Verification of Meetings at Sea Using a Grid Approach and AIS Data Group 12

Yudong Fan
2662762
y.fan@student.vu.nl

Hongyu Wu
2762404
h.wu2@student.vu.nl

Yihong Xi
2769073
y.xi2@student.vu.nl

ABSTRACT

This paper provides detailed documentation about the group project of the course Large Scale Data Engineering held by Peter Boncz at Vrije Universiteit Amsterdam. The topic of the project is to verify potential ship-to-ship (STS) meetings at sea by analyzing AIS data[1]. Specifically, a data engineering pipeline which is capable of analyzing, extracting, and transforming AIS data and verifying potential STS meetings within the North Atlantic Ocean is introduced. Throughout the project, the AIS data of July 2016 with a size of approximately 160GB was processed, and the potential STS meetings are verified by utilizing the Geohash algorithm [13] and the definition of STS meetings proposed by Miller et al. [10]. As a result, approximately 700,000 ships which engage in potential STS meetings within North Atlantic Ocean in July 2016 are selected and stored in JavaScript Object Notation(JSON) format as the final data product of this project. Furthermore, the data product is visualized into a ReactJS-based front-end representation for showcasing.

1. INTRODUCTION

Ships can engage in a certain event which is described as a meeting at sea or a ship-to-ship (STS) meeting for various purposes. For example, cargo transshipment, crew changing, on-boarding pilot, and more are valid reasons [1]. Nonetheless, even though most of the reasons for STS meetings are rational and legitimate, illegal actions such as smuggling can be engaged as well [1]. In addition, STS meetings are sophisticated actions and can lead to serious incidents such as a collision of ships if they are carried out in an inappropriate manner. As a matter of fact, ships in principle need to meet a series of prerequisites including assessing the meeting area, notifying relevant authorities, deploying special equipment, etc. to properly perform an STS meeting [16]. Therefore, STS meetings need to be supervised for good reasons. Consequently, a system that is capable of analyzing maritime data, verifying potential STS meetings, and providing extra information about the meetings is of interest.

This project concerns mainly how to verify potential STS meetings using AIS messages. In short, AIS stands for automatic identification system, and AIS message is a typical maritime data which is capable of providing position, identification, and other information about the ship to other ships and to coastal authorities [6]. Theoretically, STS meetings happen when two conditions are fulfilled. Namely, two or more than two ships must be close enough to each other to make cargo transshipment possible, and they have to maintain that relatively close distance for an extended period of time. Notably, unlike traditional meetings, STS meetings do not necessarily require involved ships to stop at a fixed location. That is, ships can keep a short distance from others both when they are underway together or anchored together [16]. This nature of STS meetings makes the verification for potential STS meetings a more complex task than a naive geographical data analysis for ships because moving targets need to be considered as well.

As a result, this project intends to design a data process pipeline that is capable of extracting and analyzing AIS data to verify potential STS meetings. Furthermore, extra information about the verified STS meetings which are extracted from the AIS data will be provided.

The remainder of the paper is structured as follows. First, the research questions of this project will be elaborated. Specifically, both the inquiries regarding how to verify potential STS meetings using AIS data and the concerns on the technological side of the solution will be discussed. Second, related work will be discussed to provide an overview of the tools and methodologies this project builds on. Third, the design and implementation of the project will be discussed thoroughly. It includes a detailed description of the input and output data, the ETL (Extract-Transform-Load) data engineering pipeline of the project, the evaluation of the final data product and the ETL pipeline, and the visualization of the final data product. Then, a discussion section that evaluates the project setup is carried out. Finally, a conclusion that answers the research questions and wraps up the entire paper is given as well as a subsection that talks about the limitations of the project and future work directions.

2. RESEARCH QUESTION

The research questions of this project can be divided into two perspectives. The first perspective concerns the topic of the project, namely how to identify potential STS meetings using AIS data. Specifically, we are curious about what is the explicit and quantified definition of potential STS meetings. Also, we want to know what are the characteristics of AIS data, and what features in AIS data are correlated to the detection of potential STS meetings. Furthermore, what methodologies, techniques, or tools are suitable in terms of building a data engineering pipeline that is capable of analyzing large-scale AIS data and identifying potential STS

[1]`https://en.wikipedia.org/wiki/Automatic\_identification\_system`

meetings? Finally, what does the exact data engineering pipeline look like?

The second perspective concerns the technical side of the selected approach for detecting STS meetings. Particularly, we want to know the performance of the data engineering pipeline. The performance includes the feasibility and efficiency of the approach, especially when applied to a large dataset, the scalability of the approach, and the limitation or drawbacks of the approach.

3. RELATED WORK

In this section, the related work concerning tools and methodologies for the verification of potential STS meetings is discussed. The first subsection introduces the Geohash algorithm which is used to encode geographic locations into a grid in which each cell of the grid has a unique ID and a predetermined scale [13]. The second one is related to the identification of STS meetings including the definition of STS meetings [10].

3.1 Geohash

Geohash was invented and made public by G. Niemeyer in 2008 [13]. It is an algorithm that encodes geographic locations into unique strings of letters and digits. Particularly, Geohash is a hierarchical spatial data structure that implements the *Z-order curve* and splits geographic locations into cells of grid shape. In short, *Z-order curve* is a function that maps multidimensional data to one dimension while preserving the locality of the data points [12]. Given these characteristics of the Geohash algorithm, one of the most common applications of the algorithm is to grid the world with unique identifiers (i.e., Geohash strings) for every cell of the grid.

The Geohash string is encoded in the base 32 format, which means every character of the Geohash string can represent a value from 0 to 31. That is, every character in the string divides the map into a grid with 32 cells. Notably, the precision of the Geohash algorithm, namely the scale of the grid is determined by the length of the corresponding Geohash string. For example, as illustrated in Figure 1, the first digit of the Geohash string divides the world into a grid with 32 cells, and the scale for the grid cell is approximately $5000km * 5000km$. Furthermore, each cell contains a smaller grid with 32 cells, which is represented by the second digit of the Geohash string, and the scale for the smaller grid cell is approximately $1250km * 625km$. This process repeats until the Geohash string reaches its last digit. Specifically, the maximum level of Geohash precision is 12, namely a Geohash string can contain up to 12 characters, and the scale for the grid cell, in that case, is $3.7cm * 1.9cm$. This characteristic of the Geohash makes it possible to grid the world on different scales.

In the context of verifying potential STS meetings, the Geohash grid can be applied to identify whether ships are within a certain range at a given moment. For example, assuming the maximum distance that ships have to keep in order to engage in an STS meeting is 150m, a feasible approach is to use Geohash with a precision of 7 to encode the sea locations into a grid with a cell scale of approximately $150m * 150m$ and compare the cell IDs of which the two ships are inside at the given moment. If the cell IDs of the ships are identical, we can conclude that these two ships are within the same $150m * 150m$ area of the sea at the given

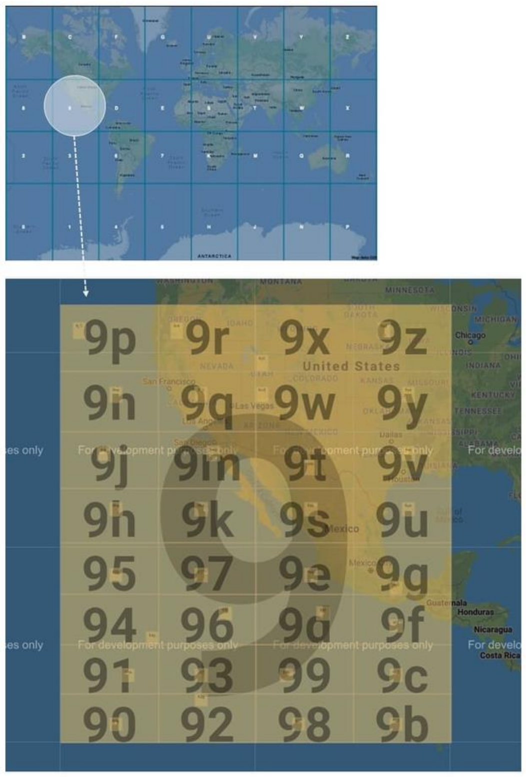


Figure 1: World Map with 2 Layers of Geohash [9]

moment. Given that each cell of the grid has a unique ID, the correctness of this approach is guaranteed. Also, it is possible to analyze moving ships because ships will always have the same cell ID as long as they maintain their relative distance even when they are moving.

Besides, another significant advantage of Geohash is the high efficiency of the algorithm in comparison to conventional queries such as selecting all ship pairs and comparing their geographic coordinates at a given moment. According to research, [9], because of the *Z-order curve* data structure, a task of encoding geographic features and querying those geographic features using the Geohash filter instead of conventional queries based on latitude and longitude can result in a 75 times performance improvement. The improvement in performance can be even larger when more complex queries are concerned. Because the scale of the data engineering for this project is relatively large, good efficiency of the algorithm is crucial.

However, one drawback of the Geohash is that the possible scales for gridding are predetermined and unchangeable by the nature of the algorithm. Moreover, detecting STS meetings is more complex than detecting ships' locations at a moment as the former requires comparisons for ships' cell IDs not just for a moment but for an extended period of time. Furthermore, the duration and the range for validating STS meetings need to be specified. Therefore, a clearer definition of STS meetings is required, and a more comprehensive query needs to be built on top of that. Nonetheless, Geohash is still suitable as the basic tool for analyzing STS meetings due to its advantages in feasibility, efficiency, and scalability. As a result, a python module [14] which provides functions for decoding and encoding Geohash is introduced for this project.

3.2 Definition for STS Meetings

STS meeting problem belongs to the subquestion of ship encounter problems. In previous studies, researchers have used a variety of methods to define the ship encounter problem. In the study by W.H. van Iperen [7], the encounter problem is summarized by Close Point of Approach (CPA) and Time Close Point of Approach (TCPA), which indicates not only spatial but also temporal proximity of two vessels Zheng Liu has proposed a spatial logical approach to analyze the encounter circumstance and project the encountered problem into a geographic convex domain based on Descartes' idea [8]. In research conducted on ship transshipment [11], the rule-based definition was presented. According to the authors, the vessel encounter will be considered to occur when a fishing vessel and a transshipment vessel stay within a range of 500m for at least 2 hours, while moving at or below 2 knots and at least 10km away from a coastal line.

However, despite that the research above helped us to understand the STS meeting problem more clearly, they also faced the dilemma and limitations. For example, none of them considered the multi-ship encounter problem, and the usage scenario is limited (e.g., only the cargo transfer problem is considered). In STS meetings, the multi-ship meeting problem and generality of the meeting(i.e., the meeting can be applied to any type of vessel) should be taken into consideration. Therefore, inspired by the previous research and the Geohash algorithm, we developed our definition for STS meetings which is our definition for STS meetings is: STS meeting takes place when two or more ships encounter in the identical 6th Geohash cell ($1500m * 600m$) and duration of the current meeting should last at least 30 minutes.

4. PROJECT SETUP

This section discusses the design and implementation of the project. An initial data investigation is outlined as well as an ETL data engineering pipeline for the project. Furthermore, the data extraction, transformation, and visualization are described and motivated in thorough details.

4.1 Initial Data Investigation

The original dataset is stored in the format of TXT files. After roughly evaluating the data size and potential workload, we plan to restrict the amount of AIS data involved in this project to one month, namely July 2016, and the focused sea area is restricted to the North Atlantic Ocean. The reason for this choice is that the data for July 2016 is relatively complete, and the month of July conventionally is one of the busiest months for maritime transportation [18]. Therefore we assume that the density of AIS messages in July is higher than others which might provide more information to perform STS meeting detection. By scanning through the target dataset, there are about 3.5 billion raw AIS messages in NMEA 0183 standard[2]. The size of the dataset is approximately 160GB. Typically, AIS messages consist of several fields as shown in table 1.

Unprocessed AIS data is very dirty and unreliable by its nature. For example, the condition of the AIS receivers, the human interventions, and even the weather on the sea can impact the quality of AIS data [2]. Thus, it is normal for AIS data to be noisy and inconsistent. That is, careful data wrangling and feature engineering are needed. For the initial data exploration, Spark is used to enable parallelism for data analysis. Notably, only a small subset of the original dataset, namely the AIS data for 1 hour which makes up about 0.14% of the target data, is used to do the initial data investigation given the large size of the target data. In summary, there are about 5,000,000 raw AIS messages in the experiment subset. The AIS messages are loaded into Spark RDD and decoded using *pyais* which is an open-source AIS decoder [15]. Subsequently, about 8% of the messages cannot get decoded and are discarded due to errors in the raw AIS data. After decoding, the AIS messages are converted into a Spark dataframe.

Field	Description	Example
Field1	Communication data source	!AIVDM
Field2	Number of segments in current message	1
Field3	Id of segmented message	1
Field4	Id of multi-segments data	1
Field5	Radio channel code	A
Field6	ASI payload	177KQJ5000'K¿RA1w
Field7	Padding of payload	2*55

Table 1: Fields for AIS messages

One interesting characteristic of the dataset that is worth mentioning is the way data encodes its time feature. Unlike other modern systems, the AIS system naturally does not encode complete timestamps in its data. Instead, the given AIS dataset uses the path names to represent the time feature of the data. For example, the data in the file under the path of prefix:/2016/07/01/08-30 are entitled to a timestamp of July 1st, 2016, at 8:30 a.m. As a result, one necessary step in the data extraction pipeline is to parse the time feature for the selected AIS messages according to the original path names and encode complete timestamps to the corresponding AIS data.

There are in total 27 basic message types that separate the purpose of AIS messages. Notably, different types of messages entail different schemas of the conveyed AIS data. For example, a type 1 message can greatly differ from a type 4 message in terms of data features. Thus, the initial dataframe containing all types of AIS messages has a large variety of features. In fact, there are more than 100 features in the dataframe. However, the dataframe is quite sparse because features that are not mutually contained across different types of messages have null values. This problem needs to be tackled as it uses excessive memory and slows down the pipeline. Besides, not all message types are relevant to the topic of this project. For example, a type 4 message is a special type of message which is sent by AIS base stations instead of ships [19]. So, it is rational to exclude all irrelevant types of messages such as the type 4 messages as they contain non-relevant information for detecting STS meetings. Consequently, only messages with types 1, 2, 3, 5, 18, 19, 24, and 27 are selected and kept because they contain static and dynamic data for ships which might be useful in

[2]`https://en.wikipedia.org/wiki/NMEA_0183`

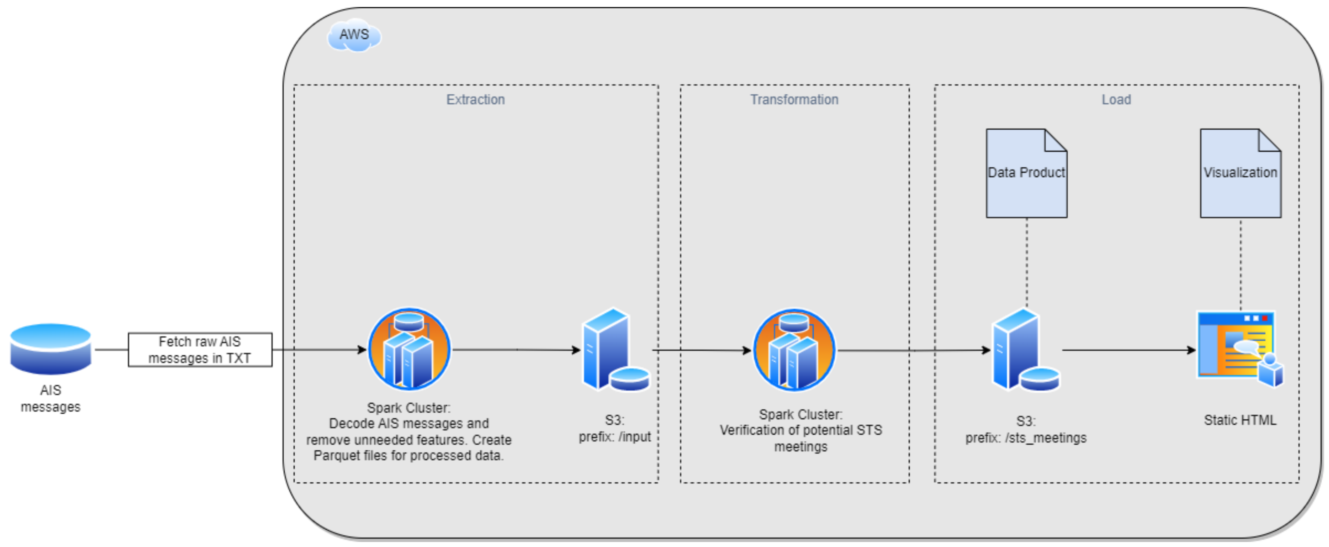

AWS
Extraction
Transformation
Load
AIS messages
Fetch raw AIS messages in TXT
Spark Cluster: Decode AIS messages and remove unneeded features. Create Parquet files for processed data.
S3: prefix: /input
Spark Cluster: Verification of potential STS meetings
Data Product
Visualization
S3: prefix: /sts_meetings
Static HTML

Figure 2: Data pipeline for the project

terms of detecting STS meetings [19].

After removing non-relevant types of messages and the corresponding features, the number of the remaining messages was reduced by approximately 15%, and the number of the remaining features was reduced by approximately 50%. However, the reduced dataframe is still sparse and contains meaningless information because the schema of AIS data varies greatly from message types and not all features remaining are relevant to the task. For example, the RAIM-flag is a remaining feature which indicates whether the RAIM (receiver autonomous integrity monitoring) device is in use. This information does not provide any insight in terms of detecting STS meetings. In order to further reduce the size and increase the relevance of the dataframe to the task, feature engineering is carried out to select and generate the features that are potentially meaningful to the task. As a result, the schema of the dataframe with the desired features is shown as follows:

```
Struct{
    mmsi: long
    msg_type: long
    imo: long
    callsign: string
    status: float
    shipname: string
    ship_type: long
    lon: float
    lat: float
    to_bow: long
    to_stern: long
    to_port: long
    to_starboard: long
    heading: long
    course: float
    speed: float
    draught: float
    destination: string
    radio: long
    eta: string
    timestamp: string
}
```

Note that the 'eta' (estimated time of arrival) and the 'timestamp' are generated based on other features. Essentially, 'mmsi' (ship ID), 'lon' (longitude), 'lat' (latitude), and 'timestamp' are necessary for identifying STS meetings given the approach discussed in the related work. Therefore, messages without these features need to be excluded. In other words, these 4 features are not nullable. Conversely, other features are nullable since they are only used to provide extra information about the ships.

4.2 Data Extraction

An ETL data engineering pipeline is provided as shown in Figure 2. The original AIS data is acquired from the AIS database in the DBFS. Specifically, the data extraction pipeline reads 1 hour's AIS data (60 files) as a batch into a Spark dataframe. First, the corresponding files' names for the data are encoded into the dataframe using the function *input_file_name()* from *pyspark.sql.functions*. As mentioned before, the file path will be used to parse and encode complete time features for the data. Then, the data is loaded into Spark RDD and decoded in a paralleled manner using the *rdd.map()* operation and the *pyais* AIS decoder [15]. After that, the pipeline selects the desired features, the desired message types, and the desired sea area as discussed in the initial data investigation section. Next, the 'eta' and 'timestamp' are parsed and encoded into the dataframe in a paralleled manner by utilizing again the *rdd.map()* operation. Finally, the features used to generate 'eta' and 'timestamp' are removed, and the dataframe is properly pruned into the desired schema as illustrated in the initial data investigation section. As a result, the processed data is written out in parquet files of days (e.g., 2016-07-01) and stored in the directory of prefix:/input. Upon completion of the data extraction, approximately 160GB of the AIS data which contains about 3.5 billion raw AIS messages is decoded, pruned, and extracted into the desired format of input files. The

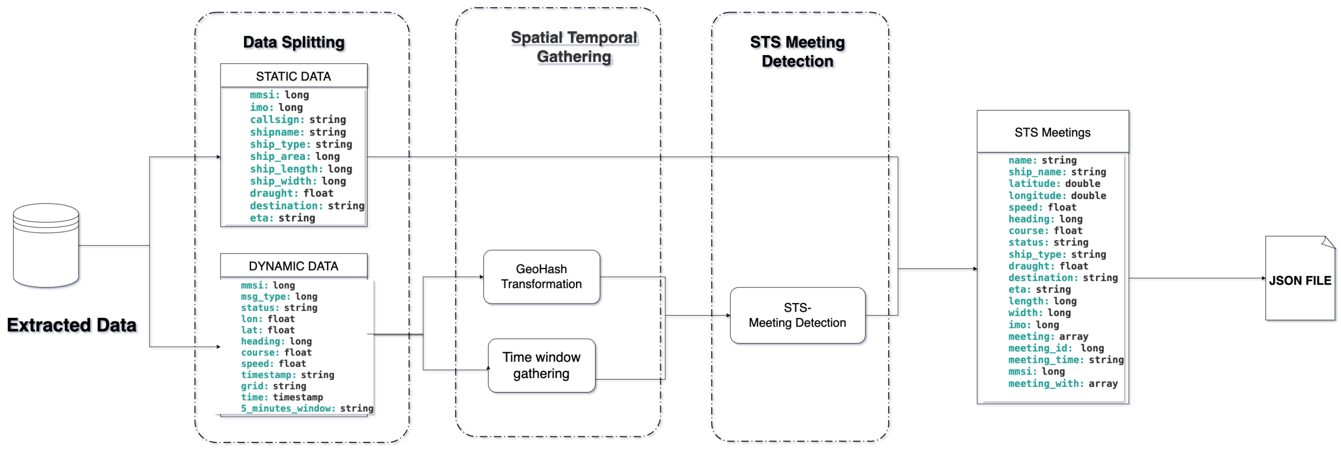


Figure 3: ETL pipeline for transformation

extracted data is approximately 25GB large in size and contains roughly 2 billion decoded AIS messages in the proposed schema.

4.3 Data Transformation

As soon as the data extraction is completed, the data transformation is carried out. In this subsection, a more detailed data transformation pipeline is introduced as presented in Figure 3. After transformation, the data product will be generated in JSON format for final visualization. The following parts provide a detailed description of these data transformation steps.

4.3.1 Data splitting and wrangling

As mentioned in the initial data investigation section, AIS data can be divided into dynamic parts and static parts. That is, two tables are generated for the 2 types of messages. To be noticed, both types of messages contain numerical code fields (e.g., ship status and ship type), and each code represents a unique piece of metadata associated with the vessels' activities and characteristics (i.e., 0 represents using the engine in the status field). Therefore, for each numerical code field, we use dict-mapping to convert them into string representations. However, many missing values are detected in the data due to the poor quality of AIS data [5]. Consequently, by using PySpark function *fillna()*, we replaced the missing string values with "Unknown" and fill the missing numerical values with 0.

4.3.2 Spatial-Temporal gathering

According to the proposed definition of the STS meeting, it is crucial to obtain time and geographic information of ships. Previously, We introduced CPA methods, which are able to capture information about two ships in terms of the time to encounter and the distance in between. However, under the context of large-scale computation, it is extremely expensive because the algorithm will need to traverse through every possible ship pair and compute CPA variables for them. Therefore, in order to reduce the complexity of computation, we proposed a time-space aggregation method as a result of all the factors above combined. The steps are described as follows.

- The ships are aggregated in time windows (i.e., the activities and locations of ships are partitioned by a specific time gap) instead of the unit time of AIS data (i.e., second).

- Using Geohash, the latitude and longitude are encoded into Geohash strings to represent the location for a certain area.

In terms of selecting parameters, the grid cell range is limited due to the Geohash restrictions. For example, the range covered by the grid cell with a precision of 6 is $1500m*600m$. However, when the precision of the Geohash algorithm increased to 7, the range covered by the grid cell changes to $150m * 150m$. Based on our study on ships encounters [11], the proper distance for identifying ships' encounters is 500m in the transshipment situation. Thus, considering the generality of the current task (i.e, all types and situations of vessels should be included), the Geohash grid with a precision of 7 ($1500m * 600m$) is selected as our gathering scope. As for the time windows, we choose 5 minutes window to gather AIS messages accordingly. By doing so, the computation steps needed will be largely reduced while maintaining a reasonable precision for the time feature of the AIS data.

4.3.3 STS meeting detection - the grid approach

The STS meeting detection is the most important part of the whole data transformation stage. Our algorithm relies on the condition that two or more ships should be in an identical spatial-temporal gathering. Namely, for the current time window, the vessels in the same grid cell will be viewed as encounters. However, STS meeting detection focuses on the encounter behavior for an extended period of time. Thus, only the encounters which have a duration of at least 30 minutes will be selected as potential STS meetings. More specifically, the algorithm follows the steps below:

- Apply the Geohash on the map

- Partition the data into 5 minutes windows, for every window:

    1. Search and mark ships' encounters (vessels that are in the same cell)

2. Check whether the same encounters are detected in later windows
3. Select potential meetings (the duration between the first encounter and the last encounter is greater than 30 minutes)

- The table of meeting result is generated

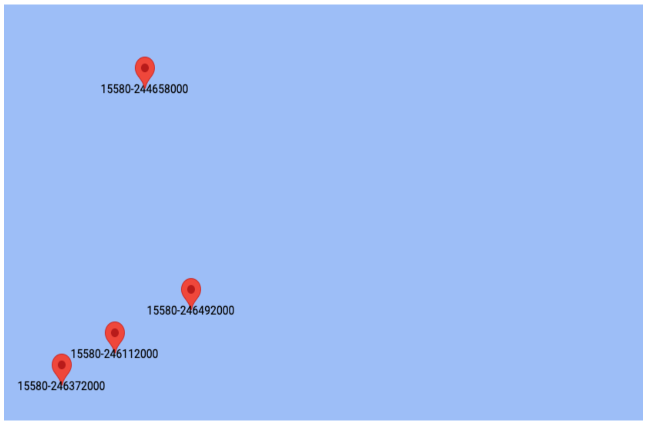


Figure 4: STS multi-ships detection

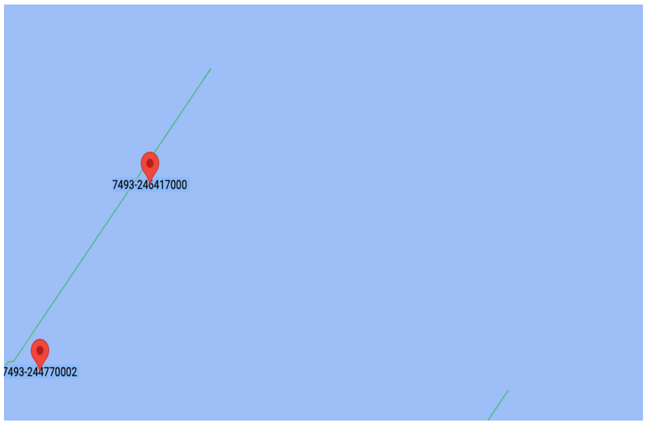


Figure 5: STS meeting in the normal situation

The algorithm described above is balanced in terms of efficiency and capability. Besides, the multi-ship meeting can also be recognized as it shows in Figure 4. To be noticed, the red markers in the map represent the ships involved in a multi-ship meeting with the meeting IDs joined with the MMSI numbers of the ships as the annotations.

Nevertheless, it has to be acknowledged that there are also some shortcomings in the current approach. First and foremost, it can only detect the first meeting. Namely, the meeting will not be counted again if two or more meeting vessels encounter a new approaching vessel in the next time window. For example, there are three ships: ship A, ship B, and ship C. The STS meeting of Ship A and ship B is detected at 00:00. However, while ship C is approaching ship A and ship B at 00:05, the new STS meetings are recognized. In the new STS meetings, the original STS meeting of A and B will not be counted. Instead, it turns out to be a meeting with newcomers respectively. Thus, there will be three meetings in the same area, they can be seen as follows:

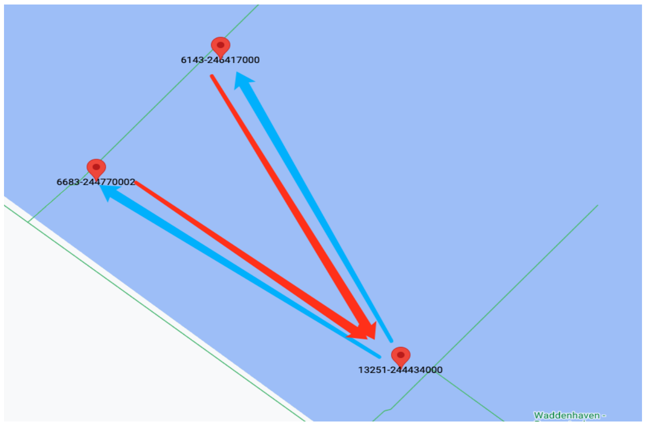


Figure 6: Meetings change when facing new encounter

1. For A, the new STS meeting contains A, C
2. For B, the new STS meeting contains B, C
3. For C, the new STS meeting contains C, B, A

Therefore the ID of the meeting will change accordingly. As shown in Figure 5 and Figure 6, for the new approaching vessel depicted in the bottom right of Figure 6, there is a multi-ship meeting for 3 ships (including itself). However, for two ships that have previously met with each other, the STS meeting will be verified as two separate ship meetings with the newcomer respectively. This problem is the primary limitation of the current algorithm. Besides, due to the limitation of the Geohash algorithm, the STS meeting cannot be detected correctly in some circumstances. For example, when two or more vessels are geographically close (e.g., the distance between ships is less than 5 meters) to each other, which should be recognized as an STS meeting. However, such STS meetings might not be detected when the ships are close to the margins of the grid cells (i.e., the algorithm splits adjacent ships into different cells).

4.3.4 Output data product

After the STS detection algorithm is carried out, at the end of the transformation pipeline, the meeting query table will be joined with static messages of ships to augment the STS meeting. After that, the final data will be partitioned by time windows (i.e., every 5 minutes) and output in the format of JSON. The fields of final output JSON data can be seen as follows:

```
Struct{
    name: string
    ship_name: string
    latitude: double
    longitude: double
    speed: float
    heading: float
    course: float
    status: string
    ship_type: string
    draught: float
    destination: string
    eta: string
    length: long
    width: long
    imo: long
    meeting: array
    meeting_id: long
    meeting_time: string
    mmsi:long
    meeting_with: array

}
```

4.4 Visualization

The visualization of this project is based on an interactive web page application. This application utilizes ReactJS, an efficient JavaScript framework for creating user interfaces based on the structure of shipwreck [4]. We built an interactive and responsive web application that is applied to both desktop and mobile devices. The final data product of our project is stored in Databricks's DBFS Service in the format of JSON which is a standard text-based format for representing structured data based on JavaScript object syntax. This website mainly uses Google Maps JavaScript API to render the whole world map and draws each ship on a specific position according to our data product. We utilize the common JavaScript pack tool "web-pack" to bundle this project into HTML, JavaScript, and CSS files which are stored in Amazon's server provided since only static HTML is required. This website contains one homepage with two scales and is structured as follows:

4.4.1 *Homepage*

There is a search button on the top right, a date-time picker on the top left, and a google static map on the homepage of this website. With this search button, we can search the MMSI numbers of ships. The date time picker component is utilized to select a specific time to show the STS meetings 5-minutes-wise. Moreover, there are a plus button and a minus button on the bottom right which can be utilized to zoom in and out, the same function can be done by using the mouse scroll wheel.

4.4.2 *North Atlantic scale*

The one depicted in Figure 7 is a global or high-level presentation of the data product. In this case, we use color-themed circles to indicate typical STS meetings within a specific range. Circles mean clusters of ships in that area. In addition, to make the colored circles more intuitive, the quantity of STS meetings will be denoted on them. In short, the purpose of this high-level presentation is to provide an overview of the data product.

4.4.3 *Ships scale*

Another presentation as shown in Figure 8 focuses on a smaller scale that zooms into the area in which we can see specific meetings and ships involved clearly. In such a scenario, the figure depicts ships in a small range that may host SIS meetings. We utilized Google map component markers

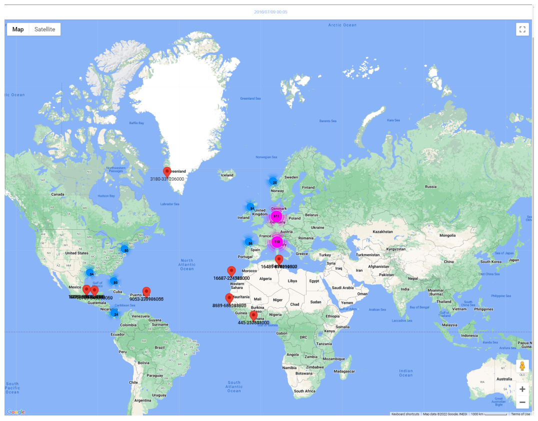


Figure 7: North Atlantic scale

to symbolize ships. That is, whenever STS meetings take place, the ships will be depicted with markers. When the markers are clicked, a card with three types of information will be displayed. Namely, the first type is the basic ship data including ship name, MMSI number, speed, location, destination, etc. Then, the second type is the basic STS meeting data such as the meeting time, the MMSI numbers of ships involved, etc. Lastly, other detailed information about the ship and the meeting such as the ship type and the ship's image will be displayed when applicable. To make STS meetings more intuitive, we tried to draw red circles to cover ships involved in meetings (The red circle's center is in the middle of the line connecting ships and the radius of it is half the distance of that line) but failed. The reason for that will be elaborated in the discussion section. In conclusion, this low-level presentation demonstrates situations where specific STS meetings are augmented with extra information when a smaller focus range is interesting.

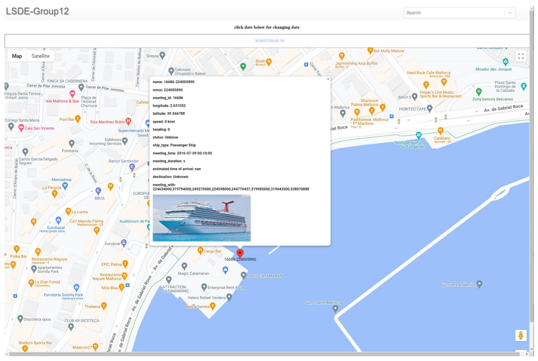


Figure 8: Ships scale

4.5 Project Timeline

The proposed timeline for the project is shown in Figure 9. Generally, the deadlines are protected. However, we underestimated the difficulties in modeling and visualization and spent longer time than we expected. We weren't able to solve every problem we encountered within the time limitation. Thus, we made several trade-offs in order to complete the project on time. The trade-offs will be elaborated in the discussion section.

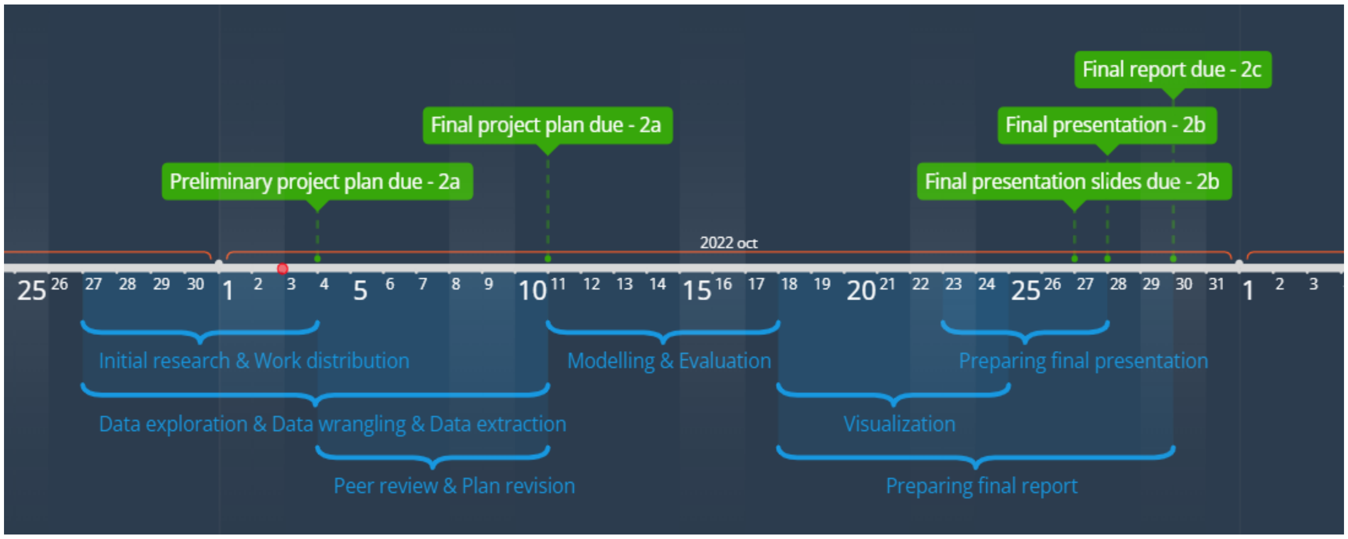

Final report due - 2c
Final project plan due - 2a
Final presentation - 2b
Preliminary project plan due - 2a
Final presentation slides due - 2b
2022 oct
25 26 27 28 29 30 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 1 2 3
Initial research & Work distribution
Modelling & Evaluation
Preparing final presentation
Data exploration & Data wrangling & Data extraction
Visualization
Peer review & Plan revision
Preparing final report

Figure 9: Proposed project timeline

5. DISCUSSION

5.1 Visualization

We analyzed the distribution of the data product and discussed what was the better time scale for the visualization. In the beginning, we thought that the hour-wise precision is a proper time scale to visualize meetings since if we choose a day-wise presentation, the granularity of time is too coarse, and it results in overlaps of ships in the same position. That is, some of the ships tend to stay in the same location for a relatively long time (e.g., moored ships in ports). However, they might be detected in meetings with other ships multiple times in different time windows throughout a day. Conversely, if we choose a second-wise precision, the granularity of time becomes too delicate to present enough information under the context of STS meetings. Thus, we divided the data file for 1 day into 24 data files for 24 hours. The date time picker on the web page can be used to select a time in hours to see the detected meetings in that specific hour. However, after applying the hour-wise time scale, we found that the problem of overlapping ships still existed. As a result, we decided to further refine the time scale for the visualization to 5 minutes, which is consistent with the partition time window of the data transformation pipeline. Finally, the problem of overlapping ships is solved.

Moreover, we wanted to implement a function that can show a ship's image on the detailed information card by searching for its MMSI number on some websites. We did find an appropriate website called FleetMon [3] which can help us search ships' images with their MMSI or IMO number, but it charges for the usage of its API at an expensive price that we cannot afford. As a result, we decided to download images of different ships' types in advance and display them according to the ship type data to make markers more intuitive. But this approach only works as an alternate illustration to show that we want to add a visual aid in the information card.

Additionally, we had a discussion about how to forward meetings instead of ships. For example, we tried to use extra annotations (e.g., circles) to highlight encounter situations.

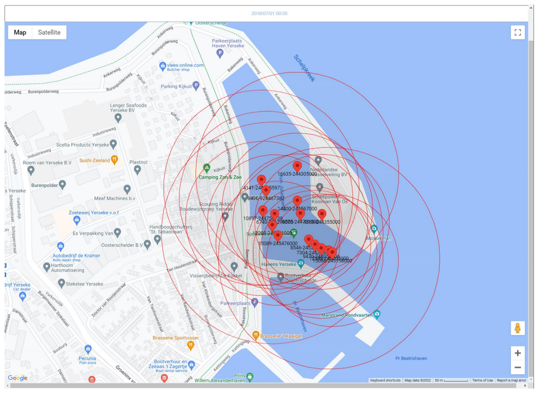


Figure 10: Meetings contain too many circles

However, due to the limitation of the multi-ship detection algorithm, STS meetings that involve a number of ships in a small area will generate many overlap circles as shown in Figure 10, which will reduce the performance and intelligibility of our visualization. As a result, we disposed of the method of circles to forward STS meetings. Instead, we added an array attribute called "meeting_with" which contains the IDs of ships that are involved in the STS meeting for the target ship. With that, it is easier to observe STS meetings that involve more than two ships.

5.2 Data Product

5.2.1 *Missing data*

When evaluating our final data product, we found that occasionally there were missing data in the visualization. More precisely, for STS meetings, there should be at least two or more vessels presenting on the map. However, in some circumstances, only one vessel is displayed, and there are no other ships around. As shown in Figure 11, the current ship should have been meeting with MMSI '211693490', which cannot be observed on our visualization map.

This problem seriously affected the correctness of the data product and the performance of the visualization. To solve this problem, we first checked the data transformation pipeline and the visualization component in detail but end up with nothing. Subsequently, we found there was a mismatch between the generated meeting table and the output JSON files. After we looked into this problem, we speculated that it might be caused by the incorrect use of the *coalesce()* and *repartiton()* operations of Spark while writing the data into JSON files. According to the user manual of Spark, [17], these operations will shuffle the data across many partitions to minimize repartition as much as possible. Namely, it is possible that the data was shuffled to another time partition instead of the partition we specified. We tried to fix this technical problem by eliminating the use of *coalesce()* and *repartiton()*. However, this attempt did not solve the problem either. We have to acknowledge that we are not able to solve this technical problem within the time limit of this project, and it is a primary drawback of the current data product.

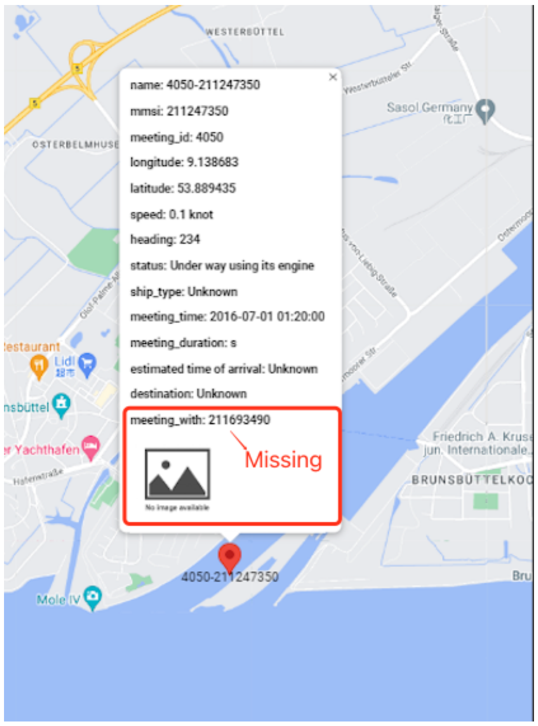

name: 4050-211247350
mmsi: 211247350
meeting_id: 4050
longitude: 9.138683
latitude: 53.889435
speed: 0.1 knot
heading: 234
status: Under way using its engine
ship_type: Unknown
meeting_time: 2016-07-01 01:20:00
meeting_duration: s
estimated time of arrival: Unknown
destination: Unknown
meeting_with: 211693490

Figure 11: Circumstance of missing data

5.2.2 *Uncertainty in continuity*

Due to the nature of AIS data, the continuity of the data cannot be guaranteed. In other words, there is an unpredictable interval between AIS messages sent by the same ship. Also, since the algorithm can only detect the first meeting, the potential secondary meeting cannot be identified. For example, if ship A meets ship B in the morning before sailing, and then meets ship B again when returning to port in the evening. In this case, there should be two meetings, but only one will be recognized because it is not possible for the current algorithm to determine whether the meeting for A and B is continuous. This is due to the characteristics of the AIS data and the limitation of our algorithm. It also negatively affects the final result of our data product.

5.2.3 *Data distribution*

The final data product incorporates approximately 700,000 ships in STS meeting in July 2016, and the size of it is roughly 260MB. To be noticed, with regard to data distribution, most detected vessels aggregate in the first hour of the day. For example, the data distribution of 2016-07-01 is shown in Figure12. It is obvious that a large amount of data aggregates in the first hour of the day. This is because the extracted data are divided into parquet files of days, and each day starts with a large number of anchored ships near the ports which will be recognized as STS meetings by the algorithm. However, the skewness of the dataset affects the performance of Spark during the partitioning, which is considered a shortage of the current data pipeline.

6. CONCLUSION

To sum up, we build a data pipeline that is capable of analyzing maritime data, verifying potential STS meetings, and providing extra information about the meetings. Moreover, we set up a visualization website to showcase the data product of the pipeline, namely the detected STS meetings at sea.

First of all, the concept of STS meetings is introduced, and the reasons why detecting potential STS meetings are interesting and necessary are motivated. To make the problem more intuitive, the research questions of this project are put forward. Moreover, related work which contains tools and methodologies (i.e., the Geohash algorithm) for the verification of potential STS meetings as well as the quantified definition of STS meetings are discussed.

Based on the related work, a grid approach that is capable of detecting potential STS meetings using AIS data is developed. An ETL data engineering pipeline is provided and discussed in detail as well as the means for visualizing the final data product of the pipeline. Consequently, the potential STS meetings that happened in the North Atlantic Ocean in July 2016 were detected and visualized.

In the end, a discussion session that evaluates the project setup and the problems encountered is carried out. Particularly, the decision-making and critical thinking throughout the project implementation are elaborated.

6.1 Answers to the Research Questions

In conclusion, answers to our research questions contain two perspectives. From the topic perspective, firstly, the explicit and quantified definition for potential STS meetings is proposed based on the previous research. Namely, an STS meeting is identified when two or more ships stay in the identical Geohash cell with a precision of 6 (1500m * 600m) for at least 30 minutes.

Secondly, when it comes to the characteristics of AIS data and its feature, we observe that unprocessed AIS data is very dirty and unreliable. After carefully examining the experiment subset of the AIS data and related work for detecting potential STS meetings, the required features in AIS data for detecting potential STS meetings are: ‘mmsi’ (ship ID), ‘lon’ (longitude), ‘lat’ (latitude), and ‘timestamp’.

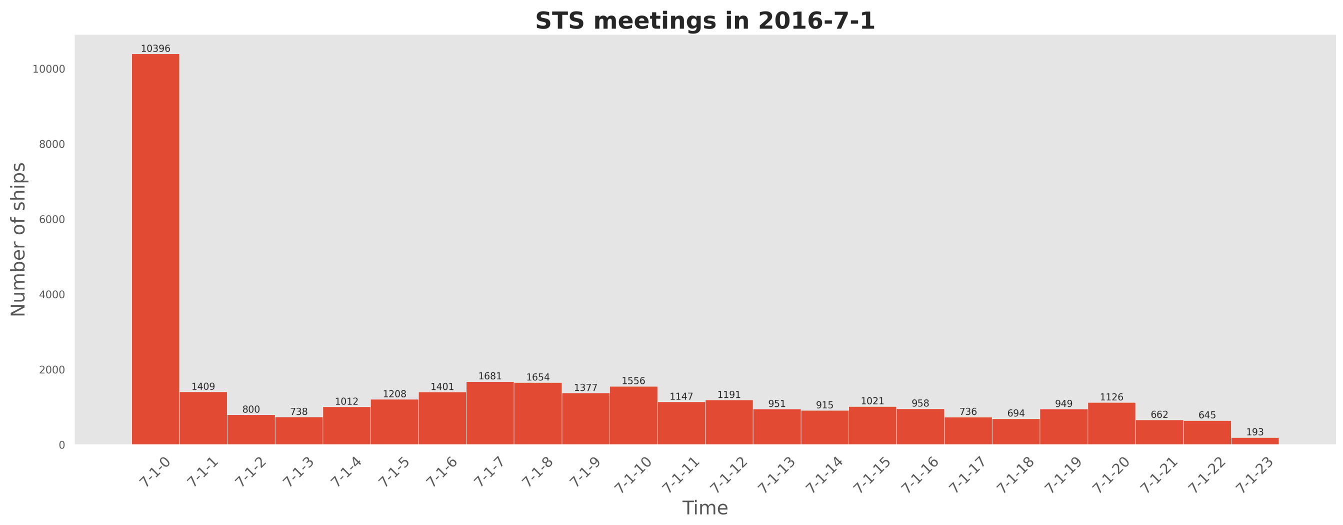


Figure 12: Data distribution in one day

Thirdly, to build a data engineering pipeline that is capable of analyzing large-scale AIS data and identifying potential STS meetings, Spark and the pyais AIS decoder are used to decode and process AIS data in a paralleled manner. A grid approach that detects potential STS meetings is proposed based on the Geohash algorithm and the definition of the STS meeting. In the end, ReactJS is utilized to visualize detected STS meetings in a static website based on the Google Maps JavaScript API.

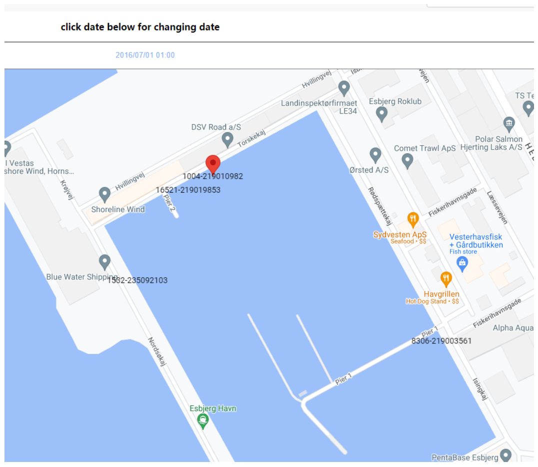
Finally, the data engineering pipeline contains three stages: extraction, transformation, and loading. The extraction stage decodes, prunes, and filters raw AIS data. Then, in the transformation stage, the proposed grid approach is applied to the extracted data to detect potential STS meetings. In the end, the loading stage makes use of static HTML to visualize the final data product.

For the technical perspective, the time that the data transformation pipeline takes to process the extracted data which is approximately 25GB in size is about 3 hours and 47 minutes. Because we cannot find an established baseline concerning the same or similar task in the existing literature, it is hard to give a meaningful conclusion about the performance of the data transformation pipeline. However, given the observation about the tools that the pipeline builds on, we claim that the performance of the pipeline is reasonable in comparison to a naive traversing query, and the feasibility for the pipeline is relatively good as well as the scalability due to the characteristics of the Geohash algorithm. However, the pipeline also has some drawbacks such as only being able to detect the first meeting and being invalid when it comes to the margin cases.

6.2 Limitations and Future Work

6.2.1 Visualization

The future work of visualization can concentrate on STS meetings' tendencies. For example, it will be more intuitive to add a timeline bar on the bottom of the Google map to show the tendency of STS meetings happening. Since the current version of this web application only shows STS meetings in a 5 minutes duration, it is not intuitive to make a judgment or draw a conclusion about the tendency of STS meetings. If a timeline bar was added, we can drag the bar to see if the meetings are increasing or decreasing in a specific area or worldwide as time moves.


click date below for changing date

2016/07/01 01:00

Figure 13: Ships markers not showing

We also found that this website works more smoothly on Chrome Browser than other browsers since static map API and Chrome are both developed by Google. Consequently, We recommend users view this website application on Chrome Browser. Moreover, when testing on different browsers, we found that sometimes there are some markers missing as shown in Figure 13, which we think is a problem with the marker component of Google static map JavaScript API. It happens occasionally on different browsers and devices. However, the chance for this error to appear is relatively low, and the exact reason for it remains unknown to us. Therefore, we decided to leave this problem as a future work that needs to be done.

Furthermore, as is mentioned in the discussion part, we disposed of red circles to forward STS meetings. However, we still want to find another appropriate way to make meetings more intuitive since the topic of the project is STS meetings. In future work, one possible solution is to color the ships that are in the same meeting with the same color. In that case, how to color ships and how many colors are needed are the new challenges that await.

Last, we added a search input component on the top right of the map and it supports searching MMSI numbers of markers drawn on the map. However, due to the lack of development time, we did not implement the function that the map can zoom in on the location of the target ship automatically when users select one MMSI number. We plan to implement this function in future work as it enriches the functionality of our visualization.

6.2.2 Data product

In the data transformation section, the limitations of the current STS detection method are mentioned. In future work, we will improve the STS meeting detection algorithm in terms of accuracy and feasibility. Firstly, the accuracy of the current algorithm is limited because of the margin detection problem with the Geohash algorithm. It can be improved by introducing another distance measurement (e.g, directly calculating the distance between vessels in the margin area), which should be used as an aid in STS meeting detection. Besides, more features should be considered in the algorithm. In the current STS meeting recognition, many features such as velocity and heading which are vital characteristics during voyaging are not taken into account. In fact, most of the detected STS meetings involve anchored vessels that either stay in or near ports. After combining the voyaging features in the algorithm, the accuracy of the algorithm is expected to be improved. However, performances in the efficiency and accuracy of the algorithm need to be re-balanced as well.

7. REFERENCES

[1] V. Airfleet. What are ship-to-ship meetings?, Nov 2021.

[2] T. Emmens, C. Amrit, A. Abdi, and M. Ghosh. The promises and perils of automatic identification system data. *Expert Systems with Applications*, 178:114975, 2021.

[3] FleetMon. Live ais vessel tracker with ship and port database.

[4] L. Harpf. shipwrecks.

[5] W. He, J. Lei, X. Chu, S. Xie, C. Zhong, and Z. Li. A visual analysis approach to understand and explore quality problems of ais data. *Journal of Marine Science and Engineering*, 9(2), 2021.

[6] IMO. Ais transponders, 2022.

[7] W. Iperen. Classifying ship encounters to monitor traffic safety on the north sea from ais data. *TransNav: International Journal on Marine Navigation and Safety of Sea Transportation*, 9(1), 2015.

[8] Z. Liu, Y. Li, S. Dong, and Z. Zhang. Spatial logical relationship analysis model of ship encounter space. *Ocean Engineering*, 239:109912, 2021.

[9] Q. G. P. Ltd. Quadrant geohash algorithms, 2022.

[10] N. A. Miller, A. Roan, T. Hochberg, J. Amos, and D. A. Kroodsma. Identifying global patterns of transshipment behavior. *Frontiers in Marine Science*, 5, 2018.

[11] N. A. Miller, A. Roan, T. Hochberg, J. Amos, and D. A. Kroodsma. Identifying global patterns of transshipment behavior. *Frontiers in Marine Science*, 5:240, 2018.

[12] G. M. Morton. A computer oriented geodetic data base and a new technique in file sequencing. 1966.

[13] G. Niemeyer. Geohash. 2008.

[14] L. Norrgard. Geohash, 2015.

[15] L. M. Richter. Pyais, 2022.

[16] SKULD. Ship to ship transfer safety, 2020.

[17] A. Spark. Pyspark.sql.dataframe.coalesce¶, 2022.

[18] J. Thomsen. Busiest season for container lines approaches – but ships are already overflowing, 2021.

[19] United-States-Coast-Guard. Ais messages, 2022.

APPENDIX

A. DISTRIBUTION OF WORK

The Report contribution overview is shown in Table 2, and the project contribution overview is shown in Table 3.

Task	**Person**
Abstract	Yudong Fan
Introduction	Yudong Fan
Research Question	Yudong Fan
Related Work - Geohash	Yudong Fan
Related Work - STS Definiton	Hongyu Wu
Initial Data Investigation	Yudong Fan
Data Extraction	Yudong Fan
STS Meeting Identification	Hongyu Wu
Visualization	Yihong Xi
Project Timeline	Yudong Fan
Discussion - Visualization	Yihong Xi
Discussion - Pipeline and Data Product	Hongyu Wu
Conclusion	Yihong Xi, Hongyu Wu

Table 2: Report contribution overview

Task	**Person**
Brainstorming, Regular Meetings and Discussions	All
Initial Data Exploration	Hongyu Wu, Yudong Fan
Data Extraction	Yudong Fan
STS Meeting Identification	Hongyu Wu
Visualization	Yihong Xi

Table 3: Project contribution overview

B. VISUALIZATION ON DBFS

Our Visualization can be found on DBFS at dbfs:/mnt/lsde/group12/lsde-group12.zip, and there is a README file at dbfs:/mnt/lsde/group12/lsde-group12.md showing how to set up this visualization application.