's attempt to cloak its oil exports. https://tinyurl.com/yvjsd3ft, 2018. Accessed: 07-Jun-2021.

[104] Valentin Schatz. Combating illegal fishing in the exclusive economic zone – flag state obligations in the context of the primary responsibility of the coastal state. *International Legal Tools to Combat Illegal, Unreported and Unregulated (IUU) Fishing*, 7:383–414, 11 2016.

[105] Global Fishing Watch. Going dark: When vessels turn off ais broadcasts. https://globalfishingwatch.org/data/going-dark-when-vessels-turn-off-ais-broadcasts/, 2016. Accessed: 07-Jun-2021.

[106] Sanket Chintapalli, Derek Dagit, Bobby Evans, Reza Farivar, Thomas Graves, Mark Holderbaugh, Zhuo Liu, Kyle Nusbaum, Kishorkumar Patil, Boyang Peng, and Paul Poulosky. Benchmarking streaming computation engines: Storm, flink and spark streaming. In *2016 IEEE International Parallel and Distributed Processing Symposium Workshops, IPDPS Workshops 2016, Chicago, IL, USA, May 23-27, 2016*, pages 1789–1792. IEEE Computer Society, 2016.

[107] Angelos Valsamis, Konstantinos Tserpes, Dimitrios Zissis, Dimosthenis Anagnostopoulos, and Theodora A. Varvarigou. Employing traditional machine learning algorithms for big data streams analysis: The case of object trajectory prediction. *Journal of Systems and Software*, 127:249–257, 2017.

[108] Jakub Montewka, Pentti Kujala, and Jutta Ylitalo. The quantitative assessment of marine traffic safety in the gulf of finland, on the basis of ais data. *Zeszyty Naukowe/Akademia Morska w Szczecinie*, pages 105–115, 2009.

[109] Iraklis Varlamis, Konstantinos Tserpes, Mohammad Etemad, Amílcar Soares Júnior, and Stan Matwin. A network abstraction of multi-vessel trajectory data for detecting anomalies. In Paolo Papotti, editor, *Proceedings of the Workshops of the EDBT/ICDT 2019 Joint Conference, EDBT/ICDT 2019, Lisbon, Portugal, March 26, 2019*, volume 2322 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2019.

[110] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In Evangelos Simoudis, Jiawei Han, and Usama M. Fayyad, editors, *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96), Portland, Oregon, USA*, pages 226–231. AAAI Press, 1996.

[111] Ronald L Graham and F Frances Yao. Finding the convex hull of a simple polygon. *Journal of Algorithms*, 4(4):324 – 331, 1983.

[112] Qinyou Hu, F. Cai, Chun Yang, and Chaojian Shi. An algorithm for interpolating ship motion vectors. *TransNav, the International Journal on Marine Navigation and Safety of Sea Transportation*, 8:35–40, 03 2014.

[113] Ling-zhi Sang, Alan Wall, Zhe Mao, X. Yan, and Jin Wang. A novel method for restoring the trajectory of the inland waterway ship by using ais data. *Ocean Engineering*, 110:183–194, 12 2015.

[114] Van Nguyen, Nam-Kyun Im, and Sang-min Lee. The interpolation method for the missing ais data of ship. *Journal of Navigation and Port Research*, 39:377–384, 10 2015.

[115] Haoran Du, Youan Xiao, Liyu Duan, and Song Gao. An algorithm for vessel's missing trajectory restoration based on polynomial interpolation. In *4th International Conference on Transportation Information and Safety, ICTIS 2017*, pages 825–830. IEEE, 2017.

[116] Xue Jie, Wu Chao-zhong, Chen Zhi-jun, and Chen Xiao-xuan. A novel estimation algorithm for interpolating ship motion. *4th International Conference on Transportation Information and Safety, ICTIS 2017*, pages 557–562, 2017.

[117] Jed Long. Kinematic interpolation of movement data. *International Journal of Geographical Information Science*, 30:1–15, 09 2015.

[118] Jean-Paul Berrut and Lloyd N. Trefethen. Barycentric lagrange interpolation. *Society for Industrial and Applied Mathematics*, 46(3):501 – 517, 1984.

[119] Wilhelm Werner. Polynomial interpolation: Lagrange versus newton. *Mathematics of Computation*, 43(167):205 – 217, 1984.

[120] Bengt Fornberg and Julia Zuev. The runge phenomenon and spatially variable shape parameters in rbf interpolation. *Computers and Mathematics with Applications*, 54(3):379 – 398, 2007.

[121] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, pages 226–231. AAAI Press, 1996.

[122] Dimitris Zissis, Konstantinos Chatzikokolakis, Giannis Spiliopoulos, and Marios Vodas. A distributed spatial method for modeling maritime routes. *IEEE Access*, 8:47556–47568, 2020.

[123] Gerben Klaas Dirk de Vries and Maarten van Someren. Machine learning for vessel trajectories using compression, alignments and domain knowledge. *Expert Syst. Appl.*, 39(18):13426–13439, 2012.

[124] Konstantinos Chatzikokolakis, Dimitrios Zissis, Spiliopoulos, and Tserpes. A comparison of supervised learning schemes for the detection of search and rescue (SAR) vessel patterns. *GeoInformatica*, may 2019.

[125] Fabrizio Natale, Maurizio Gibin, Alfredo Alessandrini, Michele Vespe, and Anton Paulrud. Mapping fishing effort through ais data. *PLoS ONE*, 10:e0130746, 06 2015.

[126] Konstantinos Kapadais, Iraklis Varlamis, Christos Sardianos, and Konstantinos Tserpes. A framework for the detection of search and rescue patterns using shapelet classification. *Future Internet*, 11(9):192, 2019.

[127] Nathan Marz and James Warren. Big data: Principles and best practices of scalable realtime data systems. *Manning Publications*, 2015.

[128] Paul Tixier, Jade Vacquié-Garcia, Nicolas Gasco, Guy Duhamel, and Christophe Guinet. Mitigating killer whale depredation on demersal longline fisheries by changing fishing practices. *ICES Journal of Marine Science*, 72:1610–1620, 04 2015.

[129] Oleh Bodunov, Florian Schmidt, André Martin, Andrey Brito, and Christof Fetzer. Real-time destination and eta prediction for maritime traffic. In *Proceedings of the 12th ACM International Conference on Distributed and Event-Based Systems*, DEBS '18, page 198–201, New York, NY, USA, 2018. Association for Computing Machinery.

[130] Bart Lariviere and Dirk Van den Poel. Predicting customer retention and profitability by using random forests and regression forests techniques. *Expert Systems with Applications*, 29:472–484, 08 2005.

[131] Richard Duda, Peter Hart, and David G.Stork. Pattern classification. *Wiley Interscience*, 01 2001.

[132] Donald E. Knuth. The art of computer programming, volume 2 (3rd ed.): Seminumerical algorithms. *Addison-Wesley Longman Publishing Co., Inc.*, 1997.

[133] Trevor Hastie, Robert Tibshirani, Jerome Friedman, and James Franklin. The elements of statistical learning: Data mining, inference, and prediction. *The Mathematical Intelligencer*, 27:83–85, 11 2004.

[134] Waseem Rawat and Zenghui Wang. Deep convolutional neural networks for image classification: A comprehensive review. *Neural Computation*, 29:1–98, 06 2017.

[135] Jiajun Wu, Yinan Yu, Chang Huang, and Kai Yu. Deep multiple instance learning for image classification and auto-annotation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3460–3469, 06 2015.

[136] A. Khotanzad and Y. H. Hong. Invariant image recognition by zernike moments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(5):489–497, May 1990.

[137] Zi-Quan Hong. Algebraic feature extraction of image for recognition. *Pattern Recognition*, 24(3):211 – 219, 1991.

[138] Mihael Ankerst, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: Ordering points to identify the clustering structure. *ACM Sigmod record*, 28(2):49–60, 1999.

[139] Ford Lumban Gaol. Bresenham algorithm: Implementation and analysis in raster shape. *Journal of Computers*, 8(1):69–78, 2013.

[140] A. Bosch, A. Zisserman, and Xavier Muñoz. Image classification using random forests and ferns. *IEEE 11th International Conference on Computer Vision*, 1:14–21, 01 2007.

[141] Peijun Du, Alim Samat, Björn Waske, Sicong Liu, and Zhenhong Li. Random forest and rotation forest for fully polarized sar image classification using polarimetric and spatial features. *ISPRS Journal of Photogrammetry and Remote Sensing*, 105:38 – 53, 2015.

[142] J. Xia, P. Ghamisi, N. Yokoya, and A. Iwasaki. Random forest ensembles and extended multiextinction profiles for hyperspectral image classification. *IEEE Transactions on Geoscience and Remote Sensing*, 56(1):202–216, 2018.

[143] Ming-Kuei Hu. Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, 8(2):179–187, 1962.

[144] M. Lukic, E. Tuba, and M. Tuba. Leaf recognition algorithm using support vector machine with hu moments and local binary patterns. In *2017 IEEE 15th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*, pages 000485–000490, 2017.

[145] Jin Soo Noh and Kang Hyeon Rhee. Palmprint identification algorithm using hu invariant moments and otsu binarization. In *Fourth Annual ACIS International Conference on Computer and Information Science (ICIS'05)*, pages 94–99, 2005.

[146] L. Zhang, F. Xiang, J. Pu, and Z. Zhang. Application of improved hu moments in object recognition. In *2012 IEEE International Conference on Automation and Logistics*, pages 554–558, 2012.

[147] S. Sergyan. Color histogram features based image classification in content-based image retrieval systems. In *2008 6th International Symposium on Applied Machine Intelligence and Informatics*, pages 221–224, 2008.

[148] Peizhong Liu, Jing-Ming Guo, Kosin Chamnongthai, and Heri Prasetyo. Fusion of color histogram and lbp-based features for texture image retrieval and classification. *Information Sciences*, 390:95 – 111, 2017.

[149] K. Kim, S. Park, and Y. Choi. Deciding the number of color histogram bins for vehicle color recognition. In *2008 IEEE Asia-Pacific Services Computing Conference*, pages 134–138, 2008.

[150] Ioannis Kontopoulos, Marios Vodas, Giannis Spiliopoulos, Konstantinos Tserpes, and Dimitris Zissis. Single Ground Based AIS Receiver Vessel Tracking Dataset, Apr 2020.

[151] Di Wang, Tomio Miwa, and Takayuki Morikawa. Big trajectory data mining: A survey of methods, applications, and services. *Sensors*, 20(16):4571, 2020.

[152] Camila Leite da Silva, Lucas May Petry, and Vania Bogorny. A survey and comparison of trajectory classification methods. In *8th Brazilian Conference on Intelligent Systems, BRACIS 2019, Salvador, Brazil, October 15-18, 2019*, pages 788–793. IEEE, 2019.

[153] Moti Bachar, Gal Elimelech, Itai Gat, Gil Sobol, Nicolo Rivetti, and Avigdor Gal. Venilia, on-line learning and prediction of vessel destination. In *Proceedings of the 12th ACM International Conference on Distributed and Event-Based Systems*, DEBS '18, page 209–212, New York, NY, USA, 2018. Association for Computing Machinery.

[154] Sheng-hua Zhong, Yan Liu, and Yang Liu. Bilinear deep learning for image classification. In K. Selçuk Candan, Sethuraman Panchanathan, Balakrishnan Prabhakaran, Hari Sundaram, Wu-chi Feng, and Nicu Sebe, editors, *Proceedings of the 19th International Conference on Multimedia 2011, Scottsdale, AZ, USA, November 28 - December 1, 2011*, pages 343–352. ACM, 2011.

[155] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[156] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2818–2826. IEEE Computer Society, 2016.

[157] Barret Zoph, Vijay Vasudevan, Jonathon Shlens, and Quoc V. Le. Learning transferable architectures for scalable image recognition. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 8697–8710. IEEE Computer Society, 2018.

[158] Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, pages 2261–2269. IEEE Computer Society, 2017.

[159] Pengfei Zhang and Jinsong Zhao. The obligations of an anchored vessel to avoid collision at sea. *The Journal of Navigation*, 66(3):473–477, 2013.

[160] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359, 2009.

[161] Rikiya Yamashita, Mizuho Nishio, Richard Kinh Gian Do, and Kaori Togashi. Convolutional neural networks: an overview and application in radiology. *Insights into imaging*, 9(4):611–629, 2018.

[162] Yushi Chen, Hanlu Jiang, Chunyang Li, Xiuping Jia, and Pedram Ghamisi. Deep feature extraction and classification of hyperspectral images based on convolutional neural networks. *IEEE Trans. Geosci. Remote. Sens.*, 54(10):6232–6251, 2016.

[163] Yoshua Bengio. Deep learning of representations for unsupervised and transfer learning. In Isabelle Guyon, Gideon Dror, Vincent Lemaire, Graham W. Taylor, and Daniel L. Silver, editors, *Unsupervised and Transfer Learning - Workshop held at ICML 2011, Bellevue, Washington, USA, July 2, 2011*, volume 27 of *JMLR Proceedings*, pages 17–36. JMLR.org, 2012.

[164] Jeff Donahue, Yangqing Jia, Oriol Vinyals, Judy Hoffman, Ning Zhang, Eric Tzeng, and Trevor Darrell. Decaf: A deep convolutional activation feature for generic visual recognition. In *Proceedings of the 31th International Conference on Machine Learning, ICML 2014, Beijing, China, 21-26 June 2014*, volume 32 of *JMLR Workshop and Conference Proceedings*, pages 647–655. JMLR.org, 2014.

[165] Sharada Prasanna Mohanty, David P. Hughes, and Marcel Salathé. Using deep learning for image-based plant disease detection. *CoRR*, abs/1604.03169, 2016.

[166] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Fei-Fei Li. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*, pages 248–255. IEEE Computer Society, 2009.

[167] Matthew D. Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In David J. Fleet, Tomás Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I*, volume 8689 of *Lecture Notes in Computer Science*, pages 818–833. Springer, 2014.

[168] Min Lin, Qiang Chen, and Shuicheng Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

[169] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[170] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

[171] Douglas M Hawkins. The problem of overfitting. *Journal of chemical information and computer sciences*, 44(1):1–12, 2004.

[172] Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *CoRR*, abs/1712.04621, 2017.

[173] Antonios Makris, Konstantinos Tserpes, and Dimosthenis Anagnostopoulos. A novel object placement protocol for minimizing the average response time of get operations in distributed key-value stores. In Jian-Yun Nie, Zoran Obradovic, Toyotaro Suzumura, Rumi Ghosh, Raghunath Nambiar, Chonggang Wang, Hui Zang, Ricardo Baeza-Yates, Xiaohua Hu, Jeremy Kepner, Alfredo Cuzzocrea, Jian Tang, and Masashi Toyoda, editors, *2017 IEEE International Conference on Big Data, BigData 2017, Boston, MA, USA, December 11-14, 2017*, pages 3196–3205. IEEE Computer Society, 2017.

[174] Antonios Makris, Konstantinos Tserpes, Dimosthenis Anagnostopoulos, and Jörn Altmann. Load balancing for minimizing the average response time of get operations in distributed key-value stores. In Giancarlo Fortino, MengChu Zhou, Zofia Lukszo, Athanasios V. Vasilakos, Francesco Basile, Carlos Enrique Palau, Antonio Liotta, Maria Pia Fanti, Antonio Guerrieri, and Andrea Vinci, editors, *14th IEEE International Conference on Networking, Sensing and Control, ICNSC 2017, Calabria, Italy, May 16-18, 2017*, pages 263–269. IEEE, 2017.

[175] Antonios Makris, Konstantinos Tserpes, Giannis Spiliopoulos, and Dimosthenis Anagnostopoulos. Performance evaluation of mongodb and postgresql for spatio-temporal data. In Paolo Papotti, editor, *Proceedings of the Workshops of the EDBT/ICDT 2019 Joint Conference, EDBT/ICDT 2019, Lisbon, Portugal, March 26, 2019*, volume 2322 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2019.

[176] Antonios Makris, Konstantinos Tserpes, Giannis Spiliopoulos, Dimitrios Zissis, and Dimosthenis Anagnostopoulos. Mongodb vs postgresql: a comparative study on performance aspects. *GeoInformatica*, 25(1):241–242, 2021.

[177] Antonios Makris, Konstantinos Tserpes, Dimosthenis Anagnostopoulos, Mara Nikolaidou, and José Antônio Fernandes de Macêdo. Database system comparison based on spatiotemporal functionality. In Bipin C. Desai, Dimosthenis Anagnostopoulos, Yannis Manolopoulos, and Mara Nikolaidou, editors, *Proceedings of the 23rd International Database Applications & Engineering Symposium, IDEAS 2019, Athens, Greece, June 10-12, 2019*, pages 21:1–21:7. ACM, 2019.

[178] Michalis Potamias, Kostas Patroumpas, and Timos K. Sellis. Sampling trajectory streams with spatiotemporal criteria. In *18th International Conference on Scientific and Statistical Database Management, SSDBM 2006, 3-5 July 2006, Vienna, Austria, Proceedings*, pages 275–284. IEEE Computer Society, 2006.

[179] Jonathan Muckell, Jeong-Hyon Hwang, Vikram Patil, Catherine T. Lawson, Fan Ping, and S. S. Ravi. SQUISH: an online approach for GPS trajectory compression. In Lindi Liao, editor, *Proceedings of the 2nd International Conference and Exhibition on Computing for Geospatial Research & Application, COM.Geo 2011, Washington, DC, USA, May 23-25, 2011*, ACM International Conference Proceeding Series, pages 13:1–13:8. ACM, 2011.

[180] Goce Trajcevski, Hu Cao, Peter Scheuermann, Ouri Wolfson, and Dennis Vaccaro. On-line data reduction and the quality of history in moving objects databases. In Panos K. Chrysanthis, Christian S. Jensen, Vijay Kumar, and Alexandros Labrinidis, editors, *Fifth ACM International Workshop on Data Engineering for Wireless and Mobile Access, Mobide 2006, June 25, 2006, Chicago, IL, USA, Proceedings*, pages 19–26. ACM, 2006.

[181] Elias Dritsas, Andreas Kanavos, Maria Trigka, Spyros Sioutas, and Athanasios K. Tsakalidis. Storage efficient trajectory clustering and k-nn for robust privacy preserving spatio-temporal databases. *Algorithms*, 12(12):266, 2019.

[182] Nirvana Meratnia and Rolf A. de By. Spatiotemporal compression techniques for moving point objects. In Elisa Bertino, Stavros Christodoulakis, Dimitris Plexousakis, Vassilis Christophides, Manolis Koubarakis, Klemens Böhm, and Elena Ferrari, editors, *Advances in Database Technology - EDBT 2004, 9th International Conference on Extending Database Technology, Heraklion, Crete, Greece, March 14-18, 2004, Proceedings*, volume 2992 of *Lecture Notes in Computer Science*, pages 765–782. Springer, 2004.

[183] Kennedy Chengeta. A review of local feature algorithms and deep learning approaches in facial expression recognition with tensorflow and keras. In Jesús Ariel Carrasco-Ochoa, José Francisco Martínez Trinidad, José Arturo Olvera-López, and Joaquín Salas, editors, *Pattern Recognition - 11th Mexican Conference, MCPR 2019, Querétaro, Mexico, June 26-29, 2019, Proceedings*, volume 11524 of *Lecture Notes in Computer Science*, pages 127–138. Springer, 2019.

[184] Jonathan Muckell, Paul W. Olsen, Jeong-Hyon Hwang, Catherine T. Lawson, and S. S. Ravi. Compression of trajectory data: a comprehensive evaluation and new approach. *GeoInformatica*, 18(3):435–460, 2014.

[185] Guangwen Liu, Masayuki Iwai, and Kaoru Sezaki. A method for online trajectory simplification by enclosed area metric. In *Proc. of the Sixth International Conference on Mobile Computing and Ubiquitous Networking*, 2012.

[186] Gail Langran. Issues of implementing a spatiotemporal system. *International Journal of Geographical Information Science*, 7(4):305–314, 1993.

[187] Penghui Sun, Shixiong Xia, Guan Yuan, and Daxing Li. An overview of moving object trajectory compression algorithms. *Mathematical Problems in Engineering*, 2016, 2016.

[188] Yu Zheng, Licia Capra, Ouri Wolfson, and Hai Yang. Urban computing: concepts, methodologies, and applications. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 5(3):38, 2014.

[189] John Edward Hershberger and Jack Snoeyink. Speeding up the douglas-peucker line-simplification algorithm. *University of British Columbia, Department of Computer Science Vancouver, BC*, 1992.

[190] Shu-kai Zhang, Guo-you Shi, Zheng-jiang Liu, Zhi-wei Zhao, and Zhao-lin Wu. Data-driven based automatic maritime routing from massive ais trajectories in the face of disparity. *Ocean Engineering*, 155:240–250, 2018.

[191] H Rong, AP Teixeira, and C Guedes Soares. Data mining approach to shipping route characterization and anomaly detection based on ais data. *Ocean Engineering*, 198:106936, 2020.

[192] Haozhou Wang, Han Su, Kai Zheng, Shazia Wasim Sadiq, and Xiaofang Zhou. An effectiveness study on trajectory similarity measures. In Hua Wang and Rui Zhang, editors, *Twenty-Fourth Australasian Database Conference, ADC 2013, Adelaide, Australia, February 2013*, volume 137 of *CRPIT*, pages 13–22. Australian Computer Society, 2013.

[193] Kevin Toohey and Matt Duckham. Trajectory similarity measures. *ACM SIGSPATIAL Special*, 7(1):43–50, 2015.

[194] Peter Ranacher and Katerina Tzavella. How to compare movement? a review of physical movement similarity measures in geographic information science and beyond. *Cartography and Geographic Information Science*, 41:pages 286–307, 03 2014.

[195] N. Magdy, M. A. Sakr, T. Mostafa, and K. El-Bahnasy. Review on trajectory similarity measures. In *2015 IEEE Seventh International Conference on Intelligent Computing and Information Systems (ICICIS)*, pages 613–619, 2015.

[196] Roniel S. de Sousa, Azzedine Boukerche, and Antonio A. F. Loureiro. Vehicle trajectory similarity: Models, methods, and applications. *ACM Comput. Surv.*, 53(5):94:1–94:32, 2020.

[197] Han Su, Shuncheng Liu, Bolong Zheng, Xiaofang Zhou, and Kai Zheng. A survey of trajectory distance measures and performance evaluation. *VLDB J.*, 29(1):3–32, 2020.

[198] Andre Salvaro Furtado, Luis Otavio Campos Alvares, Nikos Pelekis, Yannis Theodoridis, and Vania Bogorny. Unveiling movement uncertainty for robust trajectory similarity analysis. *International Journal of Geographical Information Science*, 32(1):140–168, 2018.

[199] Maohan Liang, Ryan Wen Liu, Shichen Li, Zhe Xiao, Xin Liu, and Feng Lu. An unsupervised learning method with convolutional auto-encoder for vessel trajectory similarity computation. *arXiv preprint arXiv:2101.03169*, 2021.

[200] Nehal Magdy, Mahmoud A Sakr, Tamer Mostafa, and Khaled El-Bahnasy. Review on trajectory similarity measures. In *2015 IEEE seventh international conference on Intelligent Computing and Information Systems (ICICIS)*, pages 613–619. IEEE, 2015.

[201] Michail Vlachos, Dimitrios Gunopulos, and George Kollios. Discovering similar multidimensional trajectories. In Rakesh Agrawal and Klaus R. Dittrich, editors, *Proceedings of the 18th International Conference on Data Engineering, San Jose, CA, USA, February 26 - March 1, 2002*, pages 673–684. IEEE Computer Society, 2002.

[202] Lei Chen and Raymond T. Ng. On the marriage of lp-norms and edit distance. In Mario A. Nascimento, M. Tamer Özsu, Donald Kossmann, Renée J. Miller, José A. Blakeley, and K. Bernhard Schiefer, editors, *(e)Proceedings of the Thirtieth International Conference on Very Large Data Bases, VLDB 2004, Toronto, Canada, August 31 - September 3 2004*, pages 792–803. Morgan Kaufmann, 2004.

[203] Donald J. Berndt and James Clifford. Using dynamic time warping to find patterns in time series. In Usama M. Fayyad and Ramasamy Uthurusamy, editors, *Knowledge Discovery in Databases: Papers from the 1994 AAAI Workshop, Seattle, Washington, USA, July 1994. Technical Report WS-94-03*, pages 359–370. AAAI Press, 1994.

[204] Helmut Alt and Michael Godau. Computing the fréchet distance between two polygonal curves. *Int. J. Comput. Geom. Appl.*, 5:75–91, 1995.

[205] Philippe C. Besse, Brendan Guillouet, Jean-Michel Loubes, and François Royer. Review and perspective for distance based trajectory clustering. *CoRR*, abs/1508.04904, 2015.