Dansk Referat

Hver dag krydser mere end 50 tusinde skibe verdenshavene med varer mellem lande. Derfor er skibsnavigation og sikkerhed et afgørende emne indenfor transport af materialer og varer over havene. FN har udpeget et særligt agentur, Den Internationale Søfartsorganisation (IMO), til at gå i spidsen for arbejdet med navigations- og sikkerhedsaspekterne ved sejlads samt forebyggelse af atmosfærisk og havforurening fra skibe. Det automatiske identifikationssystem (AIS), udviklet af IMO, er et system, hvor skibsmonterede enheder sender og modtager skibsdata til nærliggende skibe og AIS-basestationer. Dataene indeholder information såsom skibsposition, skibsspecifikation og aktuelle navigationsoplysninger. Skibs bevægelsesdata leveret af AIS har uden tvivl kickstartet digitaliseringens æra inden for skibsindustrien på grund af dens høje frekvens, informationsrige struktur og fornuftige kvalitet.

I denne artikel præsenterer vi et multifunktionel data warehouse som er designet til at fungere som et effektivt og organiseret centralt lager, der er i stand til at indeholde store mængder af historiske AIS-data. For at eksemplificere det multifunktionelle aspekt af systemet, designer og implementerer vi en full-stack applikation, der bruger vores data warehouse specifikt til batymetrisk analyse. Gennem tæt samarbejder med Geodatastyrelsen (GST) har vi påvist den potentielle anvendelse af AIS-data inden for batymetrisk analyse. Mere specifikt har vi sammenlignet et Minimum Depth Chart (MDC) genereret af vores system med en Dybdemodel leveret af GST. MDC'en blev genereret ved hjælp af 1,8 milliarder AIS-data punkter, der spændte over fem måneder i 2021, og begge modeller var baseret på data indsamlet i danske farvande. Resultaterne fra vores sammenligning viser, at vores model kan identificere områder af interesse, der kan kræve yderligere eller nye batymetriske undersøgelser. Feedback modtaget fra GST bekræfter, at vores system beviser, hvordan AIS-data kan bruges til at levere hurtige og billige analyser som kan besvare spørgsmål fra den virkelige verden vedrørende skibe på havet. Ydermere er systemet skræddersyet til at være fleksibelt for at understøtte forskellige use-cases. Dette gøres ved at gemme data omkring skibe, ture og celler som ikke direkte bruges til MDC. F.eks. kan systemet lave et heatmap ved at tælle ture der går igemmen celler i stedet for dybten.

Vores applikation kan betragtes som et proof-of-concept for hvordan man kan benytte vores data warehouse som fundament i et analyseværktøj som alt sammen er baseret på AIS data.

Preface

This paper is a continuation of the previous paper Egede et al. in which the potential of AIS data within a multi-purpose data warehouse was described and brought to attention. In this paper, we introduce new topics whilst reusing some of the findings from the previous paper. In the following table, we describe the overall change for each section in relation to the previous paper. A change is described as either: minor, medium, major, or new. A minor change describes small changes such as commas, wording, or reformulation. Medium changes describe that we have supplemented or changed values in a section. Major changes describe sections where we introduce new techniques, topics, or principles. And finally, the new change describes sections that have not been introduced before.

Section	**Change type**
1 Introduction	NEW
2 Related Work	MEDIUM
3 Analysis	
3.1 AIS Data	MEDIUM
3.2 Data Reliability	MAJOR
3.3 Data Assumptions	MINOR
3.4 Data Cleaning	MEDIUM
3.5 Trajectories	MEDIUM
3.6 Missing Draughts	MEDIUM
3.7 , 3.8	NEW
4 Data Warehouse Design	
4.1 Logical Level	MAJOR
4.2 Physical Level	MAJOR
5 Trajectory Reconstruction	MAJOR
6 Implementation	
6.1 System Overview	MAJOR
6.2 Extract, Transform, and Load	MAJOR
6.3, 6.4, 6.5, 6.6,	NEW
7 Results	
7.1, 7.2, 7.3, 7.4, 7.5	NEW
8 Conclusion	NEW
8 Future Work	MAJOR

To clarify what we have changed, we will describe the changes to each minor, medium, or major section. Related Work is medium and mostly the same with the addition of the new trustworthiness paragraph. AIS Data is medium because the core idea of explaining the AIS format is still present while there has been added some more analysis regarding the fields and their data quality. Data Reliability is major and describes how reliable the data is. We discuss the transponder types and noisy data which can contribute to an improper analysis. Data Assumptions has minor changes regarding assumptions. Data Cleaning is medium and has the same cleaning rules with some supplemented descriptive text. Trajectories is medium because the core concept of points, journeys, and trajectories are the same but they have been formally defined in this paper. Missing Draughts is medium and the analysis on null draught is more condensed than in the previous paper. Logical Level and Physical Level has major additions to the previous data warehouse design regarding fact tables and other design features. Trajectory Reconstruction is major because the basic idea is retained but the implementation has brand new code and features. System Overview and Extract, Transform, and Load are major because the data warehouse has been extended with a system consisting of an API, frontend, and more ETL processes to support the new data warehouse design features. Future Work is major as we propose more future work to be done on the system.

We would like to thank the Danish Geodata Agency for their contribution and willingness to work together closely. More specifically we would like to express our gratitude to Ove Andersen for his continuous feedback, insights, and expertise. Furthermore, we thank Anne Mette Egge Olsen for providing a map outlining the coastal areas of Denmark, which greatly helped us during our data cleaning process.

Analyzing Billions of AIS Records in a Multi-Purpose AIS Data Warehouse

Alihan Øztürk
Department of Computer Science, AAU
Aalborg, Denmark
aaztar17@student.aau.dk

Kasper Suamchiang Hvitfeldt Nielsen
Department of Computer Science, AAU
Aalborg, Denmark
kshn16@student.aau.dk

Marcus Egge Olsen
Department of Computer Science, AAU
Aalborg, Denmark
meol17@student.aau.dk

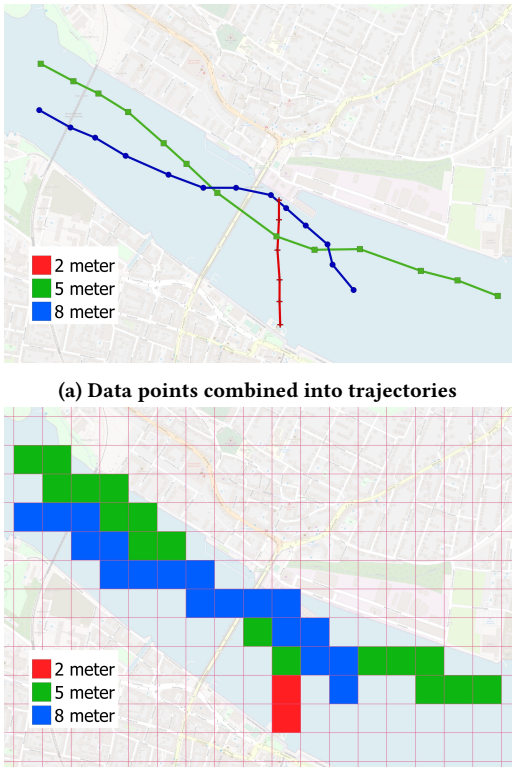
Peter Kiib Egede
Department of Computer Science, AAU
Aalborg, Denmark
pegede17@student.aau.dk

ABSTRACT

The vessel movement data provided by the Automatic Identification System (AIS) has arguably kickstarted the era of digitalization within the shipping industry due to its high frequency, information-rich structure, and reasonable quality. In this paper, we present a multi-purpose data warehouse designed to operate as an efficient and organized central repository capable of containing large amounts of historical AIS data. To exemplify the multi-purpose aspect of the system, we design and implement a full-stack application that utilizes our data warehouse specifically for bathymetric analysis. Through close collaborations with the Danish Geodata Agency (DGA), we have demonstrated the potential application of AIS data within bathymetric analysis. More specifically, we have compared a Minimum Depth Chart (MDC) generated by our system to a Depth Model provided by the DGA. The MDC was generated using 1.8 billion AIS records spanning five months in 2021, and both models were based on data gathered within Danish waters. The results from our comparison show that our model can identify areas of interest that may require additional or new bathymetric surveys. Feedback received from the DGA confirms that our system proves how AIS data can be used to provide fast and cheap analysis for real-world questions regarding ships on the sea.

KEYWORDS

Big data, AIS, Trajectory, Data Warehouse, ETL



(a) Data points combined into trajectories

(b) Raster representation of an MDC

Figure 1: Example showing part of the process when creating an MDC

1 INTRODUCTION

The seaborn trade is essential to the global trading economy. Every day more than 50 thousand ships traverse the oceans bearing goods for foreign countries [43]. Ship navigation and safety are crucial for transporting a large number of materials and goods across the oceans. The United Nations have assigned a special agency, The International Maritime Organization (IMO), to take the lead in working with the navigation and safety aspects of sailing as well as preventing atmospheric and marine pollution by ships. The IMO presents guidelines and regulations for all aspects of international shipping, i.e. ship design, construction, equipment, operation, and disposal [1]. As the IMO influences every part of the shipping process, it enables them to drive the industry towards a green economy with the end goal being a more sustainable future [8].

The Automatic Identification System (AIS), developed by the IMO, is a system where ship-mounted devices transmit and receive ship data to nearby ships and AIS base stations [13]. The data contains information such as ship position, ship specification, and current navigational information. The data is transmitted every 3rd minute or more often, depending on navigational status such as if the ship is turning or sailing at a fast speed. AIS data is plentiful because it is required on ships larger commercial ships, i.e. above 300 Gross Tonnage [13]. This combined with the fact that it is freely available in Denmark makes it widely used in trajectory

research [43]. The AIS was created for safety reasons such as avoiding ship collisions, but over time, researchers have shown that it can be used in many other areas as well [13]. The data obtained from the AIS transponders can be quite erroneous due to positional technology inaccuracy at sea, equipment failure, human errors, and areas with no reception [23, 30, 38]. Therefore, before using AIS data for any kind of analysis, it is crucial to clean or rectify erroneous data entries to ensure that future data analysis will provide a more truthful analysis.

The International Hydrographic Organisation (IHO) is responsible for bathymetry, i.e. mapping the seafloor and water depth. Their bathymetry data is freely available and the IHO makes use of crowd-sourcing the data from a range of suppliers. A more complete bathymetry chart of the world will contribute to understanding ocean circulation, tides, fishing resources, and much more [4]. Currently, the bathymetric mapping is done using expensive multibeam and single-beam sonars that are equipped on oceanographic vessels during surveys [6]. These sonars are expensive to deploy and do not cover a wide area. It is estimated that we currently have mapped less than 18% of the deep ocean floor, and approximately 50% of the world's coastal waters remain unsurveyed as of 2019 according to GEBCO [4]. AIS data can be used to contribute to IHO's depth mapping by using ship position and inferring depth out of the AIS message as seen in [21]. By utilizing draught information from AIS data, the data can be used to create a Minimum Depth Chart (MDC). A simple example of an MDC is illustrated in Figure 1 where Figure 1a shows three trajectories with different draught values, and Figure 1b illustrates the result of using the trajectories to generate an MDC. The chart is based on the maximum ship draughts recorded in an area and we, therefore, infer that the depth of an area is at least this deep. In [21], an 50m x 50m MDC was created based on five months of AIS data and resulted in a coverage of 62.5%.

The Danish Geodata Agency (DGA) is the authority in Denmark that surveys and maps all of the Danish land and sea domains [5]. Every year, 65 thousand ships pass through Danish waters, and the DGA is responsible for providing nautical charts that enable safe navigation [5]. According to the DGA, the nautical charts contain measurements that are very old and contain sparse data for areas that are not designated waterways [14]. The DGA has expressed interest in the potential of using AIS data in combination with their current analytical tools for bathymetric-related analysis.

In this paper, we extend the system from [21] by refining the trajectory reconstruction process and introducing a representation of the world as geographical cells in the data warehouse. The cells contain aggregated information about AIS data to enable more in-depth analysis in specific areas. A web-based frontend presents the cell information to the user and provides a set of filters that can be adjusted based on the user's preferences. The data warehouse serves as a foundation for many analytical use-cases but is exemplified through bathymetric analysis in this paper.

2 RELATED WORK

In this section, we review related work for creating data warehouses based on AIS data, the process of generating trajectories based on AIS data, methods for exploring bathymetry exploration based on AIS data, and finally, we look at related work in which the concept of trustworthiness is introduced.

Creating an AIS data warehouse. Tsou created a data warehouse for AIS data to apply online analysis processes to AIS data. They created fact tables based on the AIS data, with measures such as the count of trajectories, and average, minimum, and maximum speed over ground which were used for generating maps showing different information regarding the speed of the ships. In addition, a grid was imposed on the maps to attach the aforementioned trajectory information to the cells where the trajectory intersects the cells. Examples of dimensions used by Tsou to facilitate Online Analytical Processing operations are a temporal dimension, a ship dimension and an area dimension [42]. C. Tang et al. and F. Zhu also create a multidimensional data warehouse, with a focus on traffic flow and maritime surveillance respectively [41, 47]. We create a similar data warehouse design using many of the same principles as Tsou, C. Tang et al. and F. Zhu, however instead of having one kind of analysis in mind, we have a focus on creating a multi-purpose data warehouse which can be useful for many different kinds of analysis, in this paper exemplified by bathymetry related analysis.

Trajectory reconstruction. The trajectory reconstruction process aims to prepare data for subsequent analysis by utilizing anomaly removal studies, and partitioning techniques to combine AIS data into their respective trajectories [17, 19, 44]. L. Zhao et al. proposed a faster and more efficient method of AIS data compression by using an improved version of the Douglas-Peucker algorithm that can retain the characteristics of the original trajectory [46]. Y. Sun et al. introduce an Artificial Neural Network-based framework that suppresses outliers within trajectories via Hampel Filter and Butterworth Filter [40]. K. Sheng et al. have utilized a logistic regression model to construct a ship classifier. The classifier can split trajectories based on one of three proposed basic movement patterns: anchored-off, turning, or straight-sailing [39]. Similarly, Z. Yan et al. propose a model that can extract ship traffic routes based on AIS data [43]. The model split trajectories into similar types of movement patterns as [39]. Z. Yan et al. propose a multi-regime trajectory reconstruction model through a three-step process that allows trajectories to be constructed in different navigation states, namely hoteling, maneuvering, and normal-speed sailing. A characteristic that is shared among Z. Yan et al., K. Sheng et al., and L. Zhang et al. is that they all utilize the ship's sailing speed to determine the partitioning of trajectories [39, 43, 45]. More specifically K. Sheng et al. first split their trajectories into anchored-off and straight-sailing by using a speed threshold in relation to the distance traveled, and afterward the straight-sailing trajectories are split into turning.

We introduce an algorithm for trajectory reconstruction that uses advanced partitioning techniques capable of generating trajectories and splitting them based on one of two movement patterns: sailing- or stopped based on speed and time between records. It utilizes the speed over ground (SOG) values, in combination with calculated speed from the geographical and temporal values from the AIS data during reconstruction to ensure that outliers are skipped. Upon completion of the trajectory reconstruction, we utilize the Douglas-Peucker compression algorithm to greatly decrease our data sizes.

Bathymetry exploration. Seafloor mapping is essential for marine management and marine spatial planning. In combination

with the continuous change of the seafloor, we experience areas that are so outdated that the nautical data available no longer are reliable [15, 31, 37]. In an effort to minimize the costs of bathymetric surveys, E. Novaczek et al. propose a method to generate regional bathymetry and geomorphometry maps from crowd-sourced depth soundings [31]. A. Alessandrini et al. introduce a number of informative demonstrations that help us understand which kind of trends, and behaviors that exist on the sea [15]. When using AIS data to explore bathymetric features, we often use one of two approaches: a draught-based, or sailing trends-based approach. In a draught-based analysis, the draught of a ship is determined either from calculations or based on the supplied AIS data. C. O'Brian et al. presents a Draught Information System that can calculate the under-keel clearance of a ship in real-time, thus able to provide a reliable maximum draught for a ship. The value of a ship's draught can then be used to determine the minimum depth of the area that a ship has sailed [32]. In the sailing trends-based approach, areas are determined of interest if there is a shift in the sailing trends. For instance, if all ships are starting to avoid a particular area, it may indicate that the area has become very shallow, or possibly something is blocking the way. This approach is used by the Alaska Ocean Observing System (AOOS) among other things, to identify areas that require a bathymetric survey [9]. We utilize a draught-based approach to create a minimum depth chart based on the reported draughts from the AIS data.

Trustworthiness Trustworthiness describes and addresses the quality of information and its sources. It is often used to measure and quantify the quality of information coming from online resources and systems [33].

mPASS is a system designed to collect data from crowdsourcing and crowdsensing to map urban and architectural accessibility by providing reliable information coming from different data sources. The system is specifically designed for people with specific needs and is capable of providing personalized paths based on their preferences. When working with data coming from many different sources, the levels of trustworthiness varies a lot. The articles [35], [34] and [36] describe how they categorize their sources based on one of three categories: sensor data, user data, and authoritative data. This allows them to rank the categories by how reliable the data is, which is particularly important when working with crowdsourced data. In this paper, we take inspiration from the definition of trustworthiness, by adopting a pragmatic approach to define trustworthiness within our data. More specifically we introduce a function that estimates if the draught values reported by a ship are considered trustworthy.

3 ANALYSIS

In this section, we explore the characteristics of the AIS data provided by DMA. We present assumptions regarding the AIS data and analyze how we can determine the reliability. Finally, we present how to represent trajectories and geographical cells in our system.

3.1 AIS Data

AIS data is transmitted by AIS transmitters on one of two VHF frequencies. Anyone can install a transceiver and start transmitting data, and it is often used by sailors to not only receive positional details about other ships in the area but also to ensure that other ships know their position at any given time [22].

Ships of more than 300 gross tonnages are subject to the SOLAS Convention [13]. These ships are required to be equipped with Class A transceivers, which not only provide a larger broadcast range and more frequent transmissions but often also automatically read data about the ship through onboard sensors [22].

AIS data contains 26 different fields as seen in Appendix Table 5 and Appendix Table 6. The data found in Appendix Table 5 are static data such as the size of the ship, draught, and identifiers. The static data is provided by the ship crew and is sent every 6 minutes. The data found in Appendix Table 6 are dynamic data such as navigational status, latitude, longitude, and heading. The dynamic data is sent every few seconds or minutes depending on their navigational status and speed [18]. Some of the values in both the static and dynamic data are set manually by the crew and are therefore error-prone, while other values are set automatically by the equipment. Which values are set manually and automatically varies by ship, but we have found that freetext fields such as destination and ETA are of poor data quality because they are often manually reported, e.g., a ship going to Copenhagen may report *cph port*, *koebenhavn* or simply *home*.

The fields introduced within the tables, on Appendix A, are fields that are general for AIS data. Depending on the provider we see that most, if not all, of the fields are included, but typically differ in the naming convention. AIS data providers also tend to add additional fields such as `ETA_PREDICTED` or `LOCODE` which are fields that are computed and appended to the data specified in their system [2, 3].

3.2 Data Reliability

As AIS data can be transmitted by anyone with a suitable device, it is important to assess how reliable the data is, because it otherwise may contribute to incorrect conclusions. In 2017, Harati-Mokhtari et al. analyzed the reliability of AIS data and defined a number of common errors, all of which originated from human errors [24]. Harati-Mokhtari et al. conclude that AIS is rather unreliable, indicating that it is necessary to identify and remove data that are detected as error-prone, before further analysis. A description of how data cleaning processes can be utilized to improve the quality of AIS data is described in Section 3.4.

As mentioned in Section 3.1, AIS data transmitted by Class A transceivers are typical of higher quality and often read data from onboard sensors, and thus can be considered a more reliable source of information. An example in which data reliability is particularly important is when we want to explore the draught of ships' to assess the bathymetry details of certain areas. Even though only considering Class A draughts may prove to be more reliable, it is still worth considering that if many different Class B ships report a similar draught that it might also be considered reliable. Such analysis would allow us to utilize both types of transceivers and in theory, also give us a more accurate and reliable answer if done correctly. When comparing the amount of data received by Class A versus Class B we see that our data is close to equally distributed between the two. Approximately 46,3% are of type Class A and 53,6% are Class B.

Apart from determining the reliability of AIS data, another important consideration is *noise*. Noisy data is a term for data that have no contribution during analysis. Examples of such data can be corrupted data, unrealistic or gibberish values, or varying intervals within temporal data. Unfortunately, we observe noisy data in AIS and if not handled properly, it can ultimately result in incorrect conclusions or improper analysis. Specifically for AIS data provided by The Danish Maritime Authority (DMA), we identified a time period in the datasets, in which almost all static data fields are set to null values. This means that a large amount of noisy data is included every day. Upon consultation with DMA, we were made aware that the null values are a result of restarting their services at midnight and that AIS transceivers only transmit details about their static data once approximately every 6 minutes, ultimately resulting in many rows with changing dynamic data fields but without any static data associated.

Another type of noisy data which we have observed is noise in the geographical data. In Figure 2, a section of the data points from the ship with MMSI 266473000 from 26-10-2021 are shown. The ship sailed from Copenhagen to Göteborg, via a route going east of Anholt. Figure 2 shows part of this trajectory as a red line consisting of many data points very close together as well as a single red data point that does not reside on the line. The red data point highlighted by the black square is an outlier. Because a ship can't sail on a line and then for one second be many kilometers away from the line, and then the next second be back on the line again.

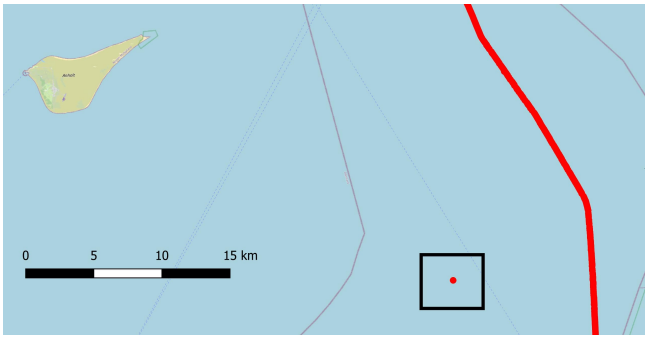


Figure 2: A segment from a ship's journey, where the red data point surrounded by a square is a geographical outlier (MMSI: 266473000, date: 26-10-2021) [21]

3.3 Data Assumptions

While examining the data we observed that in very rare cases, ships changed their MMSI during a journey. Since it is very infrequent, we assume that the MMSI is a unique identification for ships. When analyzing bathymetry details, the tide is something you often take into account. In this paper, we assume that ships only sail where they are able to sail at both high and low tide. A more detailed tide model could be considered for future work. We assume that we can get a steady stream of AIS data from everywhere to avoid handling areas with no signal reception.

To summarize, we make the following assumptions about the AIS data:

- A ship's MMSI is a unique identifier
- A ship only sails where they are able to sail at both low and high tide
- A ship sending an AIS signal always reaches a base station

3.4 Data Cleaning

To improve the overall quality of our data, a data cleaning process is applied based on some prerequisites and a set of cleaning rules. In this section, we explain and list these preliminaries.

The majority of the rules are designed to filter out general noise that exists in AIS data, while other rules function as additions that are specific to our use-cases. The use of case-specific rules is denoted in the list below with the asterisk (*) symbol.

(1) Duplicate data entries
(2) AIS data that does not follow the MMSI ship format [10]
(3) Ships with a draught above 28.5 meters [11]
(4) Ships with a length above 488 meters [11]
(5) Ships with a width above 74 meters [11]
(6) *AIS data points that are outside Danish waters
(7) *AIS data points that are on mainland Denmark

By following the cleaning rules when extracting data, we can ensure that our other data processes are not impacted by the noise that otherwise may have resulted in incorrect data.

3.5 Trajectories

When working with the geographical aspect of AIS data it can be beneficial to view the data as continuous lines instead of distinct points. This can be accomplished by performing trajectory reconstruction based on the journey of a ship. When talking about trajectories and the reconstruction of these we use the following definitions.

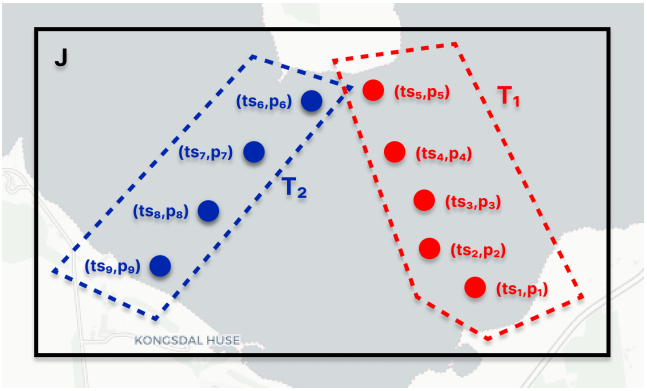


Figure 3: Example data for one ship showing the relation between points (ts_i, p_i), trajectories T_1 and T_2, and a journey J

Point: A *point* $p = (lat, lng)$ which includes *latitude lat, longitude lng*

- A point contains the spatial information

Journey: A journey $J = \{(ts_1, p_1), (ts_2, p_2), ..., (ts_n, p_n)\}$, where ts_i is the timestamp of point p_i, and where $ts_i < ts_{i+1}$.

- A journey represents the complete set of points for a ship.

Trajectory: A trajectory $T = \{(ts_1, p_1), (ts_2, p_2), ..., (ts_m, p_m)\} \subset J$ where $\forall T | T_k \cap T_l = \emptyset$ whenever $k \neq l$, where k and l are trajectory ids.

- A trajectory is a disjoint subset of a journey, determined by the sailing patterns of the ship.

An example of a journey which is split into two trajectories can be seen in Figure 3.

The trajectory reconstruction used in this paper is described in Section 5. When splitting a journey into different trajectories, we distinguish between sailing and stopped states, since having this distinction can be useful for analyzing the data. This distinction is described also described in Section 5.

3.6 Missing Draughts

When analyzing the AIS data we observed that 30.7% of the uncleaned AIS records and 24.9% of the cleaned do not include a draught value. This shows that the dirty data has a larger percentage of the data with no draught value compared to the cleaned data and that we improve the quality of the draught data with our data cleaning process.

Another observation regarding the draught data is that in many coastal areas we only have data with no draught. This is partly due to the fact that it is mostly sailing and pleasure ships that sail in the coastal areas. These ship types have a very large percentage of data points with no draught value as seen in Figure 4. The impact of this is shown in Section 7.4, where we see that missing draughts have a large impact on the ability to cover coastal areas with MDCs.

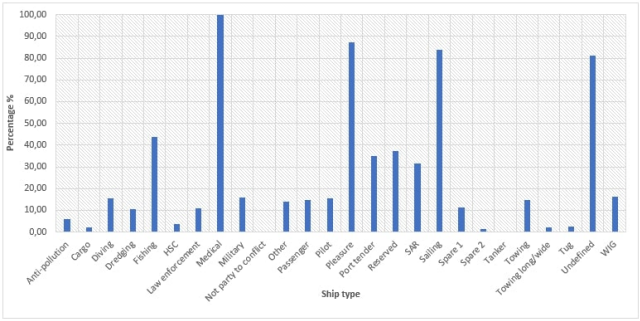


Figure 4: Bar chart showing the respective distribution of null values in draught information based on ship types [21]

3.7 Trustworthiness

Trustworthiness describes whether a trajectory is considered reliable based on a set of criteria. This evaluation is particularly important when working with AIS data because our analysis shows data in which ships have reported abnormal draught values, which our cleaning process has not filtered. We still want to include the data in our data warehouse, since these data may still be useful for analysis. Therefore, we introduce a `trust flag` that is assigned during data processing.

We however observed that this intuition is not always true with small ships, where the relation does not follow the same pattern. Because of this, we introduce a minimum draught threshold, under which we trust all values. We do this because small draughts do not contaminate our MCDs to the same degree as large draughts. As a result, we flag the trajectories that have reported draught values that are very abnormal in relation to the ship size, because the data would otherwise contaminate our MDCs.

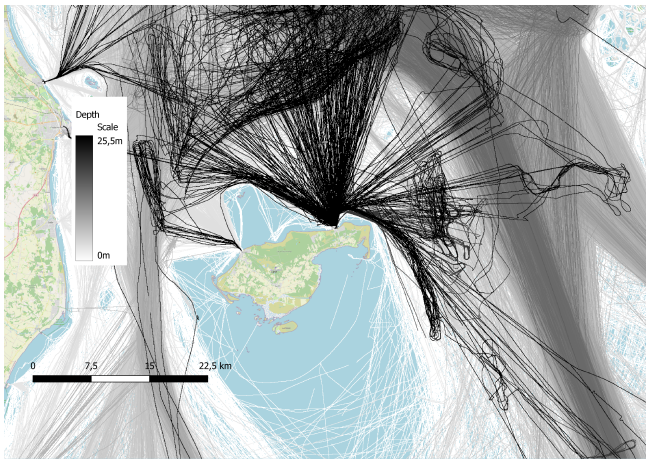


Figure 5: MDC around of Læsø, where the black area (25.5m draught) is from a single fishing ship with MMSI 220334000

Width * Length (meters^2)	Draught (meters) 0-5	5-10	10-15	15-20	20-25	25-30
0 - 1000	83.28	13.86	0.11	0.53	60.25	71.34
1000 - 2000	13.93	40.16	0.03	2.11	0.00	3.43
2000 - 3000	1.88	9.12	0.01	0.00	0.00	0.06
3000 - 4000	0.86	15.33	1.06	0.00	0.00	25.14
4000 - 5000	0.04	8.44	12.52	0.00	0.00	0.00
5000 - 6000	0.00	6.05	12.86	0.00	0.00	0.02
6000 - 7000	0.00	2.79	9.59	0.00	0.00	0.00
7000 - 8000	0.00	1.51	11.58	1.05	0.62	0.00
8000 - 9000	0.00	0.50	2.09	0.32	0.00	0.00
9000 - 10000	0.00	0.53	2.95	1.90	0.00	0.00
10000 - 11000	0.00	0.57	5.77	18.76	0.00	0.00
11000 - 12000	0.00	0.57	11.26	4.32	0.00	0.00
12000 - 13000	0.00	0.22	12.18	0.84	0.00	0.00
13000 - 14000	0.00	0.21	2.38	56.27	0.00	0.00
14000 - 15000	0.00	0.08	3.64	0.00	0.00	0.00
15000 - 16000	0.00	0.02	2.28	0.00	0.00	0.00
16000 - 17000	0.00	0.00	0.00	0.00	0.00	0.00
17000 - 18000	0.00	0.00	0.52	0.00	0.00	0.00
18000 - 19000	0.00	0.01	2.11	0.84	0.00	0.00
19000 - 20000	0.00	0.00	0.58	1.05	39.13	0.00
20000 - 21000	0.00	0.00	0.03	0.00	0.00	0.00
21000 - 22000	0.00	0.00	0.21	0.00	0.00	0.00
22000 - 23000	0.00	0.00	0.24	0.00	0.00	0.00
23000 - 24000	0.00	0.00	5.13	12.01	0.00	0.00
24000 - 25000	0.00	0.00	0.86	0.00	0.00	0.00

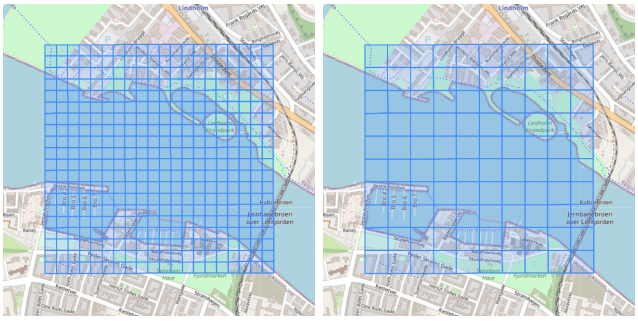
Table 1: Distribution of ship sizes (length $*$ width) per draught interval

In Equation 1 we show the equation used for assigning the flag, which we refer to as *is_draught_trusted*. The values 60 and 3 were chosen based on initial testing, as these values showed good results with regards to flagging abnormal draughts. If the formula is improved or optimized, it can be changed in the code with little effort, but it requires some reprocessing.

$$is\_draught\_trusted = \begin{cases} true, & if \ \text{draught} < 3 \\ false, & if \ \text{draught} = null \\ false, & if \ \text{width} = null \\ false, & if \ \text{length} = null \\ false, & if \ \frac{length * width}{draught} < 60 \\ true, & otherwise \end{cases} \quad (1)$$

3.8 Representing the World in Grids

When working with large amounts of data it can often be more useful to look at aggregated values instead of individual data because we can generalize patterns and draw conclusions if many are following the same patterns. To accomplish such analysis we introduce a technique to represent the world in grids. The idea is that we want to be able to create aggregated measures on a cell basis, e.g. if 1000 ships have sailed through a single cell we can accumulate the draughts reported by ships sailing through the cell and store them for future analysis. Not only does this allow us to store aggregated values separately based on specific cells, but it also allows us to faster retrieve information on areas because we can utilize the aggregated values directly instead of computing them.



(a) 50m x 50m (b) 100m x 100m

(c) 500m x 500m (d) 1000m x 1000m

Figure 6: All possible representations of our geographical cell design

On Figure 6 the idea of representing areas in grids is shown. Here we have illustrated how we can vary the granularity of our cells in case we want to be more precise. The goal is to represent the world in such grids and refer to each cell in the grid with a unique identifier.

4 DATA WAREHOUSE DESIGN

In this section, we describe the two abstraction levels of our data warehouse design, namely the logical- and physical level. The logical level describes the data warehouse's overall structure as seen in Figure 7, and the physical level describes how data is stored in the data warehouse. Furthermore, compression-, indexing- and partitioning techniques are also explained as part of the physical level.

4.1 Logical Level

The logical level describes the data warehouse's overall structure as well as the relationships between data tables. In this section, we describe the data warehouse scheme in terms of dimensions and fact tables, as well as the relationships that reside between them.

Enterprise Bus Matrix. The enterprise bus matrix provides a high-level abstraction of the data warehouse's architectural level. It gives an overview of the relationships between the dimensions and the fact tables providing further context to understand the purpose of the fact tables [25]. To retain the multi-purpose design of the data warehouse, it is desirable to be able to obtain as much information as possible. We retain the information across the different fact tables by having foreign keys to most of the dimensions in the fact tables.

	dim_junk_ais_clean	dim_navigational_status	dim_cargo_type	dim_type_of_position_fixing_device	dim_destination	dim_data_source_type	dim_type_of_mobile	dim_audit	dim_time	dim_date	dim_ship	dim_ship_type	dim_cell
(STAGING) fact_ais		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	
fact_ais_clean	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
fact_trajectory_sailing			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
fact_trajectory_stopped			✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
fact_cell_sailing							✓	✓	✓	✓		✓	✓

Table 2: Enterprise bus matrix

Dimensions. During the design of the data warehouse, it was decided that all dimensions are fixed, meaning that once a row has been inserted, it can never be changed. The decision was made for simplification purposes. In Figure 7, the dimensions are visualized and can be recognized by the `dim_` prefix in the table names, furthermore, the dimensions are also color-coordinated to represent their respective area of interest. The red dimension is a spatial dimension adding locality context by determining which geographical cell a trajectory or point is located in. The blue dimensions contain ship-specific information such as MMSI, ship type, and name. The brown dimensions contain metadata for processed records. The green dimensions describe the temporal aspect. The purple dimensions contain AIS meta information such as which positional device has been used or the data source type. The yellow dimension contains attributes that do not belong in the other dimensions or fact tables.

The relations between the tables are shown by color coding and naming conventions. The color of the foreign key corresponds to the color of the table it references. The naming of the foreign keys

is the same as the primary key in the table it references, except for the cases where a table has multiple references to the same table where another classifier has been added to the name of the foreign key.

The dim_date and dim_time dimensions store temporal information that we derive from a timestamp. None of the dimensions store information about time zones which makes it possible to represent both local and UTC time for any entry. Both tables' primary keys have been constructed as smart keys, which allows us to represent any date or time in a human-readable integer format making queries faster. For example, the 20th of June 2022 15:00 has the dim_date key value of 20220620 and dim_time key value of 150000.

The dim_destination contains entries of destinations that have been extracted from the AIS data. The destinations are reported in a free text format directly by a crew member on the ship and therefore vary widely in quality and consistency. Apart from the reported destination, the dimension also has a mapped_destination attribute which contains a cleaned version of the destination.

The dim_navigational_status dimension contains entries of the navigational states that a ship can be in at any point in time. The entries are a fixed set of AIS navigational states as described in [27].

The dim_data_source_type dimension stores information about the data source that has been used while transmitting the AIS data. Currently, it only contains AIS but may contain other sources in the future.

The dim_cargo_type dimension includes a fixed set of available AIS cargo types.

The dim_audit dimension has one entry for each load operation that has been performed on the data warehouse. The dimension includes six attributes to describe the ETL process such as processed records, load time, and ETL-version.

The dim_junk_ais_clean dimension is a junk dimension as described by Kimball [25]. The dimension contains flags for metadata from the data cleaning, such as whether an attribute has been patched.

The dim_ship dimension has three foreign keys and ten attributes that describe a specific ship. A ship is identified by a combination of its MMSI number, ship type, positional type, and mobile type. The other attributes describe general static ship information such as size, name, and callsign.

The dim_ship_type dimension contains all recorded ship types such as cargo, pleasure, or passenger. The types are defined by a fixed set of AIS ship types as described in [28].

The dim_type_of_position_fixing_device dimension contains the various types of position fixing devices that are used to determine the position of a ship. Examples are GPS, GLONASS, Galileo, and Loran-C.

The dim_type_of_mobile dimension contains the various types of AIS transponders that have been transmitted in the AIS data. For example, the class A and class B transponder types are stored within this dimension. Class A and B are outfitted on ships where class A is required by law on internationally traveling ships over 300 gross tonnages while class B is optional, and can be used for ships that do not fit the requirement [7].

The dim_cell stores grid representations of the world. This is done using the EPSG:3034 map projection. The grids are created in four levels starting from the finest cell size of 50m x 50m, and then increasing to 100m x 100m, 500m x 500m, and finally, the coarsest level being 1000m x 1000m. The hierarchy of such design is as strict as we can represent all of the coarser grids with a set of 50m cells. With a strict design, we can assure that we can represent any cell using the 50m grid as an identifier. The finest cell size of 50m x 50m was chosen because the DGA uses this granularity. The entries in the dimension are pre-loaded for a given area and enable us to easily convert between different grid sizes. Each row contains a unique 50m cell, as well as coarser cells in which it resides, making it fast to create views with a coarser granularity. The geographical dimension is described in greater detail in Section 4.2.

Fact Tables. The fact tables contain foreign keys to the dimensions, described in Section 4.1, and measures that describe the analytical information that the users can aggregate. The data warehouse design consists of five fact tables. The measures were determined in collaboration with domain experts from the DGA.

The fact_ais is the fact table containing raw AIS records loaded from the data sources during the ELT process. It is used as a staging area. Having the raw data available, means that if the data processing changes, it can always be reprocessed from its original state. The table has foreign keys to dim_navigational_status, dim_cargo_type, dim_destination, dim_time, dim_type_of_mobile, dim_date, dim_data_source_type, dim_audit, dim_ship_type, dim_ship, and dim_type_of_position_fixing_device.

The measures *coordinates*, *longitude*, and *latitude* describes the ships location. The measures *heading*, *SOG*, *COG*, *ROT* are dynamic values that describes the ship's movement. The final measure is *draught*.

The fact table fact_ais_clean contains a subset of records derived from fact_ais. The subset has been cleaned in accordance with the rules described in Section 3.4. The table has foreign keys to the same dimensions as fact_ais with the addition of the dim_cell. Furthermore, fact_ais_clean has two foreign key references to the two fact tables fact_trajectory_sailing and fact_trajectory_stopped. Only one of these foreign keys will contain a valid reference since a point cannot be both stopped and sailing. Finally fact_ais_clean has a foreign key to dim_junk_ais_clean, which contains meta data from the cleaning operation. The measures are identical to those in fact_ais.

The fact_trajectory_sailing is the trajectory fact table that stores ship trajectories satisfying our sailing definition described in Section 5. The table has foreign keys to dim_destination, dim_cargo_type, dim_date, dim_data_source_type, dim_type_of_position_fixing_device, dim_type_of_mobile, dim_audit, dim_ship, dim_ship_type, and dim_time. Additionally, this fact table has a bridge table to dim_cell. The measures are draught, coordinates, duration in minutes, the total number of points used, the sailed length, average speed, and lastly, whether the draught value is trusted. The coordinates measure is a linestring representation of the trajectory.

The fact_trajectory_stopped is the trajectory fact table where the ship's trajectories are assigned to a stopped state. The foreign keys and measures are identical to the previously mentioned table,

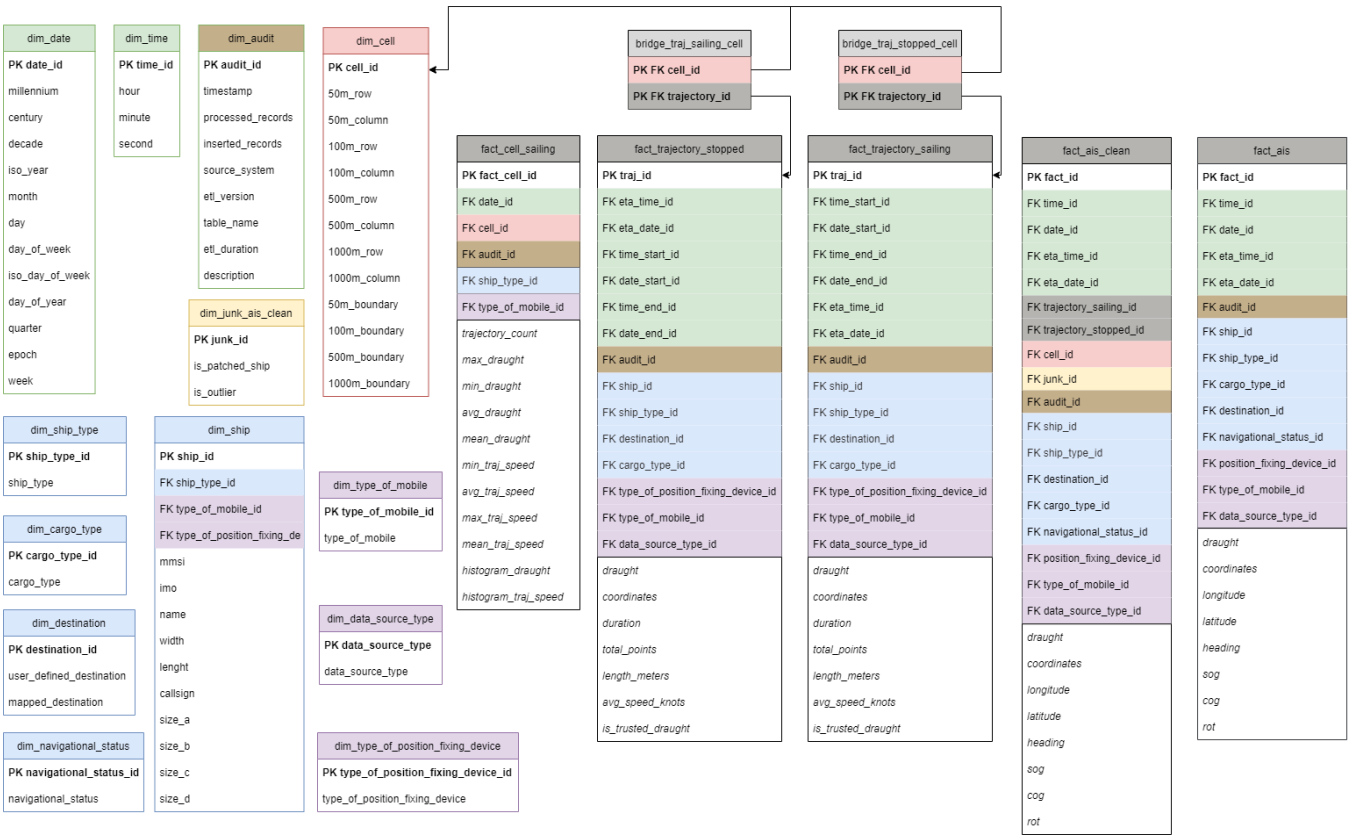


Figure 7: Full data warehouse design

`fact_trajectory_sailing`. We choose to also represent the trajectory of a stopped state as a linestring because this makes it easy to reconstruct the journey of a ship by combining the sailing and stopped linestrings.

The `fact_cell_sailing` is the fact table that will be queried by the frontend application when creating MDCs. These queries will be OLAP-based queries that enable the user to filter and refine the data to enhance their understanding of ship activity in the query area. The table contains foreign keys to `dim_cell`, `dim_date`, `dim_type_of_mobile`, `dim_ship_type`, and `dim_audit`. The measures are *trajectory_count*, that is the number of trajectories that has passed through the cell. *min_draught*, *max_draught*, *mean_draught*, and *avg_draught* covers the draught observed in the cell. *min_traj_speed*, *max_traj_speed*, *mean_traj_speed*, and *avg_traj_speed* concerns the trajectories that has passed in the cells. *histogram_draught*, *histogram_traj_speed*, and *histogram_traj_speed* are fixed size arrays where the counts of the particular unit is stored.

4.2 Physical Level

The physical design depicts how data is stored physically in the system. The design is particularly important due to the size of data warehouses and can have a large effect on query performance. In this section, we explain how indexing, partitioning, and column alignment techniques have been utilized to increase query performance.

Indexing. Indices are crucial for query performance when only a small fraction of the rows in a table need to be fetched. PostgreSQL automatically creates indices on the primary keys, and we will therefore not include these in the following. To increase performance on ETL processes, an index is created on the `ts_date_id` column in `fact_ais`, because the cleaning process only uses data from one day at a time. Therefore this index will speed up this query significantly when `fact_ais` contains data spanning many days.

To increase the speed of creating MDCs with different filters, the following indices are added:

- A btree index on the `cell_id` column in `dim_cell`
- SPGist indices on the `50m_boundary`, `100m_boundary`, `500m_boundary`, and `1000m_boundary` columns in `dim_cell`
- A btree index on the `cell_id` column in `fact_cell_sailing`
- A multicolumn btree index on the `date_id`, `ship_type_id`, `type_of_mobile_id`, and `is_draught_trusted` columns in `fact_cell_sailing`

Partitioning. Partitioning is the process of separating the data into logical segments on the physical hardware level. Partitioning of

the fact tables will be based on a range partition. A data range distribution was chosen because all requests that the users will query are based on AIS data with a specified date range. By utilizing a range partitioning, we improve the query time on the *fact_ais_clean* fact table as the DBMS does not need to do a sequential scan of the whole fact table.

An example could be that the user wants to see the ship activity in the Limfjord on the dates spanning the summer months of 2021. By having utilized partitions, the DBMS is able to filter unnecessary data and perform queries on the partitions that are relevant, based on the date range specified by the user. [26]

Column Alignment. To reduce the row size and possibly the number of IO operations, column alignment is utilized. The technique is used to order columns so that they do not cross alignment boundaries. PostgreSQL aligns data values in fixed-size values, and if a value does not match the alignment, padding is used to fill out the alignment gap. The fixed size is typically 8 bytes. For example, aligning two `int`(4 bytes) and a `double`(8 bytes) with the following order: `int,double,int` will result in 4 bytes padding on both integers resulting in three blocks of 8 bytes each making the total 24 bytes. However, aligning the values in the following order: `int,int,double` will avoid padding and the total usage will be two blocks of 8 bytes each which is a total of 16 bytes. That is a reduction of 8 bytes per row, and making small changes like this can improve the IO operations as the tables are smaller. [16] This type of column alignment has been done on all tables in the data warehouse.

Geographical Cells. As described, in Section 6.1, we wish to provide the frontend with data retrieved by the data warehouse. Due to the nature of presenting data to the frontend, we wish to handle such requests in a timely manner such that the end-user is able to make analysis efficiently. To account for the desired low response time, we have specifically designed our geographical cell tables so that they can be pre-processed and then queried in a reasonable time. This section provides details on the specifications of our tables that ultimately allow for such use-cases.

As described in Section 4.1 the geographical dimension is created using the EPSG:3034 map projection. We use a projection with meters as units instead of degrees since this allows us to easily create a 50m x 50m map which we can compare to a 50m x 50m map provided by the DGA. The dimension `dim_cell` contains conversion information between the different levels of grid granularity. Each entry stores a PostGIS *geometry* representation of the boundary of the cell as well as values for the column and row in the raster representation of the grid. For example, if we want to know which column and row that a particular cell resides on, on the 500 meter raster, we can retrieve the attributes *500m_row* and *500m_column*. By including the raster row and columns, we can simplify the calculation for mapping points and trajectories to their cells.

On Figure 6, we have visualized the difference in granularity by showcasing how we represent cells in a grid, based on the level. The level of granularity was based upon discussion with the Geodata agency and frontend limitations for viewing large amounts of data.

5 TRAJECTORY RECONSTRUCTION

In this section, we describe the algorithm used for trajectory reconstruction and techniques used to handle trajectory data. As described in Section 3.5, we perform trajectory reconstructing on the AIS records, since this enables us to view the data as continuous lines instead of distinct points.

While analyzing the trajectory data, it is useful to distinguish whether the trajectory is in a sailing or a stopped state. Such information on states is useful depending on the type of analysis. When reconstructing the trajectories we, therefore, split the trajectories based on whether they are in one of the two states.

Egede et al. [21] also use the distinction between sailing and stopped states, which is visualized in Figure 8. We expand on the definition presented by defining the sailing and stopped trajectory states as follows.

- A stopped trajectory is defined as a series of consecutive points where the ship has had a speed over ground (SOG) of less than **X** knots in **Y** minutes.
- A sailing trajectory is defined as the points that lie between two stopped trajectories.
- A trajectory consists of more than **P** points.
- A sailing trajectory is split if more than **Z** minutes have elapsed between two consecutive points.
- A point is an outlier if it has a speed of over **W** knots, and thus will not be included in any trajectory.

X, **Y**, **Z**, **W**, and **P** are parameters that can be fine-tuned based on the speed and broadcast intervals of the ships in the datasets used. It is trivial to expand this definition to have different **X**, **Y**, **Z**, **W**, and **P** values per ship type or other factors.

The datasets used in this paper we have used **X = 0.5 knots**, **Y = 5 minutes**, **Z = 5 minutes**, and **W = 100 knots**, since these values have shown good results in initial testing, with regards to splitting trajectories at points which seem natural. Changing these values is associated with a large computational cost since all of the trajectory reconstruction needs to be reprocessed.

For splitting the journeys into sailing- and stopped trajectories, we have created the algorithms shown in Algorithm 1, 2, 3, 4, and 5, where $FpNh$ is the first point not handled, $LpTh$ is the last point over **X knots**, $TsTh$ is the time since a point had a speed above the threshold **X knots**, $LpNs$ is the last point not skipped, Δt is the time since the previous point, and Δd is distance to the previous point.

The trajectory reconstruction process is divided into different algorithms, but they operate in the same global scope, meaning that all variables are shared.

Algorithm 1 is the entry point for the algorithm. In line 1, the input for the algorithm is a complete journey for a ship, meaning all of the data points from a ship in a given period, ordered by time, as well as the values for the parameters **X**, **Y**, **Z**, **W**, and **P**. $P_{sailing}$ and $P_{stopped}$ are temporary lists of AIS points that store trajectories while they are being reconstructed. Once a trajectory ends, it is pushed to the corresponding list of trajectories $T_{sailing}$ or $T_{stopped}$. The primary processing of the algorithm happens within the *for* loop shown on lines 11 to 27. We loop over each point in the journey and check if it should be a part of a sailing or stopped trajectory, or if it is an outlier. For each point, we set Δt and Δd to the time

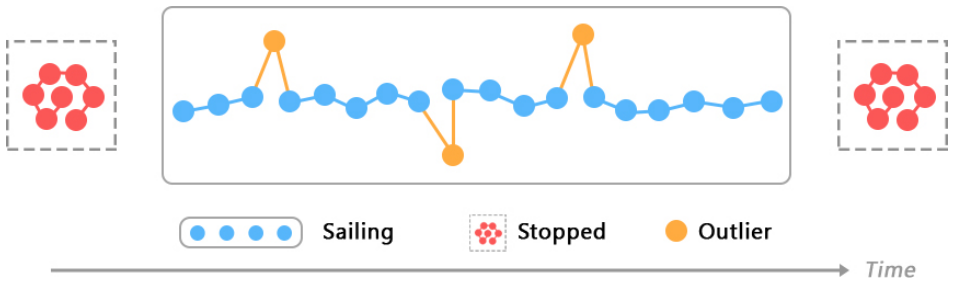


Figure 8: Illustration of trajectory splitting based on sailing and stopped movement patterns. Furthermore, outliers that exist on the trajectory have been highlighted [21]

and distance since the last point in lines 12 and 13, respectively. In line 14, we determine the speed since the last point. This is done by calculating the speed based on Δt and Δd and comparing it to the SOG value from the AIS record. If the absolute difference between the calculated speed and SOG is above a threshold (2 knots in our implementation), we use SOG since we deem it accurate enough. Otherwise, we use the calculated speed. This is done to be able to remove geographical outliers while being resilient to small variations in the accuracy of the coordinates.

In line 16, we check if the time since the last point in the journey is above the threshold **Z**, and in case it is, we end the current trajectory. This is done because we have observed large time gaps between AIS records. If we do not handle these large time gaps, we risk creating very inaccurate trajectories which, e.g., may be trajectories that go over land.

Lines 20 to 26 are three different ways of handling the point depending on the speed since the last point. The first case on line 20, is the outlier detection. The calculated speed is above the threshold possible for a vessel, and the point is, therefore, an outlier. The outlier point is skipped, as described in Algorithm 3. The second case is when the speed is above or equal to the minimum threshold **W**, meaning it should be part of a sailing trajectory, which is handled in Algorithm 4. The last case is that the speed is below the minimum threshold, meaning that it should be a part of a stopped trajectory, depending on how long it has been since a point had a speed greater than the threshold. This is handled in Algorithm 5.

Upon finishing the *for* loop, the remaining trajectories are added to either the $T_{sailing}$ or $T_{stopped}$ lists, after which we return both lists as the output of the algorithm.

EndCurrentTrajectory. . The intuition behind Algorithm 2 is to check if we are currently working on a sailing or stopped trajectory. We add any unhandled points, excluding the current point, to the trajectory, and then reset $FpNh$. If the trajectory consists of more than **P** points, we add it to the list of trajectories. Finally, we clear the temporary list of points $P_{sailing}$ or $P_{stopped}$.

SkipPoint. . The intuition behind Algorithm 3 is to add any unhandled points, excluding the current point, to the trajectory being constructed and then resetting $FpNh$.

HandleSailingPoint. . The intuition behind Algorithm 4 is to end a stopped trajectory if it is under construction and add all unhandled points including the current point to the sailing trajectory. We then set $FpNs$ to the current point and finally reset both $TsTh$ and $FpNh$.

HandleSlowPoint. . The intuition behind Algorithm 5 is to first check if too much time has passed since a point had a speed above the minimum threshold **Y**. If this is the case, the sailing trajectory is ended if it is currently under construction, and all unhandled points, including the current point, are added to the stopped trajectory. Finally, the $LpNs$ is set to the current point, and $FpNh$ is reset.

The draught measure in `fact_trajectory_sailing` and `fact_trajectory_stopped` is populated with an array consisting of all of the reported draughts from the data points making up the trajectory. The draught which is reported most often is the first element in the array and is the value that is used for the trajectory throughout the system. The foreign keys to `dim_ship_type`, `dim_cargo_type`, `dim_destination`, `dim_type_of_mobile`, `dim_data_source_type`, and `dim_type_of_position_fixing_device` are populated with the value which occurs most often in the data points making up the trajectory.

After the trajectory reconstruction and before the trajectories are added to the data warehouse, we simplify the trajectories using Douglas-Peuckers, with a tolerance of 0.001 degrees, because initial testing showed that this value gives a good balance between data reduction and information retention [20]. This is done to significantly reduce the amount of data used to store the trajectories. Simplifying using Douglas-Peuckers is especially effective because ships primarily move in straight lines, and we, therefore, see a reduction in data size of approximately 98%, as seen in Table 3.

6 IMPLEMENTATION

In this section, we describe how the system is implemented and when the implementation differs from the design. The section covers the overall system, how we process AIS data, and how the processed data is presented to the user through a web-based frontend. The source code for our implementation can be found on GitHub in which we have included a guide on running the system [48]. A web demo of the system is available at http://130.225.39.233:3000/.

6.1 System Overview

The system overview consists of various components, each with its own responsibility and role when defining the architecture of our multi-purpose data warehouse. A system overview diagram can be seen on Figure 9, in which the data flow is denoted with

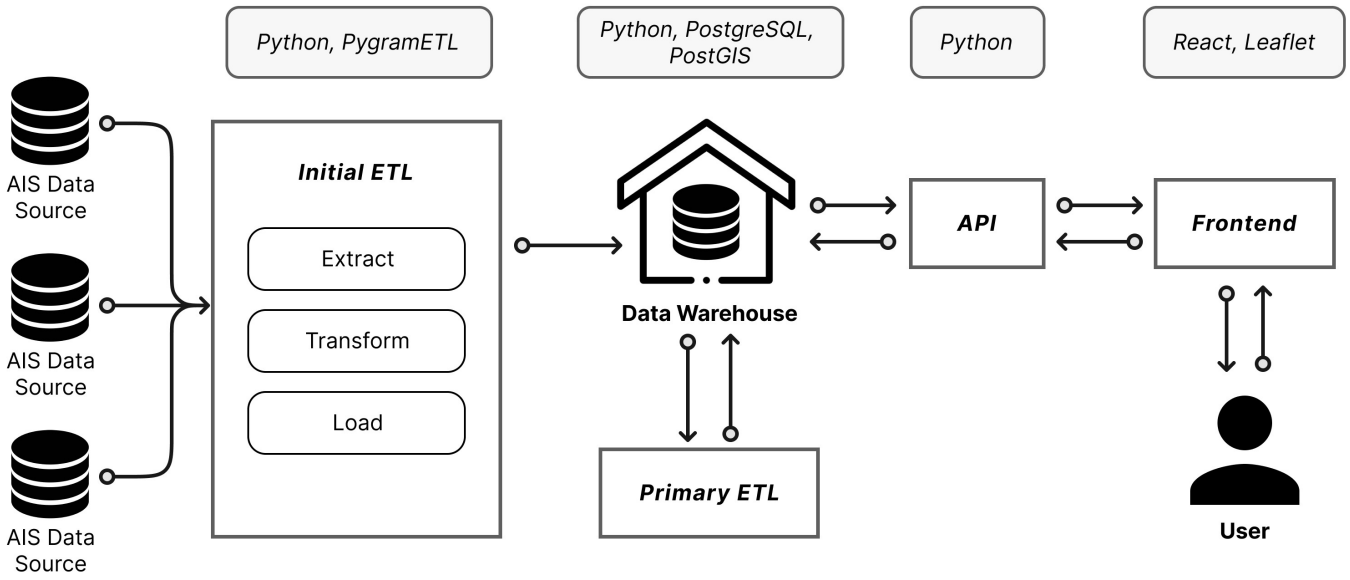


Figure 9: System overview illustrating the existing components and data flow within the system

arrows starting from the left-most component, the *AIS data source*. Furthermore, the primary technologies that have been utilized are displayed above each component. The versions used during the implementation are PostgreSQL v14.2, PostGIS v3.2, Python v3.10, and React v17.0.

The flow within our system can be separated into three primary processes: (i) initial ETL, (ii) primary ETL, and (iii) presentation. The initial ETL process consists of the *AIS data source* and *Initial ETL* components, and the process is responsible for extracting data from AIS sources and loading the data into our data warehouse. Upon completion, the next process, primary ETL, can be started. This process consists of two components, the *Data Warehouse* and *Primary ETL*. The process is responsible for extracting and transforming raw AIS data retrieved from the initial ETL process, into cleaned data, trajectories, and cell details. After the transformation phase, the new data is loaded into their respective tables in the data warehouse. In Section 6.2, a detailed description of this process can be found. The final process, presentation, is responsible for displaying data to the user. It utilizes the *API* component to transfer data between the *data warehouse* and the *frontend*. The data transfer is based on which requests the user makes.

6.2 Extract, Transform, and Load

The extract, transform, and load (ETL) process facilitates extracting data from source systems, transforming the data (e.g., applying calculations), and finally loading the data into the data warehouse. The process is particularly important when working with AIS data due to the amount of data, and the noise that exists in the datasets. In this section, we describe the three steps and explain their connection to our system.

Specifically for the AIS data provided to us by DMA, the AIS data is contained within CSV files that are generated on a per-day basis. During the extract step, data is extracted from the CSV files into a staging area. The staging area is a part of the transformation step and acts as a buffer between the data warehouse and the source system. As data may come from different sources, be corrupted, or vary in terms of format, it could be risky to directly transfer the data to the data warehouse. Therefore, the staging area is essential for data validation, cleaning, and organization.

In relation to our system, we have defined an initial staging area that only performs validation and organizational operations such as identifying the correct dimension or performing type validation on the data. This allows us to retain a representation of the original data within the data warehouse before cleaning and filter rules are applied. We refer to this data as the *raw* data. To ensure that we do not try to load corrupted, or incorrect data types, we still have to validate the extracted data. The validation process includes operations such as: ensuring data integrity, a uniform representation of unknown values, and setting initial values.

Apart from the initial staging area, we also define another staging area that is responsible for cleaning the *raw* data to improve the overall data quality and to filter data that are out of scope e.g., AIS data sent from helicopters. The second staging area is therefore a part of the primary ETL process in our system because it enforces domain-specific cleaning rules that are applied to the raw data.

The transformation step is also responsible for performing domain-specific operations such as patching of ship references and trajectory reconstruction. These steps are explained in details in Section 6.3 and Section 5 respectively. Furthermore, the transformation step is also responsible for populating the `fact_cell_sailing` table. The population is done by grouping data from `fact_trajectory_sailing` based on the foreign keys in `fact_cell_sailing`, and aggregating the measures.

After the transformation step has finished, the final loading step can begin. The loading step is responsible for loading the newly transformed data into the data warehouse. We use full loading

Algorithm 1 TrajectoryReconstruction

INPUT: complete journey for a ship as $journey$, X, Y, Z, W
OUTPUT: $T_{sailing}$, $T_{stopped}$
Begin
  $P_{sailing}$, $P_{stopped}$ ← empty lists of points
  $T_{sailing}$, $T_{stopped}$ ← empty lists of trajectories

  $TsTh \leftarrow 0$ // Used in Algorithm 4,5
  $LpNs, LpTh$ ← First point in $journey$
  $FpNh \leftarrow Null$ // Used in Algorithm 2,3,4,5

  for each $point$ in $journey$ **do**
    $\Delta t \leftarrow getTime(point, LpNs)$
    $\Delta d \leftarrow getDistance(point, LpNs)$
    $speed \leftarrow getSpeed(\Delta t, \Delta d, point)$

    if $\Delta t > Z$ **then**
      endCurrentTrajectory() //Algorithm 2
    end if

    if $speed > W$ **then**
      SkipPoint() // Algorithm 3
    else if $speed \geq X$ **then**
      HandleSailingPoint() // Algorithm 4
    else if $speed < X$ **then**
      HandleSlowPoint() // Algorithm 5
    end if
  end for

  if $length(P_{stopped}) > 0$ **then**
    add $P_{stopped}$ to $T_{stopped}$
  else if $length(P_{sailing}) > 0$ **then**
    add $P_{sailing}$ to $T_{sailing}$
  end if
  return $T_{sailing}, T_{stopped}$
End

Algorithm 2 EndCurrentTrajectory

Begin
  if $length(P_{sailing}) > 0$ **then**
    if $FpNh \neq Null$ **then**
      for each p in $journey[FpNh{:}point[$ **do**
        append p to $P_{sailing}$
      end for
      $FpNh \leftarrow Null$
    end if
    if $length(P_{sailing}) >$ **P then**
      add $P_{sailing}$ to $T_{sailing}$
    end if
    clear $P_{sailing}$
  end if
  if $length(P_{stopped}) > 0$ **then**
    if $FpNh \neq Null$ **then**
      for each p in $journey[FpNh{:}point[$ **do**
        append p to $P_{stopped}$
      end for
      $FpNh \leftarrow Null$
    end if
    if $length(P_{stopped}) >$ **P then**
      add $P_{stopped}$ to $T_{stopped}$
    end if
    clear $P_{stopped}$
  end if
End

Algorithm 3 SkipPoint

Begin
  if $FpNh = Null$ **then**
    return
  end if
  if $len(P_{stopped}) > 0$ **then**
    for each p in $journey[FpNh{:}point[$ **do**
      append p to $P_{stopped}$
    end for
  end if
  if $len(P_{sailing}) > 0$ **then**:
    for each p in $journey[FpNh{:}point[$ **do**
      append p to $P_{sailing}$
    end for
  end if
  $FpNh \leftarrow Null$
End

when loading into our fact tables, meaning that everything from the transformation goes into new, unique records in the tables. For the dimension tables, however, we utilize incremental loading, meaning that the incoming data is compared to the existing data, and we only produce additional records if new and unique information is found.

6.3 Patching References to Ships

As described in Section 6.2, all of our dimensions utilize incremental loading, meaning that new data is compared to existing data so that we do not have multiple entries of the same object. Specifically for `dim_ship` the lookup attributes used for comparison are: MMSI number, ship type, positional type, and mobile type, all of which are described in details in Section 3.1 and Section 4.1. For each unique combination in our data, we create an entry in the ship dimension. Unfortunately, we do experience multiple entries that represent the same ship, which is a result of, error-prone AIS data and the noise that we have described in Section 3.2. To account for the duplicate representations of ships, we have implemented a patching method that can fix the ship references on our data.

The patching is performed during the transformation step in the primary ETL process allowing us to utilize some of the organizational steps that already have been made. More specifically, the method utilizes a list of records being processed, grouped by their MMSI number. For each group, all ship references are counted, and the reference that occurs the most times is identified as the *correct reference*. We believe this approach is appropriate for our data

Algorithm 4 HandleSailingPoint

Begin
  $LpTh \leftarrow point$
  if $length(P_{stopped}) > 0$ **then**
    if $length(P_{stopped}) >$ **P then**
      add $P_{stopped}$ to $T_{stopped}$
    end if
    clear $P_{stopped}$
  end if
  if $FpNh \neq Null$ **then**
    for each p in $journey[FpNh{:}point]$ **do**
      append p to $P_{sailing}$
    end for
    $LpNs \leftarrow point$
  else
    append $point$ to $P_{sailing}$
    $LpNs \leftarrow point$
  end if
  $TsTh \leftarrow 0$
  $FpNh \leftarrow Null$
End

Algorithm 5 HandleSlowPoint

Begin
  if $FpNh = Null$ **then**
    $FpNh \leftarrow point$
  end if

  $TsTh \leftarrow point.time - LpTh.time$
  if $TsTh \leq Y$ **then**
    return
  end if

  if $TsTh > Y$ **then**
    if $length(P_{sailing}) > 0$ **then**
      if $length(P_{sailing}) >$ **P then**
        add $P_{sailing}$ to $T_{sailing}$
      end if
      clear $P_{sailing}$
    end if
    if $FpNh = Null$ or $point = FpNh$ **then**
      append $point$ to $P_{stopped}$
      $LpNs \leftarrow point$
    else
      for each p in $journey[FpNh{:}point]$ **do**
        append p to $P_{stopped}$
      end for
      $LpNs \leftarrow point$
    end if
    $FpNh \leftarrow Null$
  end if
End

because our observations show that in most cases, the duplicate references only exist because of noise in the AIS data from a specific time period. When a group has been identified to have more than one reference, we change the foreign key of `dim_junk_ais_clean` to the entry which indicates that the records have had their ship references patched. By doing so we are able to assign a flag that indicates the data have been modified and to what degree.

6.4 Data Warehouse

The data warehouse has been implemented in accordance with the design described in Section 4 with the exception of the changes that will be described in this section.

The data warehouse design shown on Figure 7 has been mostly implemented with the exception of some foreign keys within `fact_ais_clean`, namely the *trajectory_sailing_id* and *trajectory_stopped_id*. These foreign keys are part of the data warehouse design but are currently set to null values as we experienced some limitations within the Python library: PygramETL, which was used to implement our ETL processes. When a reconstructed trajectory is inserted in either `fact_trajectory` table, a reference to the row is not returned. This means that we lose the reference to which AIS records are used to construct the trajectory. We also did not populate the `bridge_traj_stopped_cell` because of time constraints in terms of CPU time on the server and it was not a priority for the use-case of making MDCs. The idea is exactly the same as with `bridge_traj_stopped_cell`, and it is, therefore, trivial to implement.

To improve performance when inserting many rows into the large fact tables, we utilize a combination of batch loading, disabling of triggers, dropping, and reconstruction of primary- and foreign key constraints.

Furthermore, to increase query performance for our frontend, we have implemented `fact_cell_sailing` as two fact tables, namely `fact_cell_sailing_50m` and `fact_cell_sailing_1000m`. As mentioned in Section 4.1, the cell size granularity is 50 meters and 1000 meters, and to aggregate cells from 50 meters to 1000 meters the resulting table is 400 times larger. This means that a lot of the computation time will be spent aggregating the cells up to 1000 meter cells, and this can be avoided by separating `fact_table` and storing the pre-aggregated 1000 meter cell values in a separate table. To obtain the query performance gain, the `dim_cell` dimension needs to be separated as well, one dimension for 50 meters and one for 1000 meters. Finally, as there is a many-to-many relation between `fact_tractory_sailing` and `dim_cell`, we construct a bridge table for both 50 meters and 1000 meters. Splitting `fact_cell_sailing` into multiple more granulated fact tables will of course incur a storage increase, which we deem worthwhile in relation to the performance gain.

Section 4.2 describes the indices used for the fact tables and dimensions, and the same type of indices is of course applied to the aforementioned `fact_cell_sailing`, 50 meters and 1000 meters as well as `dim_cell`, 50 meters and 1000 meters.

The dimension `dim_date` has been pre-filled with 2000 days starting from 1 January 2021. The same is true for `dim_time`. These two dimensions have been pre-filled as they contain static data.

The dimension dim_junk contains flags to categorize a record in `fact_ais_clean`. It currently holds two different kinds of flags, but more flags can be implemented to further categorize the data. The two flags are *is_patched_ship* and *is_outlier*. *is_patched_ship* is implemented and the flag is set when a ship has had its reference patched as described in Section 6.3. The flag *is_outlier* is implemented in the data warehouse and describes whether a record has been marked as an outlier in the trajectory reconstruction described in Section 5, however, this flag is always set to false because it has not been implemented in the transformation phase.

Tables Sizes. A summary of the table sizes in our data warehouse can be seen on Table 3. All dimensions and `fact_ais` are based on data spanning the whole year 2021, while the rest of the fact tables and bridge tables use data spanning from 01/05/2021 to 30/09/2021. The table name, number of rows, table size, index size, and total size are listed for each table. The total size for all tables in the current version of our data warehouse adds up to approximately 3.14 TB of storage.

6.5 API

Our API is written in Python using the web framework Flask, in addition with the RESTful extension to build a REST API. The API provides communication between the data warehouse and the frontend as seen in Figure 9. To get an MDC, the user creates a request based on their preferred filters. The filtering parameters are passed to the *get* endpoint in the API which processes the request by making a SQL query on the data warehouse with the provided parameters. The API receives the resulting data as a GeoJSON, i.e., a format that provides a representation of simple geographical features along with their non-spatial attributes [12]. The GeoJSON is returned to the frontend and is rendered as the resulting MDC. An example of a query that is sent to the data warehouse from the API is shown in Appendix Listing 1.

The histogram of draught for a cell is retrieved through a *get* request that include a `cell_id` within the request parameters. The API queries the data warehouse with the particular `cell_id` and retrieves an array representing the draught histogram. The array is returned to the frontend after which the array is transformed into an interactive graph.

6.6 Web-based Frontend

As described in Section 6.1, the system includes a frontend component within the presentation process. The frontend is implemented using the JavaScript library React in combination with Leaflet. The frontend is responsible for displaying the aggregated data retrieved from the data warehouse through HTTP requests made to the API. Furthermore, the frontend provides the user with a user interface (UI) that allows the addition of filters to the requests. Filters are a valuable way to give the user control of which type of data they wish to query. The currently supported filtering is based on the: date range, ship type, AIS transponder type, and data trustworthiness.

Upon choosing new filters, the frontend creates a new request to the API, which returns the aggregated cell data in a GeoJSON format, ultimately producing the MDC. Based on the zoom level when the request is made, the API determines the cell sizes used. Specifically for our implementation, we provide the user with two possible granularities: 50- and 1000-meter cells, ultimately allowing us to restrict the amounts of cells being shown to the user at the same time because too many cells cause poor performance due to browser limitations.

On the sidebar, we also provide the user with a switch button that can swap between showing an MDC or a heatmap of trajectory counts based on the fetched data. An example of an MDC with 1000 meter cells is shown in Figure 10, and an example of a heatmap can be seen in Appendix Figure 17

Figure 10 shows the user interface (UI) of our frontend. The UI consists of a *from* date-picker and a *to* date-picker that together represents the date range to query on. The filters have 3 categories: ship types, AIS transponder type, and trustworthiness. The ship types filter allows for selection of which ship types to include in the query, e.g, `Cargo`, `Fishing` or `Sailing`. The AIS transponder type can filter data based on the transponder types such as `Class A` and search and rescue transponders. Lastly, the trustworthiness filter includes ships with a trusted draught as described in Section 3.7. Clicking *apply* and *fetch* will retrieve the data based on the filters. In addition, the user can dynamically choose to view the MDC or the heatmap of the area, by toggling the associated button.

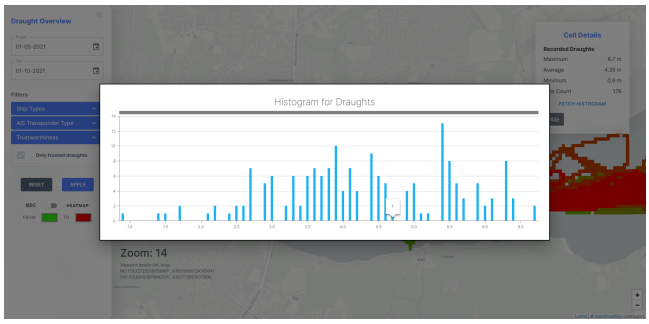


Figure 11: Histogram for a cell in Vejle Fjord. 50 meter cells.

Including all ship- and transponder types, Figure 10 shows the resulting MDC where pink areas near coastal areas indicate cells with only null draughts, and the depth grading scale goes from light blue to dark blue indicating low to high depth respectively. The two aforementioned colors are default values and can be customized through the color picker. Clicking on a cell will bring up a cell details box in which the maximum-, average-, and minimum draught and trajectory count are displayed. To display more detail about the cell, a histogram of the draughts can be requested. An example of the histogram is shown in Figure 11.

7 RESULTS

In this section, we first present our performance tests of the system. Next, a comparison between our MDC and a depth model provided by the DGA is analyzed and explained. Afterward, we address the use of waterways in Danish waters by utilizing heatmaps and then discuss the potential of including data without draught information when producing our MDCs. Finally, we summarize our most recent meeting with the DGA in which we presented the system and all of our results.

Table Name	Rows	Table Size	Index Size	Total Size
dim_type_of_position_fixing_device	12	8 kB	16 kB	24 kB
dim_navigational_status	21	8 kB	16 kB	24 kB
dim_data_source_type	2	8 kB	16 kB	24 kB
dim_type_of_mobile	8	8 kB	16 kB	24 kB
dim_junk_ais_clean	4	8 kB	16 kB	24 kB
dim_destination	29,751	1,792 kB	672 kB	2,464 kB
dim_cargo_type	8	8 kB	16 kB	24 kB
dim_ship_type	27	8 kB	16 kB	24 kB
dim_ship	88,550	8,232 kB	1,976 kB	10,208 kB
dim_date	2,002	312 kB	64 kB	376 kB
dim_time	86,400	3,776 kB	1,904 kB	5,680 kB
dim_audit	976	168 kB	40 kB	208 kB
dim_cell_50m	131,502,552	72 GB	15 GB	86 GB
dim_cell_1000m	329,430	53 MB	32 MB	84 MB
bridge_traj_sailing_cell_50m	1,044,225,732	43 GB	42 GB	85 GB
bridge_traj_sailing_cell_1000m	52,940,381	2,236 MB	2,143 MB	4,379 MB
fact_ais	4,084,765,528	652 GB	112 GB	765 GB
fact_ais_clean	1,641,590,137	424 GB	376 GB	801 GB
fact_trajectory_stopped	8,494,357	4,294 MB	356 MB	4,649 MB
fact_trajectory_sailing	2,514,635	4,043 MB	114 MB	4,157 MB
fact_cell_50m	814,984,634	1,320 GB	34 GB	1,354 GB
fact_cell_1000m	22,896,139	38 GB	841 MB	39 GB
Total	**7,804,451,286**	**2.56 TB**	**582.60 GB**	**3.14 TB**

Table 3: Table sizes

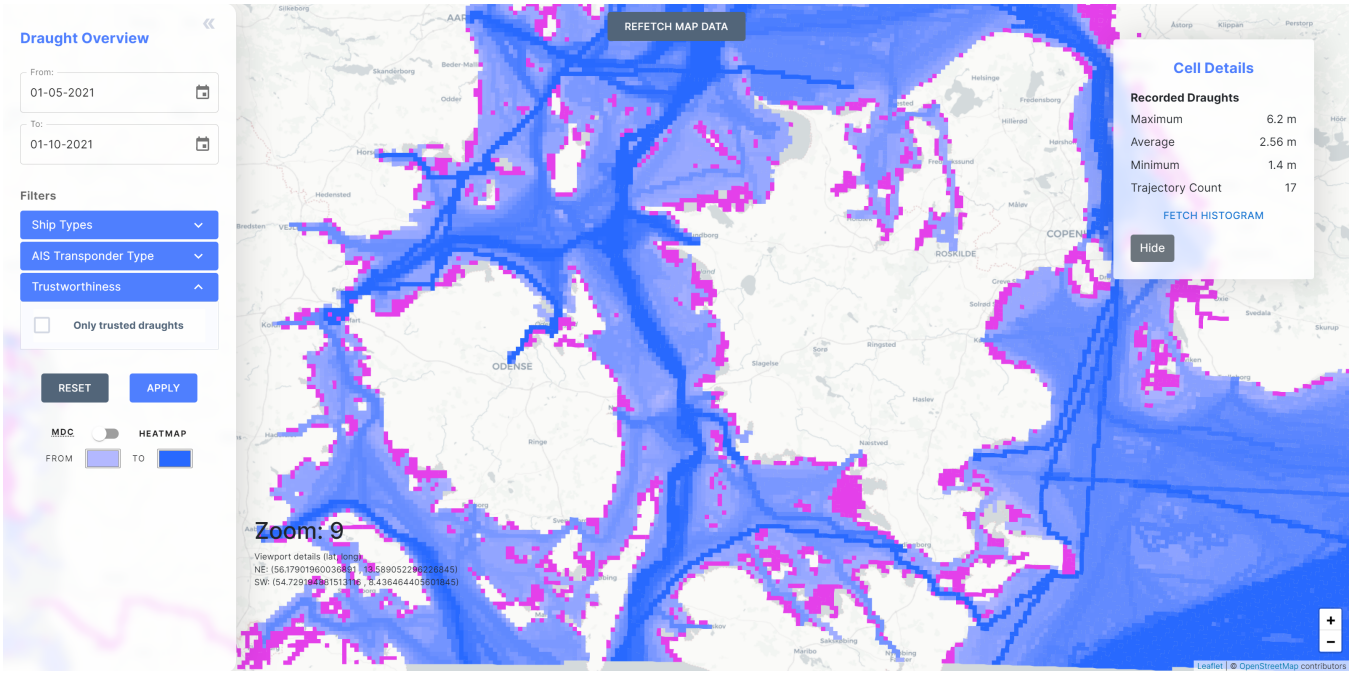


Figure 10: MDC of Fyn and Zealand with all ship- and AIS transponder types and with both trusted and untrusted draughts (150 km x 250 km)

7.1 System Performance

The performance of the system implemented in this paper can be split into two parts: the ETL performance and the query performance. The processing times were recorded on a server running Ubuntu 20.04 with 64GB of RAM and 8 AMD EPYC CPU cores running at 2GHz in a virtualized cloud environment.

ETL Performance. The ETL performance describes the processing time of extracting data from a CSV file and loading it into the data warehouse as described in Section 6.2. The processing times for the different ETL steps can be seen in Table 4. All of the processing was done on a per-day basis, and the numbers in Table 4 are based on the processing of data from 04/10/2021 consisting of 11,288,659 raw AIS records with data from 5 months already processed and inserted into the data warehouse. The exact processing times for each day vary based on the amount of data from that day, and the numbers in the table are therefore only approximate values. All of the transformation processing is independent of the amount of data in the data warehouse. However, inserts and especially constraint checks and index rebuilding are sensitive to the amount of data and gradually becomes slower.

The main takeaway from the processing times is that the total processing time of one day is comfortably within 24 hours, meaning that the system can handle data faster than it is generated.

Process	Time
Loading of AIS records	34 minutes
Cleaning of raw AIS records	34 minutes
Trajectory reconstruction	1 hour 19 minutes
Population of `fact_cell_sailing`	44 minutes
Constraint checks and index rebuilding	1 hour 57 minutes
Total	**5 hours 3 minutes**

Table 4: ETL processing times for data from 1 day

Query Performance. The query performance describes the time to perform the queries used in the frontend. These queries are highly sensitive to the number of cells in the queried area, how many trajectories have passed through the cells, as well as if the area has been queried recently as the result gets cached automatically by PostgreSQL. The fact that the server is in a virtualized environment also means that the performance can change without our knowledge and can also have an impact on the query times.

For queries within a short time span, which are either the same or very similar, we have experienced query times of below 5 seconds. If a longer time has passed since a query has been made or if it is an area that has not been queried before we have observed query times of up to 10 minutes.

The query times are higher than what is users are willing to wait for a response on a website [29]. The target users of the website are, however, domain experts who want to analyze the data and are willing to wait for a longer time. We believe the response times are within reason based on the domain and the amounts of data but could use some more optimization. The query times for histograms are very low and are within what regular users want to wait.

7.2 Verifying Ship Traffic

Figure 12 shows a heatmap of Kattegat for all ships in the period 01/05/2021 to 30/09/2021. The color gradient goes from light blue which means low intensity, to dark blue meaning high intensity. It is clear to see heavy traffic lines going into the harbors, and more interestingly, we can see two lines side by side north of Skagen going into the eastern part of Kattegat. These two lines are waterways where larger ships are required to sail due to safety regulations. The waterways split up close to Læsø depending on whether a ship's route goes to different destinations. Analyzing these waterways suggests that ships follow the guidelines set by the DMA when sailing within Danish waters. And the two close parallel lines verify that ships sail in the designated waterways.

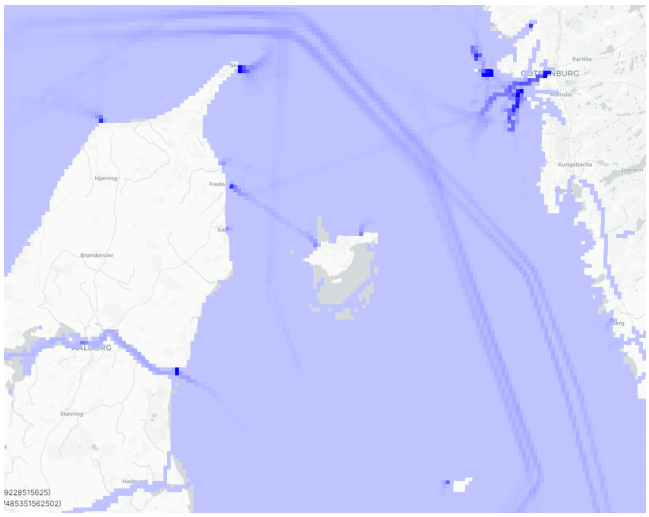


Figure 12: Heatmap of Kattegat

7.3 DGA Depth Model Comparison

Through collaboration with the DGA, we were provided a confidential depth model. The depth model is a raster consisting of 50m x 50m cells over Danish waters and is expected to be released in late 2022. The cells contain depth information that is either interpolated or measured and these data are based on the average depth for that area. Comparing the depth model to a raster version of our MDC, we can identify interesting areas in which there are inconsistencies. Making the comparison we are aware that our model uses a maximum reported draught based on AIS data, which will result in areas with large differences because the actual depth is much higher than any ship draught.

The scales on Figure 13 and Figure 14 annotate the value difference of the reported depth between the two models. The color gradient used in the scale goes from red to white to green. Red means our MDC shows a larger depth than the DGA model, and green means the DGA model shows a larger depth. White in between means the two charts show equal depth. Darker color means a larger difference and lighter color is a lower difference.

When looking at the entirety of Denmark, in Figure 13, we see almost only green, and this is as expected. Because ships should

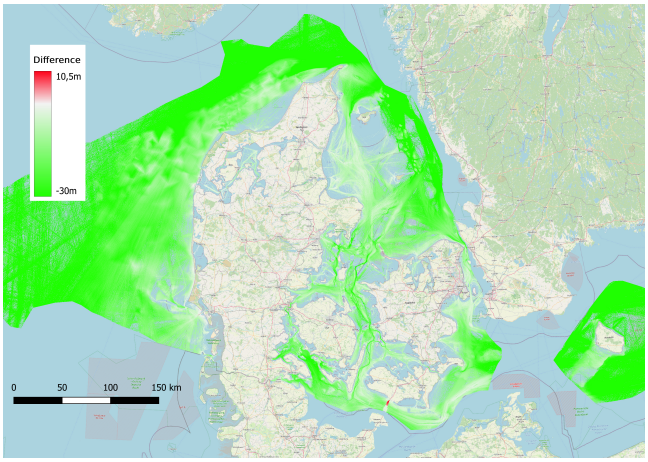


Figure 13: Comparison of the depth difference between DGA depth model and an MDC based on 5 months of data

not sail where their draught is larger than the depth, they will sail aground. In open waters far away from the coast, we see dark green. This is also as expected since our observations show that the depth can go beyond 25.5 meters, which is above the maximum draught for any ship. Light green areas such as Kattegat are interesting because these areas indicate that our MDC is close to the depth model that the DGA has provided.

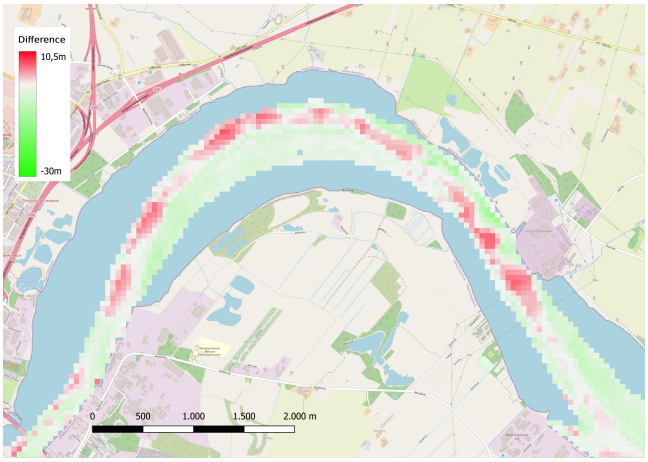


Figure 14: A comparison of our MDC and the DGA depth model in the Limfjord. The MDC is generated using 5 months of data

When looking closer into inlets at Figure 14, we observe red areas. Areas with red indicate that the MDC has recorded draughts that exceed the depth values reported by the DGA's depth model. These areas are of particular interest because they indicate that there may exist inconsistencies which imply that these areas require further investigation by the DGA. However, it is also worth mentioning that the inconsistencies may be a result of the constant change in the sea floor, inaccurate interpolated data, or outdated measurements. Also, the DGA model is based on an average depth within a cell, whereas our MDC is based on the maximum observed draught for that cell thus comparing these values may also contribute to a larger difference in depths, especially near coastal areas and waterways.

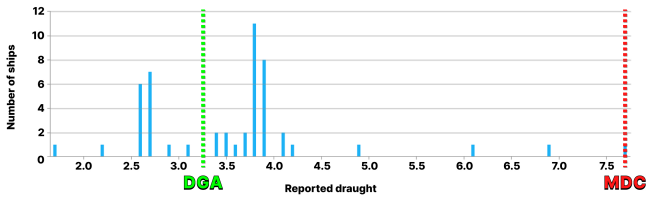


Figure 15: Histogram for a cell in the Limfjord

In Figure 15, a draught histogram of a 50m x 50m cell is shown. In this particular cell, from the Limfjord depicted in Figure 14, our MDC indicates a depth of 7.6 meters, marked with the red marker, while the DGA's model shows 3.1 meters, marked with the green marker. Further analyzing the depth inconsistency, we see that out of the 49 ships, 33 of them have reported draughts that exceed the depth reported by the DGA depth model. This demonstrates how we can utilize data from the histogram to further refine our analysis.

7.4 Including Missing Draught Data

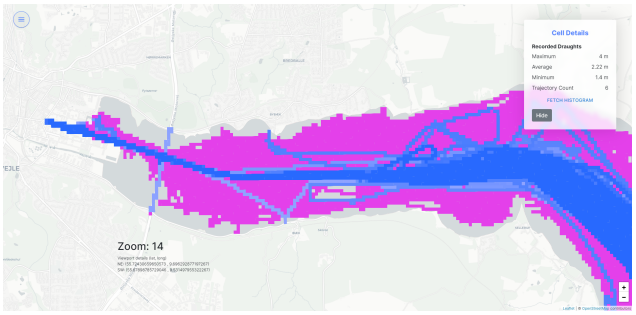
Figure 16 shows an MDC of Vejle Fjord. In Figure 16a, Vejle Fjord is depicted without null draught values and in Figure 16b null draught values as highlighted with a pink color. Otherwise, darker color means deeper and lighter means shallower water. Overall, Figure 16 highlights the potential of patching missing draught, as it could greatly increase the coverage of near coastal areas, and the trustworthiness with more measurements.

Analyzing the coverage of our data over Danish waters, we see that the coverage is 77,8% when we exclude data with null draughts and 88,7% when we include null draughts. It is an increase of 10.9 percentage points, and the coverage is greatly increased in near coastal areas. This is in line with our assumption that bigger ships, with class A transponders, sail in the open waters or where there are deep enough to sail. While smaller ships, typically with a class B transponder, sail in more shallow water. Therefore, by outfitting smaller ships with a trusted transponder, we can get more near coastal coverage. This is further discussed in Section 8.

7.5 Final Meeting with the DGA

In the final meeting with our collaboration partners at the DGA, we demonstrated our system, MDCs, and comparisons with their 50m depth model. The DGA had several representatives from different departments with different areas of interest. There was a very positive sentiment towards our system and its usefulness as a supportive tool for validation of existing bathymetry analysis, as well as a tool for determining which areas are most important for future surveys. It is important to select the areas for future surveys with care since the DGA is limited to surveying 2% of Danish waters using multi-beam per year. A topic of discussion was the west coast

(a) Vejle Fjord with only trusted draughts



(b) Vejle Fjord with non-trusted draughts

Figure 16: Vejle Fjord with and without trusted ship draughts. Five months data

of Denmark where their bathymetry data is very old. Through our tool, we showed that our MDC showed similar depth data, thus asserting that surveying the west coast may not be that high of a priority.

Representatives from the DGA also showed interest in using our system and the methods described for mapping secure waterways in very sparsely charted areas such as the waters around Greenland and the Lake Volta in Ghana. In general, they were very impressed with the possibilities of using AIS for bathymetric analysis.

8 CONCLUSION

In this paper, the design and implementation of a multi-purpose data warehouse model and a web-based application have been described. The goal has been to extend the data warehouse design that can efficiently store AIS data introduced in [21], and answer real-world questions regarding ships on the sea. Furthermore, the system is tailored to be flexible in order to support different use-cases, thereby making it multi-purpose. To demonstrate and prove the capabilities of our system, a web-based application was created that can request, process, and render data from the multi-purpose data warehouse. The application was demonstrated during a presentation for the Danish Geodata Agency (DGA) and received positive feedback in terms of capabilities and relevance. Specifically, they reacted positively to the usefulness of the MDC and histogram functionality of our system, the comparison with their 50m depth model, and the general usefulness of AIS for bathymetric analysis.

In Section 3, we analyzed the characteristics and reliability of AIS data, and also address some of the common errors that exist in the data. We further explain how trajectories can be utilized to represent sets of data as lines and describe our cleaning techniques and data assumptions. In Section 4, we describe how the data warehouse was specifically designed to store all attributes that exist in AIS data to support the data foundation for a variety of use-cases. Doing so has allowed us to exchange disk space for more detailed data, enabling analysis in many areas and not just bathymetry. The data warehouse consists of 4 billion raw AIS records that have been transmitted within Danish waters during 2021. To ensure data reliability we have processed the data for the period 01/05/2021 to 30/09/2021 and stored the 1.6 billion resulting records in a separate table which functions as the foundation for our analysis. Working with big data and AIS data, in particular, we gained most of our performance by creating indices, partitions, reconstruction of trajectories, and data compression.

Section 6 explain how we have implemented our system as an application that can generate and present an MDC or heatmap on an interactive map displayed within a web-based solution. Feedback received during meetings with DGA emphasizes the potential of our system as a supportive tool during their analysis of danish waters. The application allows DGA to analyze specific areas fast and cheaply, which they can utilize in addition to their own analysis. Such analysis does not only allow DGA to potentially identify areas that require additional or new bathymetric surveys but may also provide additional information because our model utilizes data from the ships. As DGA have mentioned, our system can be considered a proof-of-concept that not only proves to be relevant in their field of research but also addresses the potential of using AIS data for answering other real-world questions.

In Section 7.3 we describe the comparison of an MDC that was generated by our system, to a depth model provided by the DGA. The comparison shows the differences between the two models and as expected we do see that DGA often reports a much higher depth than our model which we did expect due to the nature of using reported draughts for our model. However, more interestingly, we identify areas in which our model indicates that ships have sailed at areas that the DGA model deems too shallow, indicating that our model may be able to find areas that contain incorrect depth data. These findings were presented during a meeting with DGA, and their feedback confirmed that our system potentially can find areas where their measurements may be outdated and require new surveys. Our findings have started internal discussions in DGA because our results have presented ideas as to how AIS data can be beneficial for their existing work which they hope to pursue in the future. Furthermore, our model has contributed to further confirming some of the assumptions that DGA has regarding their own data. For example, the assumption that there are a lot of changes in the seafloor in Limfjorden, and that the measurements on the west coast of Denmark are pretty reliable even though their measurements date many years back.

Future Work

There are different aspects of the multi-purpose data warehouse that could be improved on as future works. Including null draught patching, additional purposes, and query performance.

We want to emphasize the potential in patching missing draught values for future work. With the currently stored data, almost half of all sailing trajectories have reported null draught values. If a patching process could reliably estimate a draught for these trajectories, we potentially could increase the coverage of the MDC by a significant amount, especially near the coastal areas. We believe patching missing draughts can provide a lot of value and if a good data source for ship draughts can be found, it can be integrated into the existing system with little work.

The data warehouse is used for bathymetric analysis, but the multi-purpose aspect enables the data warehouse to be used for many different use-cases. For future work, we think analyzing sailing routes, near misses, or abnormal sailing patterns could be some interesting areas to look into. Even though the data warehouse holds this data, the system contains no implementation for analyzing or showing it, and it requires significant work to incorporate other types of analysis. We also believe it can be interesting to integrate new sources of AIS data that extend beyond danish waters, especially if combined with tide levels to more precisely assess the bathymetric details on the sea.

Query performance is an area that could benefit from more work. Even though we deemed our query times to be within reason, we believe this is an area where large improvements could be achieved with little work. For these improvements, tools like materialized views and partitioning could be used. This will make analyzing data more smooth and interactive.

ACKNOWLEDGMENTS

We would like to thank the Danish Geodata Agency for their contribution and willingness to work together closely. More specifically we would like to express our gratitude to Ove Andersen for his continuous feedback, insights, and expertise. Furthermore, we thank Anne Mette Egge Olsen for providing a map outlining the coastal areas of Denmark, which greatly helped us during our data cleaning process.

REFERENCES

[1] Introduction to IMO. URL https://www.imo.org/en/About/Pages/Default.aspx.
[2] AIS decoded fields descriptions and their values, . URL https://faq.spire.com/ais-decoded-fields-descriptions-and-their-values.
[3] AIS Dataset - AIS Data - VT Explorer, . URL https://api.vtexplorer.com/docs/response-ais.html.
[4] GEBCO Overview. URL https://www.gebco.net/about_us/overview/.
[5] About Us. URL http://eng.gst.dk/about-us.
[6] IHO Data Centre for Digital Bathymetry (DCDB). URL https://www.ngdc.noaa.gov/iho/.
[7] AIS transponders, . URL https://www.imo.org/en/OurWork/Safety/Pages/AIS.aspx.
[8] IMO and the Sustainable Development Goals, . URL https://www.imo.org/en/MediaCentre/HotTopics/Pages/SustainableDevelopmentGoals.aspx.
[9] Using Vessel Tracking Data to Prioritize Bathymetric Surveying in a Rapidly Changing Arctic | Alaska Ocean Observing System, . URL https://legacy.aoos.org/ais-4-bathy/.
[10] Vejledning til kaldesignal | Søfartsstyrelsen, 2021. URL https://www.soefartsstyrelsen.dk/publikationer-og-vejledninger/vejledninger/2020/sep/vejledning-til-kaldesignal.
[11] List of largest ships by gross tonnage, July 2021. URL https://en.wikipedia.org/w/index.php?title=List_of_largest_ships_by_gross_tonnage&oldid=1033137819. Page Version ID: 1033137819.
[12] GeoJSON, May 2022. URL https://en.wikipedia.org/w/index.php?title=GeoJSON&oldid=1086976873. Page Version ID: 1086976873.
[13] Automatic identification system, Apr. 2022. URL https://en.wikipedia.org/w/index.php?title=Automatic_identification_system&oldid=1085084978. Page Version ID: 1085084978.
[14] T. D. G. Agency. *Behind The Nautical Chart - Surveying, reliability and use*. The Danish Geodata Agency, Lindholm Brygge 31, third edition edition, 2018. ISBN 9788792107879.
[15] A. Alessandrini, M. Alvarez, H. Greidanus, V. Gammieri, V. F. Arguedas, F. Mazzarella, C. Santamaria, M. Stasolla, D. Tarchi, and M. Vespe. Mining vessel tracking data for maritime domain applications. In *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*, pages 361–367. IEEE, 2016.
[16] O. Andersen, B. B. Krogh, C. Thomsen, and K. Torp. An Advanced Data Warehouse for Integrating Large Sets of GPS Data. In *Proceedings of the 17th International Workshop on Data Warehousing and OLAP*, DOLAP '14, pages 13–22, New York, NY, USA, Nov. 2014. Association for Computing Machinery. ISBN 9781450309998. doi: 10.1145/2666158.2666172. URL https://doi.org/10.1145/2666158.2666172.
[17] S. Arasteh, M. A. Tayebi, Z. Zohrevand, U. Glässer, A. Y. Shahir, P. Saeedi, and H. Wehn. Fishing Vessels Activity Detection from Longitudinal AIS Data. In *Proceedings of the 28th International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '20, pages 347–356, Seattle, WA, USA, Nov. 2020. Association for Computing Machinery. ISBN 9781450380195. doi: 10.1145/3397536.3422267. URL https://doi.org/10.1145/3397536.3422267.
[18] K. Bereta, K. Chatzikokolakis, and D. Zissis. Maritime Reporting Systems. In A. Artikis and D. Zissis, editors, *Guide to Maritime Informatics*, pages 3–30. Springer International Publishing, Cham, 2021. ISBN 9783030618520. doi: 10.1007/978-3-030-61852-0_1. URL https://doi.org/10.1007/978-3-030-61852-0_1.
[19] X. Chen, J. Ling, Y. Yang, H. Zheng, P. Xiong, O. Postolache, and Y. Xiong. Ship Trajectory Reconstruction from AIS Sensory Data via Data Quality Control and Prediction. *Mathematical Problems in Engineering*, 2020:e7191296, Aug. 2020. ISSN 1024-123X. doi: 10.1155/2020/7191296. URL https://www.hindawi.com/journals/mpe/2020/7191296/.
[20] D. H. Douglas and T. K. Peucker. Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10: 112–122, 1973.
[21] P. Egede, K. Nielsen, M. Olsen, and A. Øztürk. Multi-purpose data warehousing for ais data. Jan. 2022.
[22] S. V. B. GmbH. Everything there is to know about AIS. URL https://www.svb24.com/en/guide/ais.html.
[23] A. Harati Mokhtari, A. Wall, P. Brooks, and J. Wang. Automatic identification system (ais): Data reliability and human error implications. *Journal of Navigation*, 60:373 – 389, 09 2007. doi: 10.1017/S0373463307004298.
[24] A. Harati-Mokhtari, A. Wall, P. Brooks, and J. Wang. Automatic identification system (ais): data reliability and human error implications. *The Journal of Navigation*, 60(3):373–389, 2007.
[25] R. Kimball and J. Caserta. *The Data WarehouseETL Toolkit: Practical Techniques for Extracting, Cleaning, Conforming, and Delivering Data*. John Wiley & Sons, Inc., 2011. ISBN 9780764579233. OCLC: 780986568.
[26] M. Kleppmann. *Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems*. O'Reilly Media, Boston, first edition edition, 2017. ISBN 9781449373320. OCLC: ocn893895983.
[27] MarineTraffic. What is the significance of the ais navigational status values?, 2021. URL https://help.marinetraffic.com/hc/en-us/articles/203990998-What-is-the-significance-of-the-AIS-Navigational-Status-Values-.
[28] MarineTraffic. What is the significance of the ais shiptype number?, 2021. URL https://help.marinetraffic.com/hc/en-us/articles/205579997-What-is-the-significance-of-the-AIS-Shiptype-number-.
[29] F. F.-H. Nah. A study on tolerable waiting time: how long are web users willing to wait? *Behaviour & Information Technology*, 23(3):153–163, 2004. doi: 10.1080/01449290410001669914. URL https://doi.org/10.1080/01449290410001669914.
[30] NAI. 200552 AIS Inaccuracies. URL https://www.nautinst.org/resources-page/200552-ais-inaccuracies.html.
[31] E. Novaczek, R. Devillers, and E. Edinger. Generating higher resolution regional seafloor maps from crowd-sourced bathymetry. *PloS one*, 14(6):e0216792, 2019.
[32] C. O'Brien, L. Lefebvre, J. Aubry-Morin, and K. Fadaie. The development of a draught information system for the st. lawrence seaway. *GEOMATICA*, 70(2): 137–142, 2016.
[33] J. Pattanaphanchai, K. O'Hara, and W. Hall. Trustworthiness criteria for supporting users to assess the credibility of web information. In *Proceedings of the 22nd International Conference on World Wide Web*, pages 1123–1130, 2013.
[34] C. Prandi, S. Mirri, and P. Salomoni. Trustworthiness assessment in mapping urban accessibility via sensing and crowdsourcing. 2014. URL https://dl.acm.org/doi/pdf/10.4108/icst.urb-iot.2014.257267.
[35] C. Prandi, S. Ferretti, S. Mirri, and P. Salomoni. Trustworthiness in crowd-sensed and sourced georeferenced data. 2015. URL https://ieeexplore-ieee-org.zorac.aub.aau.dk/stamp/stamp.jsp?tp=&arnumber=7134071&tag=1.
[36] C. Prandi, S. Ferretti, S. Mirri, and P. Salomoni. On the need of trustworthy sensing and crowdsourcing for urban accessibility in smart city. 2017. URL https://dl.acm.org/doi/pdf/10.1145/3133327.
[37] E. Robertson. Crowd-sourced bathymetry data via electronic charting systems. In *ESRI Ocean GIS Forum*, 2016.

[38] L. Salmon, C. Ray, and C. Claramunt. Continuous detection of black holes for moving objects at sea. In *Proceedings of the 7th ACM SIGSPATIAL International Workshop on GeoStreaming*, IWGS '16, pages 1–10, New York, NY, USA, Oct. 2016. Association for Computing Machinery. ISBN 9781450345798. doi: 10.1145/3003421.3003423. URL https://doi.org/10.1145/3003421.3003423.

[39] K. Sheng, Z. Liu, D. Zhou, A. He, and C. Feng. Research on ship classification based on trajectory features. *The Journal of Navigation*, 71(1):100–116, 2018.

[40] Y. Sun, X. Chen, L. Jun, J. Zhao, Q. Hu, X. Fang, and Y. Yan. Ship Trajectory Cleansing and Prediction with Historical AIS Data Using an Ensemble ANN Framework, 2021. URL https://doi.org/10.24507/ijicic.17.02.443.

[41] C. Tang, H. Wang, J. Liu, and C. Liu. A design for AIS data warehouse of traffic flow. In *2017 4th International Conference on Transportation Information and Safety (ICTIS)*, pages 895–898, Aug. 2017. doi: 10.1109/ICTIS.2017.8047874.

[42] M.-C. Tsou. Online analysis process on Automatic Identification System data warehouse for application in vessel traffic service. *Proceedings of the Institution of Mechanical Engineers, Part M: Journal of Engineering for the Maritime Environment*, 230(1):199–215, Feb. 2016. ISSN 1475-0902, 2041-3084. doi: 10.1177/1475090214541426. URL http://journals.sagepub.com/doi/10.1177/1475090214541426.

[43] Z. Yan, Y. Xiao, L. Cheng, R. He, X. Ruan, X. Zhou, M. Li, and R. Bin. Exploring AIS data for intelligent maritime routes extraction. *Applied Ocean Research*, 101:102271, Aug. 2020. ISSN 0141-1187. doi: 10.1016/j.apor.2020.102271. URL https://www.sciencedirect.com/science/article/pii/S0141118720303631.

[44] Z. Yuan, J. Liu, Y. Liu, and Z. Li. A novel approach for vessel trajectory reconstruction using ais data. 07 2019.

[45] L. Zhang, Q. Meng, Z. Xiao, and X. Fu. A novel ship trajectory reconstruction approach using ais data. *Ocean Engineering*, 159:165–174, 2018. ISSN 0029-8018. doi: https://doi.org/10.1016/j.oceaneng.2018.03.085. URL https://www.sciencedirect.com/science/article/pii/S0029801818303676.

[46] L. Zhao and G. Shi. A method for simplifying ship trajectory based on improved douglas–peucker algorithm. *Ocean Engineering*, 166:37–46, 2018. ISSN 0029-8018. doi: https://doi.org/10.1016/j.oceaneng.2018.08.005. URL https://www.sciencedirect.com/science/article/pii/S0029801818314872.

[47] F. Zhu. Mining ship spatial trajectory patterns from AIS database for maritime surveillance. In *2011 2nd IEEE International Conference on Emergency Management and Management Sciences*, pages 772–775, Aug. 2011. doi: 10.1109/ICEMMS.2011.6015796.

[48] A. Øztürk, P. K. Egede, M. E. Olsen, and K. S. H. Nielsen. Analyzing Billions of AIS Records in a Multi-Purpose AIS Data Warehouse, 6 2022. URL https://github.com/pegede17/AIS-Data-Warehousing-and-Bathymetric-Analysis.

A AIS DATA STRUCTURE

Static		
Field name	**Source**	**Comment**
IMO	IMO number of the vessel	Only propelled, seagoing vessels of 100 gross tons and above are assigned an IMO number.
Callsign	Callsign of the vessel	
Name	Name of the vessel	
Ship type	Describes the AIS ship type of this vessel	
Cargo type	Type of cargo from the AIS message	
Width	Width of the vessel	
Lenght	Lenght of the vessel	
Type of position fixing device	Type of positional fixing device from the AIS message	
Draught	Draugth field from AIS message	
Destination	Destination from AIS message	Manually set by the crew
Type of mobile	Type of target the message is received from	
ETA	Estimated Time of Arrival	
Data source type	Data source type, e.g. AIS	
Size A	Length from GPS to the bow	
Size B	Length from GPS to the stern	
Size C	Length from GPS to starboard side	
Size D	Length from GPS to port side	

Table 5: AIS data format [1]

Dynamic		
Field name	**Source**	**Comment**
MMSI	MMSI number of vessel	
ROT	Rate of turn from AIS message	
SOG	Speed over ground from AIS message	
COG	Course over ground from AIS message	
Latitude	Latitude of message report	Set using the device in "Type of position fixing device"
Longitude	Longitude of message report	Set using the device in "Type of position fixing device"
Heading	Heading from AIS message	
Timestamp	Timestamp from AIS basestation	
Navigational status	Navigational status from AIS message	Manually set by the crew

Table 6: AIS data format [1]

B EXAMPLES FROM APPLICATION

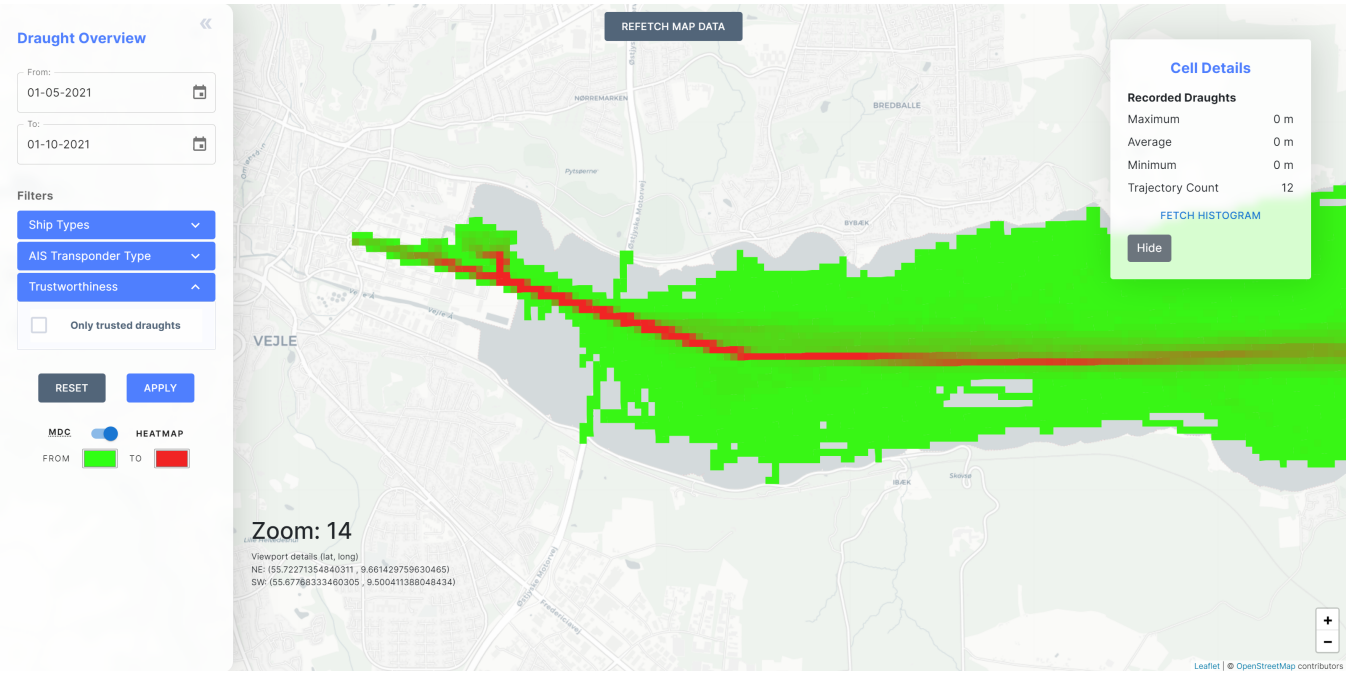


Figure 17: Heatmap of Vejle Fjord with low activity indicated by green and high activity indicated by red. Activity color gradient is from green to red. 50 meter cells.

C EXAMPLE QUERY FROM API

Listing 1: Example query of an MDC called by the API

```
SELECT
  json_build_object(
    'type',
    'FeatureCollection',
    'features',
    json_agg(
      ST_AsGeoJSON(result.*):: json
    )
  )
FROM
  (
    SELECT
      ST_Transform(boundary, 4326),
      max_draught,
      min_draught,
      trajectory_count,
      cell_data.cell_id,
      avg_draught
    FROM
      dim_cell_1000m d
      inner join (
```

[1]Gatehouse Maritime https://gatehouse.com/maritime/ and [18]

```
SELECT
  MAX(max_draught) max_draught,
  MIN(min_draught) min_draught,
  cell_id,
  SUM(trajectory_count) trajectory_count,
  SUM(avg_draught * trajectory_count)/
    CASE WHEN SUM(
        CASE WHEN avg_draught IS NOT NULL
             THEN trajectory_count ELSE 0 END
     ) > 0 THEN SUM(
     CASE WHEN avg_draught IS NOT NULL
             THEN trajectory_count ELSE 0 END
     ) ELSE 1 END avg_draught
from
  fact_cell_sailing_1000m
WHERE
  cell_id in (
    SELECT
      cell_id
    FROM
      dim_cell_1000m
    WHERE
      ST_Intersects(
        boundary,
        ST_Transform(
          ST_MakeEnvelope (
            9.335632324218752,
            55.390812118753104,
            11.972351074218752,
            56.11493571902954,
            4326
          ),
          3034
        )
      )
  )
  AND date_id BETWEEN 20210501 AND 20210507
  AND ship_type_id IN (
    SELECT
      ship_type_id
    from
      dim_ship_type
    WHERE
      ship_type IN (
        'Unknown', 'Other', 'Passenger', 'Cargo',
        'Pilot', 'Tanker', 'Sailing', 'Fishing',
        'Dredging', 'SAR', 'Undefined', 'HSC',
        'Tug', 'Port␣tender', 'Reserved',
        'Not␣party␣to␣conflict', 'Military',
        'Law␣enforcement', 'Pleasure', 'Diving',
        'Towing␣long/wide', 'Anti-pollution',
        'Towing', 'Medical', 'Spare␣2', 'Spare␣1',
```

```
                'WIG'
              )
          )
          AND type_of_mobile_id IN (
            SELECT
              type_of_mobile_id
            from
              dim_type_of_mobile
            WHERE
              mobile_type IN (
                'Unknown', 'Class␣A', 'Class␣B', 'Base␣Station',
                'AtoN', 'SAR␣Airborne', 'Search␣and␣Rescue␣Transponder'
              )
          )
          AND is_draught_trusted
        GROUP BY
          cell_id
      ) cell_data on cell_data.cell_id = d.cell_id
  ) as result(
    geom, maxDraught, minDraught, trajectory_count,
    cellId, avgDraught
  );
```