Norwegian University of
Science and Technology

An Advanced Method for Detecting Exceptional Vessel Encounters in Open Waters from High Resolution AIS Data

Haakon Akse Nordkvist

Marine Technology
Submission date: June 2018
Supervisor: Stein Haugen, IMT

Norwegian University of Science and Technology
Department of Marine Technology

An Advanced Method for Detecting Exceptional Vessel Encounters in Open Waters from High Resolution AIS Data

Haakon Akse Nordkvist

June 2018

MASTER THESIS

Department of Marine Technology (IMT)
Norwegian University of Science and Technology

Supervisor: Stein Haugen
Company: Safetec Nordic AS

Preface

This masters thesis is the culmination of my M.Sc. at the Department of Marin Technology with a specialization in Marine Systems Design. The thesis corresponds to 30 ECTs and has been written during the spring term of 2018 at Norwegian University of Science and Technology. It has been a collaboration with Safetec Nordic whom has been indispensable by sharing raw data and as a sparing partner during the thesis work.

The thesis proposes a new framework for risk assessment in open waters based on pairwise encounters from historical data. The idea came up during my project thesis when it became apparent that the current risk assessment models are not utilizing the full potential of AIS data.

Trondheim, June 18, 2018

Haakon Nordkvist

Haakon Akse Nordkvist

Acknowledgment

The work performed in this thesis has been in cooperation with Safetec Nordic, their interest have been to produce the best result possible and have room for further development.

At Safetec Nordic I would like to give a special thanks to Martin Hassel, Peter Ellevseth and Asbjørn Lein Aalberg. They have shown goodwill and provided support and invaluable guidance whenever needed.

I would further like to thank the following persons for help and support during my thesis:

My supervisor Prof. Stein Haugen for guidance throughout the the spring semester.

Prof. Bjørn Egil Asbjørnslett for being my supervisor during my project thesis and interesting lectures on maritime risk leading to my interest in the field.

Lastly, Benedicte, my family and my office mates that provided support during this period.

H.A.N.

Executive Summary

This thesis is based on the assumption that AIS is underutilized for risk assessment models. The proposed model estimates the frequency of exceptional encounters between ship pairs which can give an indication of risk. An advantage of the proposed model is the possibility of identifying periods with higher frequency of encounters, it can also find frequencies for subsets of flag states, vessel types or both.

The model relies on a ship domain approach to find potential exceptional encounters, which is well established in the navigational risk research community. From this set of encounters rate of turn is estimated before and after closest point of approach to sort out exceptional encounters where the rate of turn is larger. This is a less documented approach for quantifying the presences of risk in an encounter between two vessels. It is based on the rules of the sea where a vessel required to give way must do so early and substantial in the perspective of the other vessel.

The model was applied on an area of Vestfjorden in a four year time period from 2013 to 2016. The model found 707 ship pairs where one or both vessels had their ship domain violated. Using a threshold of 70 deg/min for rate of turn 381 encounters are classified as exceptional.

Normalizing encounters for distance and trips shows that certain groups have far too high frequency. It is apparent that the model has problems handling encounters between two fishing vessels. Further it is discovered that noise in the AIS data are contributing to rate of turn being over estimated resulting in fairly straight tracks having high values. This is particularly evident from a ferry route where a significant number of head-on encounters without course alterations are registered as exceptional. Filtering out these two types of encounters gives a normalized frequency that are similar across the dataset.

It is concluded that the hybrid solution for estimating frequency of exceptional encounters where traditional ship domain theory and evasive maneuver detection combined have potential to become a solid risk assessment tool with further development.

Sammendrag

Denne oppgaven er basert på en antagelse om at AIS er underutnyttet for risikovurderingsmodeller. Den foreslåtte modellen estimerer frekvensen av eksepsjonelle hendelser mellom skipspar som kan gi en indikasjon på risiko. En fordel med den foreslåtte modellen er muligheten til å identifisere perioder med høyere frekvens av hendelser, den kan også finne frekvenser for flaggstater, fartøystyper eller kombinasjon av dem begge.

Modellen bruker en skipsdomene-tilnærming for å finne mulige eksepsjonelle hendelser, som er godt etablert i forskningsmiljøet. For denne gruppen hendelser blir svinghastigheten beregnet før og etter nærmeste passeringspunkt for å sortere ut eksepsjonelle hendelser med høy svinghastighet. Det å bruke svinghastigheten til å kvantifisere risiko mellom to fartøy er en lite dokumentert tilnærming. Den er basert på sjøveisreglene hvor et fartøy med vikeplikt, må tidlig og tydelig endre kurs i forhold til det andre fartøyet om nødvendig. Ut fra dette vil en plutselig og kraftig kursendring i nærhet til skipet være en indikasjon på forhøyet risiko.

Modellen ble anvendt på et område rundt Vestfjorden i en fireårsperiode fra 2013 til 2016. Modellen fant 707 skipspar hvor ett eller begge fartøyenes skipsdomener ble brutt. Med av en grense på 70 grader/min for svinghastighet ble 381 møter klassifisert som eksepsjonelle.

Normalisering av hendelser for avstand og turer viser at enkelte grupper har altfor høy frekvens. Det er tydelig at modellen har problemer med å håndtere hendelser mellom to fiskefartøyer. Videre er det oppdaget at støy i AIS-dataene bidrar til at svinghastighet blir over estimert, noe som resulterer i at selv ganske rette kurser får høye verdier. Dette er spesielt tydelig på en fergerute hvor et betydelig antall skip med motsatte kurser uten kursendringer er registrert som eksepsjonell. Filtrering av de to feilkildene gir en normalisert frekvens som er nokså lik over datasettet.

Det konkluderes med at den foreslåtte modellen for å estimere frekvensen av eksepsjonelle hendelser der tradisjonell skipsdometilnærming og unnamanøverdeteksjon kombinert har potensial til å bli et solid risikovurderingsverktøy med videre utvikling.

Contents

List of Figures

1. Introduction

1.1 Background

The intention for Automatic Identification System(AIS) is to aid safe navigation by broadcasting dynamic and static information on open dedicated radio frequencies. AIS equipment is mandatory for most of the world fleet navigating on the worlds ocean and leads to a large stream of data which is possible to store for later use. It is with this data collection new possibilities opens up. The currently most used risk assessment models relies on methods developed before the introduction of AIS and does not fully utilize the possibilities of the data.

With large amounts of historical AIS data new approaches opens for risk assessment. There is potential for using AIS data directly in order to find real vessel encounters and extrapolate frequency of collisions from those. This can give new insight into individual risk and time dependent risk, and is the main purpose of this thesis.

The knowledge acquired could be useful for among others port state control, national coastal administrations, marine insurers and as training for automated ship navigation. This type of an approach could also possibly detect suspicious activities and document specific cases of under reporting which is well documented. The approach is made to take advantage of large amounts of historical data, but it may be altered to be used as a continuous tool for vessel traffic centers.

1.2 Objectives

The objectives of this thesis are:

1. Propose a suitable method for better utilization of raw AIS data for collision risk assessment

2. Develop a scaleable and efficient model for applying the proposed method and documenting the core mechanisms to level where future students can easily continue the development.

3. Perform a case study for validation of model effectiveness.

4. Document limitations for the current version and recommend future improvements and additions.

1.3 Structure

The remainder of this paper is organized as follows: chapter 2 documents the theory behind the proposed model. Chapter 3 is a thorough documentation of how the model is made and how it works. Chapter 4 is a case study of an area of Vestfjorden in an 4 year period. Chapter 5 is discussion of validity for the model. Chapter 6 is conclusion and recommendations for further work.

2. Theory

2.1 AIS

Automatic Identification System, AIS is a communication system developed to aid safe navigation at sea. The systems operates on four worldwide Channels in the VHF maritime mobile band to exchange navigation data between AIS devices. The data consists of static data such as ship dimensions and MMSI number, dynamic data such as speed and heading and voyage-related data like drought, destination and ETA (IALA 2016).

Static information is entered to the AIS memory unit during installation and will require a password to change the data IALA 2016. Since the static information seldom changes it is transmitted every 6 min (ITU (2014)). The voyage-related data may need to be set manually in the system IMO (2002a), and can lead to more errors in the AIS data (Harati-Mokhtari et al., 2007; Shelmerdine, 2015). Dynamic data will be updated automatically with an interval dependent on the ship speed and rate of turn as seen in table 3.1.

AIS systems are mandatory in the revised chapter V of SOLAS 1974, "Safety of navigation" and applies to all contracting States. Under regulation 19 paragraph 2.4 of SOLAS chapter V, all ships in international traffic above 300 GT, cargo ships not in international voyage above 500GT and passenger ships irrespective of ships are mandated to have AIS system [1] (IMO, 2014). In addition to world wide requirements, EU requires all fishing vessels above 15 meters to have AIS equipment as of 31st of May 2014[2] EU (2016).

[1] In the transition period exemptions are allowed for ships constructed before 1st July 2002, but no later than 31st of December 2004 for international traffic. For ships not engaged in international voyages no later than 1st of July 2008. The administration may have exempted ships from acquiring AIS equipment, if the ships was to be taken out of service within two years of the implementation date.

[2] The requirements were gradually implemented by requiring vessel over 24 meters to have it by 2012, and vessels over 18 meters by 2013.

From the IMO resolution MSC.74(69), AIS should improve the safety of navigating by satisfying the following functional requirements.

1. in a ship-to-ship mode for collision avoidance;
2. as a means for littoral States to obtain information about a ship and its cargo;
3. as a VTS tool, i.e. ship-to-shore (traffic management).

2.2 Maritime Risk Assessment Models

Historically the pioneering work by (Fujii et al., 1970; Fujii, 1974) and (Macduff, 1974) have been utilized for most of the collision risk assessment models Montewka et al. (2010). Models based on this work can generally be written on the form of equation 2.1 and have won their popularity from its robustness and simplicity Montewka et al. (2010).

$$N_{collisions} = N_{candidates} * P_C \tag{2.1}$$

$N_{candidates}$ is the number of geometric collision candidates and P_C is the causation probability of failing to avoid collision in an accident scenario. Pedersen et al. (1995) adopted this method and substituted ship density from (Fujii, 1974) with traffic flow .

The most used predictive maritime risk models are based on Fujii, Macduff and Pedersen where traffic is assumed to follows a few shipping lanes with probabilistic distributions Chen et al. (2017). Among these models are GRACAT(Friis-Hansen and Simonsen, 2002), IWRAP MK II(IALA 2017) and Collide(Vinnem, 2013).

2.3 Detecting near misses and ship encounters

With the introduction of AIS it has become possible to identify dangerous situations between vessels over larger areas than previously possible. This is thanks to the significantly longer reach of the VHF radio transmissions compared to Rader used before AIS (Vinnem, 2013). The methods mentioned in this section are not reliant on predefined traffic distributions in order to produce results like the previous mentioned models. Although most of the methods use ship domain methodology similar to the geometric probabilities, they are different in terms of assessing actual vessel movements instead of modelling movements as a Poisson process(Pedersen et al., 1995).

2.3.1 Analysis of high resolution rate of turn

The available approaches for identifying critical situations are based on subjective zones or numeric fear factors Mestl et al. (2016). The approach suggested by (Mestl et al., 2016) utilizes high resolution AIS data and recorded ROT.

2.3.2 Analyzing ship movements

Iperen (2015) made criteria for classifying ship encounters in order to compare old and new route structures in the Dutch part of the North Sea. The criteria were made from a training sett of 3152 encounters with consultation of an expert panel and an AIS resolution of every one minute.

Four encounter types were defined and each with its own sett of criteria. For crossing encounters where the give-way vessel are either passing ahead or behind the stand-on vessel DCPA and TCPA are used. For head-on and overtaking encounters the criteria used is violation of the 0.5 % percentile contour line of the estimated ship domain from AIS data for each respective encounter type (Iperen, 2015).

2.3.3 Vessel Conflict Ranking Operator

This method is developed and updated in (Zhang et al., 2015, 2016, 2017) and is a comprehensive method for ranking encounters with a ship domain base methodology. The VCRO score considered distance away from ship domain boundary, rate of change in distance, maneuverability in combination with relative bearings and size of the ships.

2.3.4 Analyzing concurrent trajectories

Goerlandt et al. (2012) made an near collision detection algorithm based on the Fujii ship domain see section2.4.1 and used definition (a) as the implementation of domain violation, where the largest ship is assigned as own ship and smaller ship as target ship, see section 2.4. The average transmission rate is every 5 min for the data used in (Goerlandt et al., 2012) and they acknowledge that it is insufficient to evaluate actions made by navigators.

2.4 Ship Domain

Ship domain was introduced as a concept in 1971, and have been widely used for maritime traffic engineering ever since Wang et al. (2009). The ship domain is often defined as; *"the effective* *area around a ship which a navigator would like to keep free with respect to other ships and stationary obstacles"* Goodwin (1975). Ship domains can have many different shapes, among these are circular, elliptical and other complex shapes Baran et al. (2018); Wang et al. (2009).

In addition to different shapes and sizes, the implementation of ship domains varies between researchers. In a review of ship domains Szlapczynski and Szlapczynska (2017) provided a list of four practical definitions for combining ship domain with safety criteria that encompasses various definitions used by researchers:

a) own ship's (OS) domain should not be violated by a target ship (TS),

b) a target ship's (TS) domain should not be violated by the own ship (OS),

c) neither of the ship domains should be violated (a conjunction of the first two conditions),

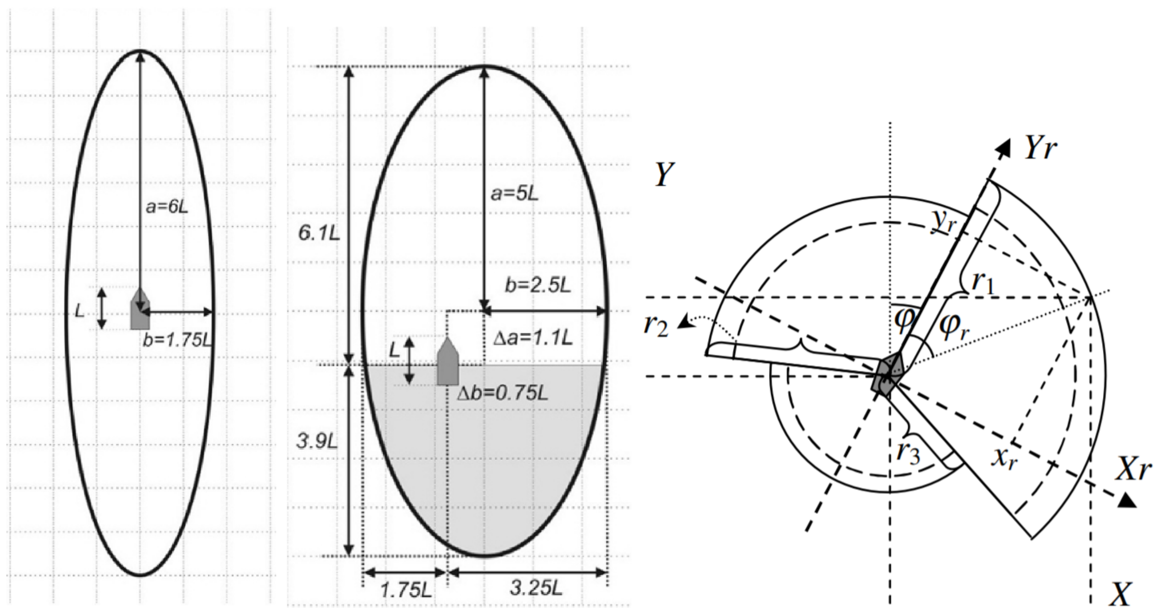
d) ship domains should not overlap - their areas should remain mutually exclusive (the effective spacing will be a sum of spacing resulting from each domain).

There are many different ship domains with various input variables and use cases. Szlapczynski and Szlapczynska (2017) have review the most used ship domain and found which ship and situational factors that may be used for describing ship domains:

- Ship length
- Own ship speed
- Own ship's manoeuvrability
- Target's length
- Target's speed
- Encounter type
- Weather conditions
- Traffic conditions
- COLREGS
- Human factors

2.4.1 Fujii

The Fujii ship domain is an empirical estimated domain based on ship movements in Japanese waters. It is elliptical in shape with semi major axis four times the length and semi minor axis 1.6 times the length (Wang et al., 2009). A limitation of empirically developed ship domains is that they are greatly dependent of the waterway geometry and traffic distribution(Szlapczynski and Szlapczynska, 2017).



(a) Coldwell - Overtaking (b) Coldwell - Crossing and head-on (c) Goodwin

Figure 2.1: Coldwell/Goodwin ship domain (Szlapczynski and Szlapczynska, 2017; Wang, 2010)

2.4.2 Coldwell

Similarly to the ship domain proposed by Fujii, the Coldwell ship domain is elliptical in shape and empirically based on rader data. Unlike Fujii which is static for all encounter types, Coldwell have two set of dimensions. It has one for overtaking encounters which is elliptical and centered around the vessel similar to Fujii, but with other lengths on the axis. For head-on and crossing situations it has a shifted centre(Szlapczynski and Szlapczynska, 2017), see figure 2.1 (c).

2.4.3 Goodwin

Another empirically based domain is the Goodwin circular, where as other empirically ship domains are typically based on its own length the circular domain have a set of different sizes. The different sectors seen in figure 2.1 (c) are normally taken with radii $r_1 = 0.85$ n.m. ,$r_2 = 0.7$ n.m.

and $r_3 = 0.45$ n.m. Wang et al. (2009).

2.4.4 Quaternion Ship Domain

The quaternion ship domain is made to solve a number of limitations with current ship domains identified by Wang (2010). The domain is based on the framework from (Wang et al., 2009) and shall be practical and feasible to use for both collision avoidance and as a risk assessment tool. The domain consists of four semi-axis indicating longitudinal and lateral radii, and the quaternion boundary can have two shapes; quadrangle or as combined ellipses see left and right domain respectively in figure 2.2 (Wang, 2010). The domain can be described on the form of:

$$f_k(x,y;Q) = \left(\frac{2x}{(1+sgnx)R_{fore} - (1-signx)R_{aft}}\right)^k + \left(\frac{2y}{(1+sgny)R_{starb} - (1-signy)R_{port}}\right)^k$$

Where the quadrangle corresponds to k=1 and and the shape of combined ellipses corresponds to k=2 or higher. The sett of radii factors in maneuverability from gains in advance and tactical diameter, speed and own vessel length. The sett of different radii will reasonably consider situations defined in COLREGS Wang (2010),see section 2.4 for detailed information of estimation of radii. There is a fuzzy version of this ship domain that can account for different levels of collision risk were $r \in (0,1)$ indicates risk by scaling the radii(Wang, 2010).

$$R_i(r) = \left(\frac{\ln\frac{1}{r}}{\ln\frac{1}{r_0}}\right) \quad R_i, i \in \{fore, aft, starb, port\}$$

$$f_k(x,y;Q(r)) = \left(\frac{2x}{(1+sgnx)R_{fore}(r) - (1-signx)R_{aft}(r)}\right)^k + \left(\frac{2y}{(1+sgny)R_{starb}(r) - (1-signy)R_{port}(r)}\right)^k$$

This domain is rather new and there are therefor limited research on the effect of this particular domain, but it has been applied successfully in the following work:(Liu et al., 2015; Qu et al., 2011; Chen et al., 2018; Zhou et al., 2018). On this basis the quaternion ship domain model is assumed to be accepted in the research community and robust enough for implementation in the proposed method.

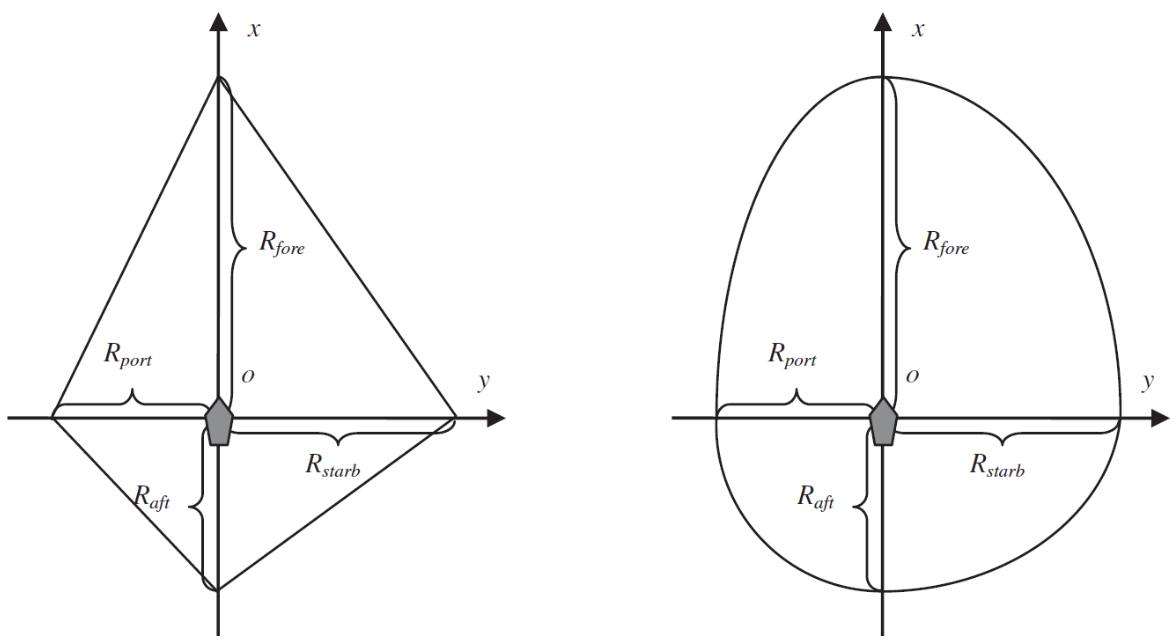


Figure 2.2: Quaternion ship domain Wang (2010)

3. Method

3.1 Model structure

The methods and algorithms used in the proposed model are based on the "Near collision detection algorithm" of (Goerlandt et al., 2012) and the critical maneuver identification proposed by (Mestl et al., 2016). The model consists of three main steps, first is to sort all AIS transmissions chronologically and conform them to a fixed interval ready for step two, see section 3.2 for further details.

Second step is to check all concurrent AIS messages hereinafter referred to as entries, for any other entry intruding into its own ship domain. If one or both ships in an encounter are intruding into the other vessel's domain all information for both vessels at that time instance will be saved into a single entry used in step 3. This is similar to the "Near collision detection algorithm" described in (Goerlandt et al., 2012). The main difference in the proposed model is that all time steps i.e. entries are treated independently while the "Near collision detection algorithm" look at concurrent trajectories at 5 minute intervals and interpolating values in between. The data used in this thesis is processed raw data which means it has a transmission rate from every 2 second to every 3 minute (IMO, 2002a). From table 3.1 it is clear that all ships en route should have a transmission rate of at least one every 12 sec. Longer periods are observed, and are caused by messages being "lost" or other technical issues.

The third step is a sorting algorithm based on (Mestl et al., 2016), in this step a threshold for minimum rate of turn is set. In a critical encounter it is to be expected that at least one of the navigators make evasive maneuvers. On the contrary if no significant maneuvers are initiated then the encounter is deemed safe by both navigators. For the present model it is assumed that the officer on watch for both vessels are alert and ready to take action if necessary, this assumption circumvent the challenges described by (Mestl et al., 2016). With this assumption there has not been implemented fail-safes for potential situation where the OOW on both vessels fails to recognize a critical situation.

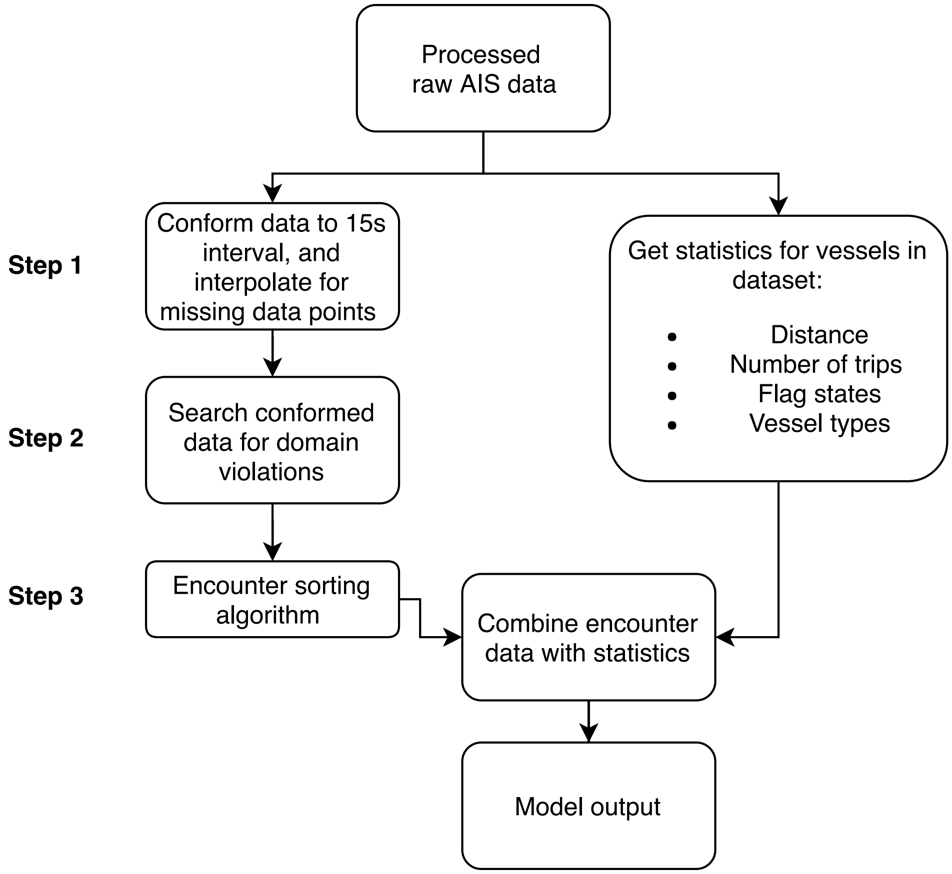


Figure 3.1: Model structure

3.2 Data handling

The approach described in this chapter is made to work with the data format provided by Safetec Nordic, but it will need minimal modifications to work with other sources. The processed raw AIS data is delivered in comma separated data files. For performance and compatibility SQLite is chosen as the data management solution. The two main advantages for choosing the SQLite database engine are the *Zero-Configuration*[1] aspect of it and that it is simple to use *SQLite3* module for Python[2].

Importing of csv files are performed by an open source tool called *DB browser for SQLite*[3] that have a simple visual user interface for interacting with SQLite databases.

The program retrieves data directly from the SQLite databases through queries which are

[1] https://www.sqlite.org/zeroconf.html
[2] https://www.pythoncentral.io/introduction-to-sqlite-in-python/
[3] http://sqlitebrowser.org/

Table 3.1: Reporting rate for dynamic messages IMO (2002a)

Reporting intervals for Class A	
Vessel status	General reporting interval
Ship at anchor	3 min
Ship 0-14 knots	12sec
Ship 0-14knots and changing course	4 sec
Ship 14-23 knots	6 sec
Ship 14-23 knots and changing course	2 sec
Ship > 23 knots	3 sec
Ship > 23 knots and changing course	2 sec

really fast operations, but model output is written to csv files that in turn are imported to the database. The reason for writing to csv files instead of directly to the database are errors when several processes are trying to write to a single database. Databases and other file types including csv files are not made to be accessible or writable simultaneously by two or more processes. To overcome this problem, access to a particular file is locked[4] while being read or written to by a process. This solution should work for both reading and writing to databases, unfortunately it does not work reliably for writing data. Since writing to csv files seemingly works flawlessly no further time is used for resolving this issue. The extra time it takes to import the the files are negligible compared to the processing time.

The regularity of AIS messages is dependent on many factors, see section 2.1. For the purposes of this thesis a frequency of 15 seconds is considered sufficient. For

The Pandas module is a popular choice for large dataset manipulation in Python and is chosen for conforming the data. For each unique MMSI number, the corresponding raw AIS data is read from the SQLite table to a dataframe[5]. A fixed 15 sec frequency DatetimeIndex is created for the year, in which all entries are conformed to. "Nearest" is the method selected for re-indexing i.e conforming, with a tolerance of 4 seconds.

For all rows not filled by nearest, the program will forward fill last known value for static data and course over ground(COG) for 4 time steps i.e. one minute. Latitude, longitude and speed over ground(SOG) are linearly interpolated from last known row towards next row for the same 4 time steps. Although COG is dynamic data, the linear interpolation will often produce erroneous values when the general heading are 0° north. COG will therefore be more accurate when filling empty rows with last known value. After filling up to 4 rows from known values, the re-

[4]The multiprocessing module in python have a specific function for locking access to files temporarily.

[5]Dataframes are similar to SQL tables, in which they have labeled columns with defined data types. `https://pandas.pydata.org/pandas-docs/stable/dsintro.html`

maining empty rows are discarded. Conforming the data reduces the number of rows by 38-52 percent see table 4.1, but it will also insert data where AIS messages are lost.
For more details see the provided code in appendix A.1

3.3 Computing

The processing intensive tasks in the proposed method takes an impractically long time running on a single core which is the normal behavior for most of python. Python has an global interpreter lock(GIL) which basically locks python code to only run on one thread. To work around this, the joblib package is used which in turn utilize the multiprocessing module which is a part of the standard python library.

The simplest way of creating multicore programs are to create workloads without dependencies. These types of workloads are often referred to as "Embarrassingly parallel". Joblib makes this is a fairly simple task by configuring the number of concurrently running jobs i.e. number of cores, giving it a function and a list of arguments. The Performance is close to linear with the number of cores which means it is easy to scale up.

It is highly recommended to use an operating system based on Linux, the current implementation of multiprocessing will not work on windows based systems[6]. The program will only run on python version 3.6 or later, this is due to the formatting of SQLite queries. The following packages not included in the standard python library will have to be installed:

- Pandas version 0.22.0
- Geographiclib 1.49
- Joblib 0.11
- Dateutil 2.7.2

[6]Joblib is by default using "fork" as start method of processes in the multiprocessing module of python which is not available on windows. https://docs.python.org/3.6/library/multiprocessing.html

3.4 Ship Domain

From section 2.4 it is clear that there are many ship domain models to choose from, a crisp quaternion ship domain is selected (Wang, 2010). The quaternion ship domain has a couple of features making it a good choice. It is dynamic in terms of speed, which is reasonable in determining severity in an encounter. Encounter situations defined in COLREGS are also taken in consideration by the asymmetric shape defined by the four radii. The ship domain considers maneuverability see eq. 3.7, but as Szlapczynski and Szlapczynska (2017) points out the effect has not been sufficiently documented.

Geographiclib for python is used to solve geodesic problems i.e. distances and angles on an ellipsoid. This package is compatible with the WGS84 coordinate reference system, which is used for Global Positioning System(GPS)GIS Geography (2018). An advantage of geographiclib is that angles are continuous below and above base 360 for heading. This leads to simpler code, one example of this is that negative 10 degrees nets the same result as 350 degrees.

True position of ship center is found by solving a direct geodesic problem see equation 3.1. Input variables are coordinates, clockwise angle *azi1_True* from 0° North and distance *s12* in meters. The distance *s12* is found from equations (3.4 - 3.6), where d_{bow} denotes distance to bow and so on from location of AIS transmitter to vessel perpendiculars. Angle β corresponds to clockwise angle to ship center standing on the position of AIS transmitter looking straight ahead.

$$True\_center = Direct.WGS84.Direct(Latitude, Longitude, azi1\_True, s12) \tag{3.1}$$

$$azi1\_True = COG + \beta \tag{3.2}$$

$$\beta = \begin{cases} 0 & for \quad d_{bow} > d_{aft} \wedge d_{starb} = d_{port} \\ 180 & for \quad d_{bow} < d_{aft} \wedge d_{starb} = d_{port} \\ 90 & for \quad d_{bow} = d_{aft} \wedge d_{starb} > d_{port} \\ -90 & for \quad d_{bow} = d_{aft} \wedge d_{starb} > d_{port} \\ \arctan\frac{dist\_y}{dist\_x} & for \quad d_{bow} > d_{aft} \wedge d_{starb} > d_{port} \\ 360 - \arctan\frac{dist\_y}{dist\_x} & for \quad d_{bow} > d_{aft} \wedge d_{starb} < d_{port} \\ 180 + \arctan\frac{dist\_y}{dist\_x} & for \quad d_{bow} < d_{aft} \wedge d_{starb} < d_{port} \\ 180 + \arctan\frac{dist\_y}{dist\_x} & for \quad d_{bow} < d_{aft} \wedge d_{starb} > d_{port} \end{cases} \tag{3.3}$$

$$s12 = \sqrt{dist\_x^2 + dist\_y^2} \tag{3.4}$$

$$dist\_x = \begin{cases} (d_{bow} + d_{aft})/2 - d_{aft} & for \quad d_{bow} > d_{aft} \\ (d_{bow} + d_{aft})/2 - d_{bow} & for \quad d_{bow} < d_{aft} \end{cases} \tag{3.5}$$

$$dist\_y = \begin{cases} (d_{starb} + d_{port})/2 - d_{port} & for \quad d_{starb} > d_{port} \\ (d_{starb} + d_{port})/2 - d_{starb} & for \quad d_{starb} < d_{port} \end{cases} \tag{3.6}$$

For a vessel position given in table 3.2 the location of AIS transmitter is at F, and ship center at H for an imagined rectangle defined by length and width, see figure 3.2.

$$dist\_x_1 = \big(|FN| + |FM|\big)/2 - |FN|$$

$$dist\_y_1 = \big(|FO| + |FP|\big)/2 - |FO|$$

$$s12 = g = \sqrt{dist\_x_1^2 + dist\_y_1^2}$$

$$azi1\_True = COG + \arctan\left(\frac{dist\_y_1}{dist\_x_1}\right)$$

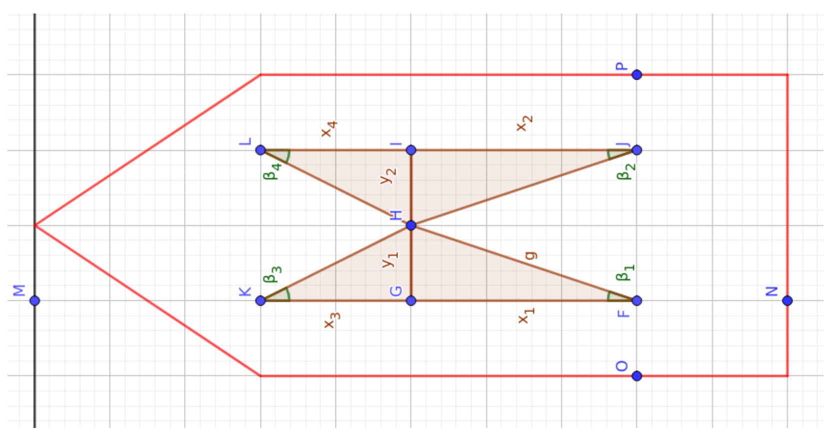


Figure 3.2: Definitions of geometries used to find true ship center.

From true ship center the distance to boundary of own ship domain in direction of target ship center is calculated from equation 3.11 similar to (Zhang et al., 2016), but where the minor and major axis is switch due to different definition of α. Angle α is defined as the angle from own ship heading(COG) measured clockwise to target ship center see equation 3.10. The geographiclib package is used to find absolute clockwise angle *azi1* from own ships center(subscript 1) to target ship center(subscript 2) from 0° North, except for when target ship is west of own ship. For cases where target ship is west of own ship the angle is negative and calculated anticlockwise. For continuous values of *azi1* clockwise from 0° North, it is defined as in equation 3.9.

True_Center_Coordinates	
Latitude	67.5° N
Longitude	11.5° E
Dimension to bow	160 m
Dimension to aft	40 m
Dimension to port	20 m
Dimension to starboard	60 m
COG	45°
True center Latitude	67.500 253 599 614 91°N
True center Longitude	11.501 324 111 061 274° E

Table 3.2: Example of finding true center ship center coordinates from AIS data

$$\begin{cases} k_{AD} = 10^{(0.3591*log_{10}(v_{own})+0.0952)} \\ k_{DT} = 10^{(0.5441*log_{10}(v_{own})-0.0795)} \end{cases} \tag{3.7}$$

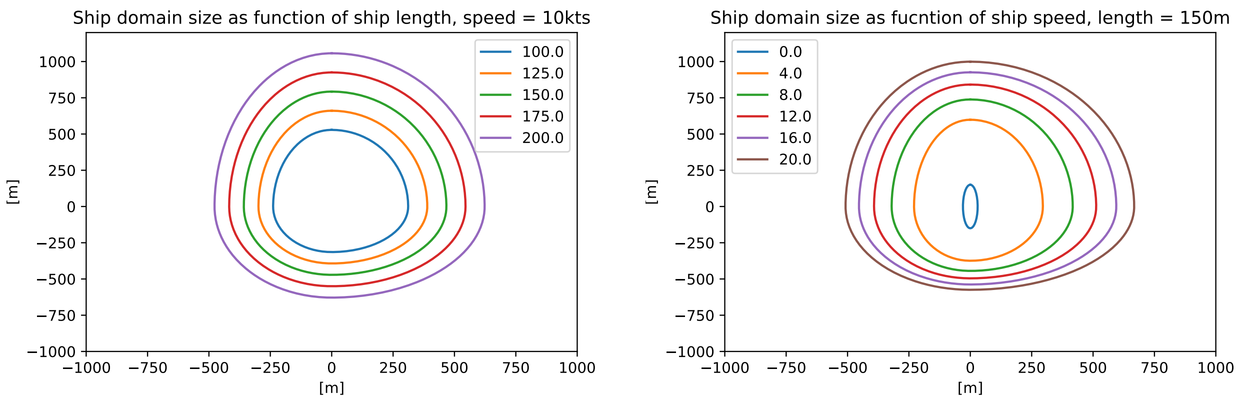
$$\begin{cases} R_{fore} = \left(1 + 1.34 * \sqrt{(k_{AD}^2 + (k_{DT}/2)^2} \right) * length \\ R_{aft} = \left(1 + 0.67 * \sqrt{k_{AD}^2 + (k_{DT}/2)^2} \right) * length \\ R_{starb} = (0.2 + k_{DT}) * length \\ R_{port} = (0.2 + 0.75 * k_{DT}) * length \end{cases} \tag{3.8}$$

$$\begin{cases} azi1 = Geodesic.WGS84.Inverse\left(lat_1, lon_1, lat_2, lon_2,\right) & if \quad lon_2 \geq lon_1 \\ azi1 = 360 + Geodesic.WGS84.Inverse\left(lat_1, lon_1, lat_2, lon_2,\right) & if \quad lon_2 < lon_1 \end{cases} \tag{3.9}$$

$$\begin{cases} \alpha = 360 - (COG - azi1) & if azi1 \leq COG \\ \alpha = azi1 - COG & if azi1 > COG \end{cases} \tag{3.10}$$

$$
\begin{cases}
l_\alpha = \left(\dfrac{1+tan^2\alpha}{\frac{1}{R_{fore}{}^2}+\frac{tan^2\alpha}{R^2_{starb}}} \right)^{1/2} & \text{if} \quad \alpha \leq \frac{\pi}{2} \\
l_\alpha = \left(\dfrac{1+tan^2\alpha}{\frac{1}{R^2_{aft}}+\frac{tan^2\alpha}{R^2_{starb}}} \right)^{1/2} & \text{if} \quad \frac{\pi}{2} < \alpha \leq \pi \\
l_\alpha = \left(\dfrac{1+tan^2\alpha}{\frac{1}{R^2_{aft}}+\frac{tan^2\alpha}{R^2_{port}}} \right)^{1/2} & \text{if} \quad \pi < \alpha \leq \frac{3}{2}\pi \\
l_\alpha = \left(\dfrac{1+tan^2\alpha}{\frac{1}{R^2_{fore}}+\frac{tan^2\alpha}{R^2_{port}}} \right)^{1/2} & \text{if} \quad \frac{3}{2}\pi < \alpha
\end{cases}
\tag{3.11}
$$

The ship domain is defined by longitudinal and lateral radii given in equation 3.7 and equation 3.8. Coefficients in equation 3.7 k_{AD} and k_{DT} represents gains from advance[7] and tactical diameter[8] respectively Wang (2010). The ship domain forms a quaternion of combined ellipses see figure 3.3.



(a) Length (b) Speed

Figure 3.3: Figure (a) is showing the size of ship domain as a function of length, where speed is constant at 10kts. Figure (b)is showing the size of ship domain as a function of speed, where length is constant at 150m.

[7] Advance is the distance travelled in the direction of the original course by the midship point of a ship from the position at which the rudder order is given to the position at which the heading has changed 90° from the original course IMO (2002b)

[8] Tactical diameter is the distance travelled by the midship point of a ship from the position at which the rudder order is given to the position at which the heading has changed 180° from the original course. It is measured in a direction perpendicular to the original heading of the ship IMO (2002b)

3.5 Detecting violations

A violation is detected when the distance between two ship centers is less than the ship domain distance of one or both vessels. This definition corresponds to the third practical definition ship domain and safety criteria from Szlapczynski and Szlapczynska (2017), see section 2.4.

For a violation to be classified as exceptional a timeseries of raw data starting 3 minutes before CPA until 2 minutes is analyzed. An algorithm based on the concept of identifying critical maneuvers from ROT (Mestl et al., 2016) is used to sort out exceptional encounters. Rate of turn is estimated from equation 3.12 for both vessels and if the vessel whose domain is violated have higher maximum ROT than a specific threshold it will be sorted as an exceptional encounter. Mestl et al. (2016) used a threshold of 150 deg/min accounting for the top 0.001% of registered ROT values for a circa 100 meter ropax vessel. Since the current model does not include ROT profiling for different vessel types and lengths, a conservative threshold of 70 deg/min is chosen. This threshold of a little under half is assumed low enough that evasive maneuvers initiated by larger vessels would exceed this limit.

$$\widehat{ROT}_t = \frac{1}{2}\sum_{i=t}^{t+1}\frac{COG_i - COG_{i-1}}{time_i - time_{i-1}} \tag{3.12}$$

In addition to detecting domain violations, all of them are classified into four groups see table 3.3. The criteria for determining encounter type are based on a simplified version of those used in (Iperen, 2015). The angle φ is the relative heading between the vessel pair at CPA, if the criteria for *Overtaking* or *Head-on* is met no further calculations are made. For crossing encounters one of the vessels is determined to have right of way while the other shall keep out of the way in accordance with rule 15 in COLREG - rules for Preventing Collisions at Sea (IMO, 1972). In the simplistic definition of stand-on and give-way vessels, rule 18 dictating responsibility between vessels are not considered. Rule 15 dictates that; "...the vessel which has the other on her own starboard side shall keep out of the way..." At $time_0$ 3 minutes before CPA the ship whom has the other vessel on their own starboard side is determined to be give-way vessel. Further at CPA, α is the clockwise angle of stand-on vessel's heading to center of give-way vessel centre.

3.6 Traffic Statistics

In addition to finding the total amount of domain violation and exceptional encounters it is necessary to normalize the results. The normalized result will show how many domain violations and exceptional encounters that occurs per distance unit or trip for individual groups. This is

Table 3.3: Ship encounter categories

Ship Domain Violations	
Encounter type	Conditions
Overtaking	$\varphi \leq 25$
Head-on	$165 \leq \varphi \leq 195$
Crossing, give way ship passing at bow	$25 < \varphi < 165$ or $195 < \varphi < 335$ $\alpha \leq 90$ or $\alpha \geq 270$
Crossing, give way ship passing at stern	$25 < \varphi < 165$ or $195 < \varphi < 335$ $90 < \alpha < 270$

useful for differentiating between higher risk from traffic volume or increased individual risk.

Flag state is simple to determine[9] from MMSI numbers, the flag for each vessel is found by comparing the three first digits to a list of all flag states[10].

Determining vessel type is more difficult, in the case study three different sources is used to categorize the vessel types. The most reliable is a comprehensive list of ships with statcode5 connected by IMO and MMSI. Most of the larger vessels are covered by this list. The second source is ship type from static AIS information which is transmitted as a two digit code[11], due to some complications this data is only available for the 2013 dataset. This data can be unreliable and is not easily corrected by the seamen operating the vessel (IALA 2016), see section 4.1.1 for further details on this. The third source is marinetraffic[12], a web scraping script is made to match IMO numbers from the raw AIS messages to AIS vessel type on the website. This approach is used for vessels not covered by the two previous sources. Where IMO number is either not found or missing, the script will then try to match MMSI number.

[9] https://help.marinetraffic.com/hc/en-us/articles/205220087-Which-way-is-information-on-a-vessel-s-flag-found-

[10] https://help.marinetraffic.com/hc/en-us/articles/215699608-MIDs-Countries-and-Flags-full-table-

[11] https://help.marinetraffic.com/hc/en-us/articles/205579997-What-is-the-significance-of-the-AIS-Shiptype-number-

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

The ship type data is found to be erroneous, especially for fishing vessels. Manual cleaning is therefore required. All common errors(see section 2.1 and vessels involved in domain violations is inspected manually, the resulting accuracy of ship type is presumed to be acceptable.

3.6.1 Distance and number of trips

Distance is calculated for each unique MMSI number by accumulating the geodesic distance between the coordinates for all data points. From section 2.1 it is clear that AIS is not flawless, a condition is set to avoid calculating distance when stationary. Some variation in the coordinates will be present while being anchored or otherwise moored, and these variations should not be included towards the traveled distance. This would artificially increase the distance for vessels being anchored or moored inside the the case study area. This is solved by using a centered three point moving average for SOG. A threshold for average speed of 0.2 kts is set, below this threshold and the distance between coordinates will not we accumulated towards the total distance. The algorithm calculating distance will remember when it last added a distance measurement, and if it is more than 9 minutes ago the algorithm will add another trip to the total. The time limit of 9 minutes are set to enable the algorithm to catch short turn around for smaller passenger vessels.

4. Case study: Vestfjorden

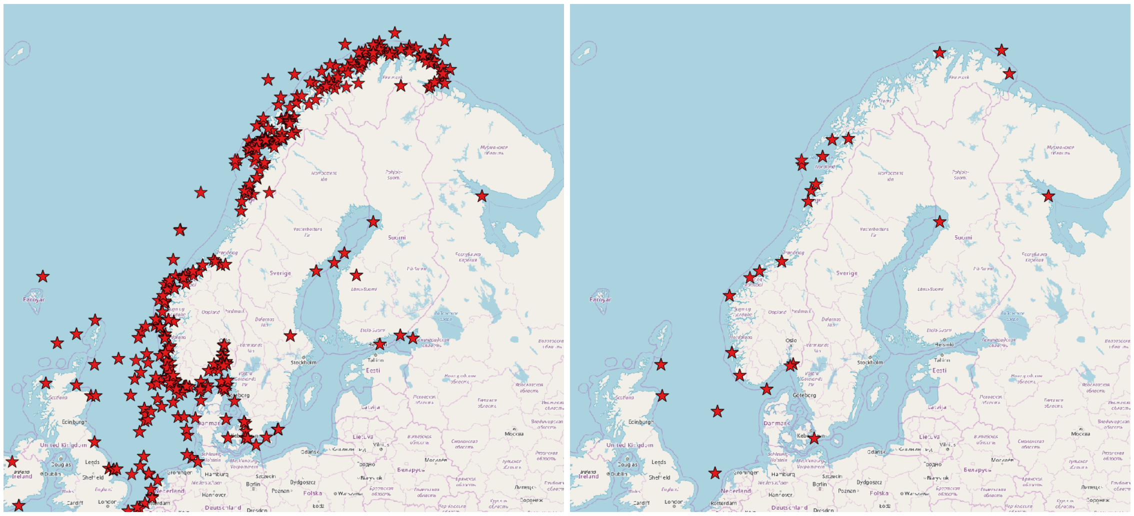
For the purpose of this thesis, processed raw AIS data were generously provided by Safetec Nordic.

For validation of suggested method, an area with known accidents are necessary. The best case scenario is an area with accidents over several years. Figure 4.1(a) shows all registered collisions at Norwegian Maritime Authority (NMA)[1] from 1987 to 2017 with filters "Ytre kystefarvann" or "Åpent havområde"(Outer coastal water or open waters) that are applicable for the model. The available AIS data starts in 2010 which limits the number of reported collisions considerably, see figure (b). Further limiting possible areas are the reach of AIS transmissions which is up to 40 nautical miles for base stations (NASA, 2015). Satellite-AIS would give much better coverage, but it is not continuous.

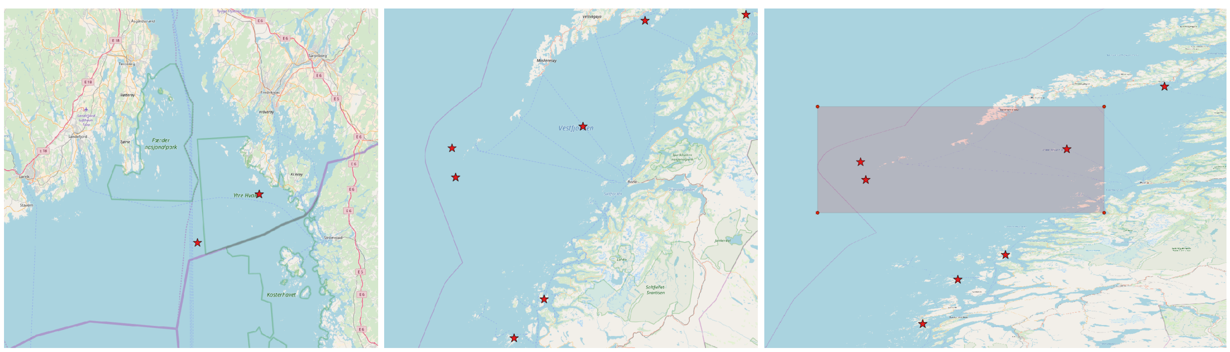
With these conditions and a visual inspection of figure 4.1(b) two possible areas are considered, see figures (c) and (d). In Vestfjorden there are three collisions that are considered to be applicable. The two westernmost collisions happened in 2010, where one of them involved 2 fishing vessels and the other involved one fishing vessel with the other unknown. The accident in the middle of Vestfjorden happened in 2013 between a ropax ferry and a fishing vessel. The collision outside Oslofjorden happend in 2010 between a cargo vessel and an unknown vessels. For the purpose of validating the model Vestfjorden is selected on the basis of having more collisions.

An area limited by coordinates SW(67',11.3'), NE(68',14') in the time span of 2010 to 2016 is selected for the case study, see figure 4.1(e). Before analyzing the seven year period a control is made to check if the vessels involved exists in the dataset. The two collisions in 2010 are unfortunately between vessels without AIS transmitters or class B in 2010. It is therefore chosen to analyze the period from 2013 to 2016 to reduce the amount of data and computational time.

[1] https://www.sdir.no/sjofart/ulykker-og-sikkerhet/ulykkesstatistikk/



(a) All collisions (b) All collisions as of 2010



(c) Oslofjorden (d) Vestfjorden (e) Marks area selected for case study

Figure 4.1: The figures shows registered collisions from 1987 to 2017 at NMA filtered by "Ytre kystefarvann" or "Åpent havområde"(Outer coastal water or open waters).

4.1 Data

The available data used in the case study is processed raw AIS class A. As mentioned in section 3.2 the data is delivered in a comma separated files and they take up 37 GB of storage. Class A signifies that it is only vessels with mandatory AIS equipment that are included in the dataset. The four years combined have 432.8 million messages, 2855 unique MMSI numbers, 66 different flag states and a combined traveled distance for all vessels of 2.85 million nautical miles.

The machine used for processing has an 8 core, 16 thread AMD 3.2 Ghz CPU. It is running Ubuntu 18.04 and have 64GB of memory available. During processing the highest memory utilization is around 1GB per active process. Conducting analysis on larger areas would significantly increase the memory requirements per process without

Table 4.1: Information of data used in analysis of Vestfjorden 2013-2016

Conforming data				
Year	2013	2014	2015	2016
Time interval [seconds]	15	15	15	15
Unique MMSI numbers	1508	1382	1584	1680
Raw data rows [millions]	78.6	101.6	109.9	142.7
Conformed data rows [millions]	47.1	49.4	68.5	88.4
Percent change in number of rows[%]	40	52.4	37.7	38.1
Time in minutes	16.4	15.3	17.9	18.6

4.1.1 AIS validity

As described earlier in section 2.1 the quality of AIS data are not perfect. Some of the observed errors are:

- SOG exceeding 100 knots which are obliviously false
- Missing values for SOG or COG
- Two different vessels have the same MMSI
- One Norwegian passenger vessel have AIS transmissions every 1 minute regardless of speed during the 4 year period.
- Erratic data see scenario 4.3.5
- Missing or false data in static information
  - Common error is dimension_to_bow being equal to vessel length and imo_number equal to 0
  - Dimension to ship perpendiculars can changing over time, or having two sets of dimensions(not related to different vessel with same MMSI)
  - Type of vessel being set wrong, many fishing vessels(type 30), are observed to be empty, 0, type 20 or type 90.
  - Null or empty fields
  - High speed craft and passenger
  - RUS fishing vessel with dimension 185m x 51m actual 51m x 10 m

4.1.2 Handling erroneous AIS data

The top part of table 4.2 is showing the amount of AIS messages where the recorded speed is unreasonably high. Number of ships is the number of unique MMSI's which is close to the number of unique vessels. The middle part shows the number AIS messages with missing dynamic information of SOG and COG.The bottom displays the number of AIS messages where static data of ship dimension are missing.

Speed over ground

Missing values of SOG in the dataset affect both statistics and the detection model. For the statistics of vessel movements missing values has limited effect. The algorithm calculating distance traveled for each vessel discards distances where the centered three point moving average of SOG is less than 0.2 kts. The average function used will omit NaN-values, but if all three values are missing the distance between messages will be accumulated. Considering the largest share of missing values in 2016 is 1.68×10^{-3} the small increase in total distance is negligible.

SOG is used to determine size of ship domain and missing values are thus more critical than for distance calculation. The solution is a simple yet robust, missing values of SOG is replaced with 5.001. As figure 3.3(b) shows, speed has a diminishing effect on domain size. The speed chosen is large enough to exceed the range of rapid increase of domain size while not being excessive. A better approach could have been to make a speed profiles for different vessel types, but it is not determined to be a rational use of time when the share of messages is low. By using three decimals it simple to identify violations where artificial values are inserted for missing values.

A curiosity observed in all 4 datasets are that the maximum registered speed over ground is 102.2 kts, this may be a systemic error in the AIS system. Smestad (2015) whom analyzed a different sett of AIS data had similar maximum speed of 102.3 knts, reinforcing that this may be a systemic error. In addition there are many ships with far too high SOG. An arbitrary limit of 26 kts is chosen for maximum speed, if an AIS messages has SOG above 26kts it will be treated like it has a SOG of 10 kts.

Course over ground

Course over ground is used to find distance from own ship to domain boundary in direction of target ship and to categorize the encounter. For missing COG the model will fall back on a circular domain with radii equal to R_{port}. In the event of missing COG for either vessel at closest point of approach, it will be categorized as unknown.

Ship dimensions

Dimensions to ship perpendiculars are used for determining true center and the combined length of dimension to bow and aft is used to determine ship domain size. In the event of missing either dimension to port or starboard, only true centre calculation will be affected. The true centre will then just be adjusted in the longitudinal direction, and if dimension to bow or aft is missing true center will not be calculated.

More important is missing data for dimension to bow or aft will result to a standard length of 40.001 meters. The length and speed of the vessel are the two factors determining ship domain size. Unlike SOG, length does not diminish as it increase and there will thus be more uncertainty related to this solution.

Ghost ships

Unique MMSI numbers with less than 3 AIS messages is an increasing trend. These unique MMSI numbers are marked with error in their ship type and disregarded in all statistics.

Table 4.2: An overview of the amount of erroneous data impacting the model.

AIS data errors				
Year	2013	2014	2015	2016
AIS messages with 102.2 kts Number of ship	1091 5	72 4	1206 6	361 4
AIS messages over 26kts Number of ships	52.9×10^4 46	49.7×10^4 44	42.3×10^4 44	104.5×10^4 57
AIS messages without SOG Number of ships	1.1×10^4 110	0.6×10^4 87	2.5×10^4 108	24.1×10^4 116
AIS messages without COG Number of ships	122.1×10^4 187	859.2×10^4 167	1401.1×10^4 221	2094.2×10^4 231
AIS messages without d_bow or d_stern Number of ships	701.9×10^4 107	800.5×10^4 112	896.2×10^4 151	1696.7×10^4 224
AIS messages without d_port or d_starboard Number of ships	724.6×10^4 115	820.7×10^4 115	974.4×10^4 165	1689.9×10^4 225
Number of unique MMSI numbers with less than 3 AIS messages	2	6	34	113

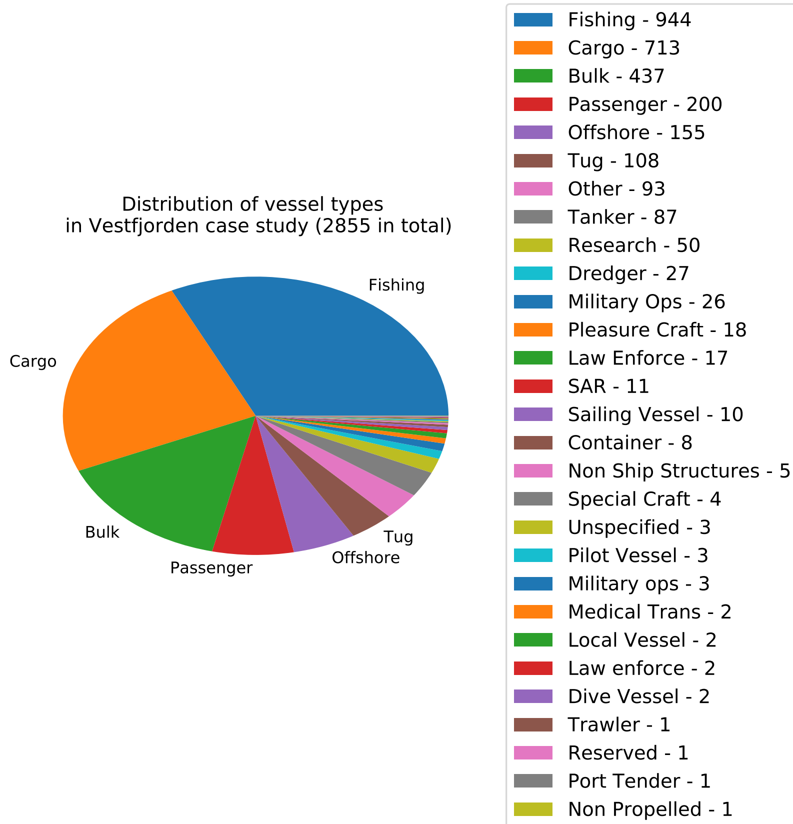


Figure 4.2: Distribution of vessel types

4.2 Statistics

From table 4.4(a) it is clear that three types of vessels make up most of total distance which is fishing, passenger and freight[2]. Considering figure 4.2 showing the amount of passenger vessels it is clear that passenger vessels on average travel much further than other vessel types. This is excepted as there are a number passenger routes with several departures each day.

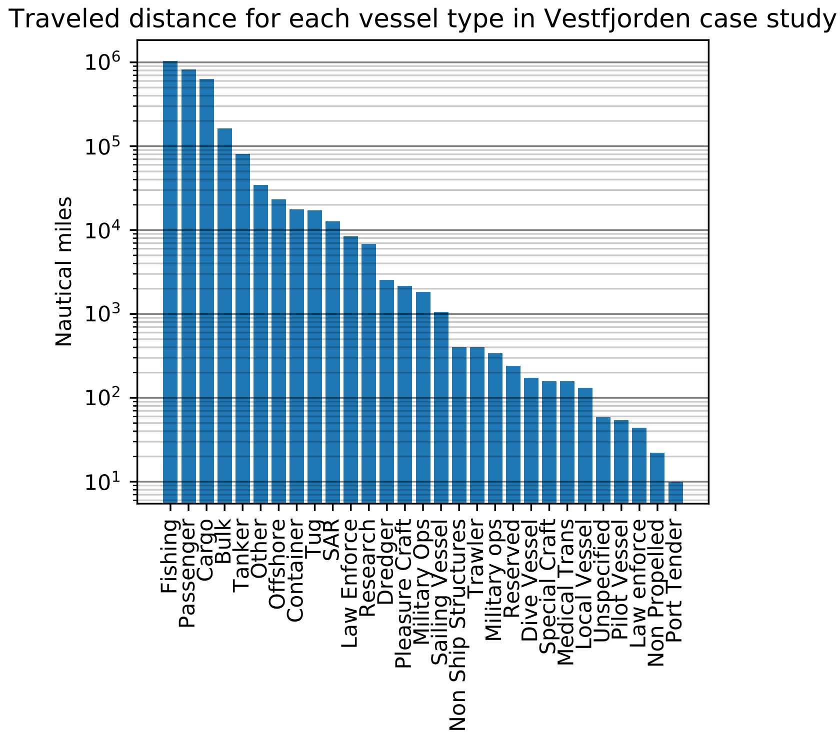
Table 4.1 shows that the data collection increases year by year, this is assumed to be an increase in coverage. The increase is not assumed to be correlated with total distance traveled by all vessels. This assumption is strengthened by statistics from the "Havbase" project from the Norwegaian Costal Administration. Table 4.3 shows all crossings of passing lines "Vestfjorden" and "Fergetrafikk Bodø - Lofoten" (see fig 4.3), there is no indicates of an increase in the traffic volume.

[2]Includes: Cargo,Bulk and Tanker

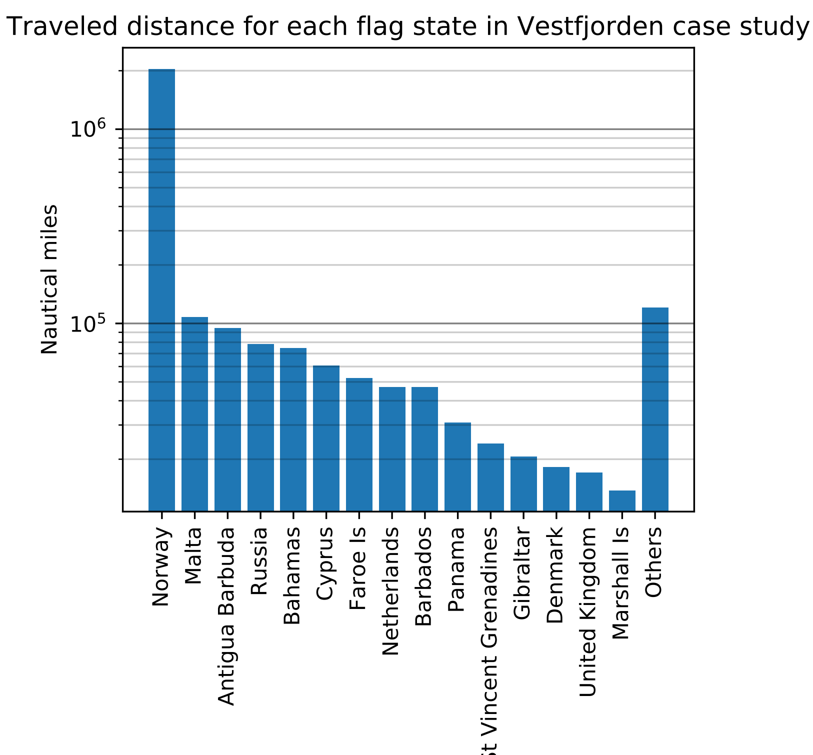
Figure 4.3: Shows passing lines Vestfjorden and "Fergetrafikk Bodø-Lofoten" from havbase.no

Table 4.3: Registered number of crossings over passing lines in both directions from havbase.no

Passing traffic from Havbase				
Year	2013	2014	2015	2016
Vestfjorden	10748	10901	10712	10579
Fergetrafikk Bodø - Lofoten	5075	4519	4325	4604



(a) Travel distance per vessel type



(b) Travel distance per flag state

Figure 4.4: Bar charts showing distance traveled for each vessel type and flag state, note the logarithmic scale

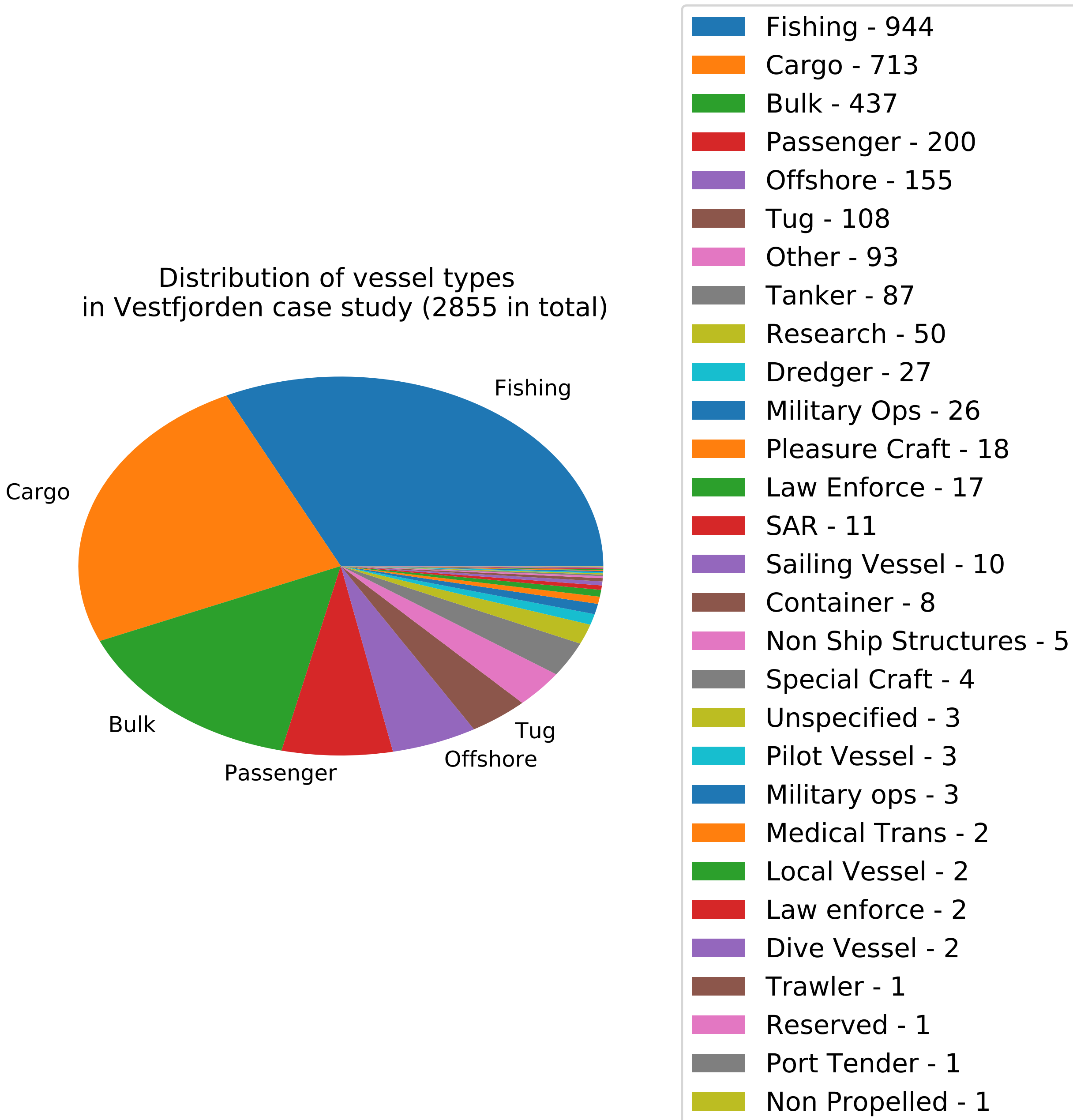


Figure 4.5: Distribution of vessel types

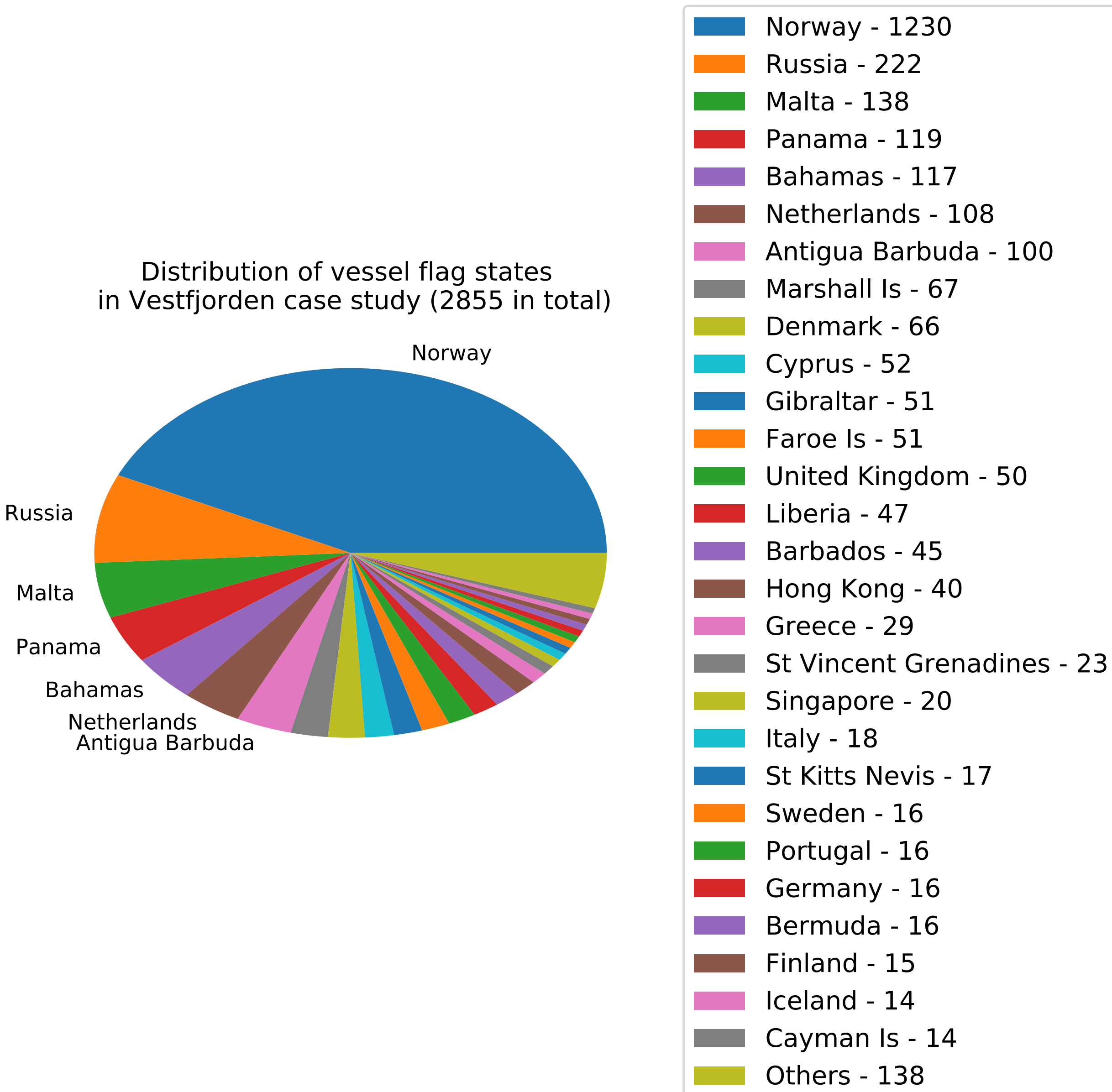


Figure 4.6: Distribution of flag states

4.3 Examples of detected encounters

The time stamp shown on some of the images in this section are not local time. An unknown error leads time stamps of the AIS data to be equal to UTC -2 instead of the standard UTC.

4.3.1 Collision between ferry and fishing vessel

Among the 133 encounters that passed the sorting algorithm for rate of turn with a threshold of 70 deg/min in 2013 is a crossing encounter that lead to a collision. This is the collision registered at NMA and proves that the proposed model has potential as a risk assessment tool.

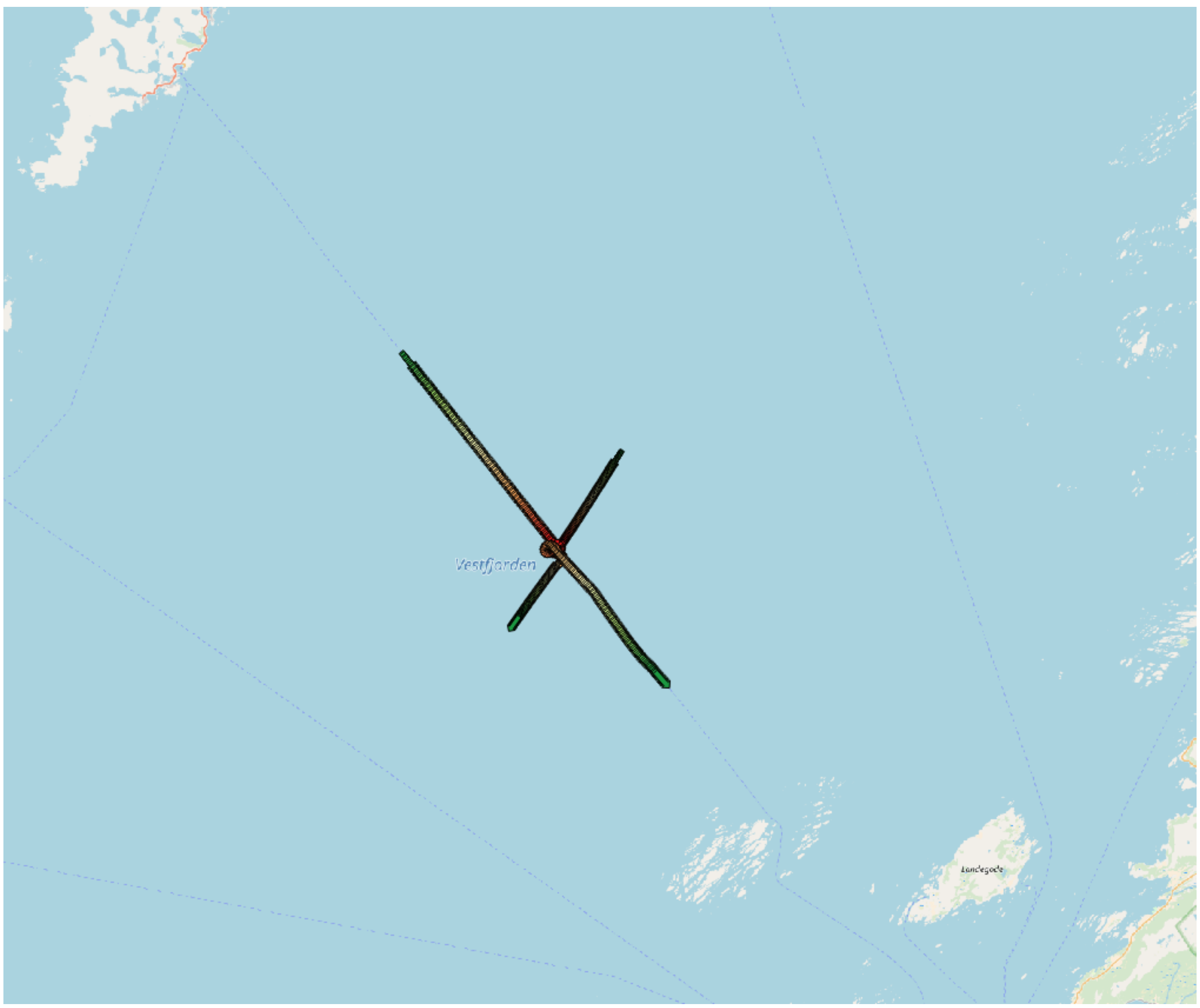


Figure 4.7: AIS tracks of collision between a ropax ferry and a smaller fishing vessel.

The collision happened in the middle of Vestfjorden, see figure 4.7 between a ropax ferry and a fishing both under Norwegian flag. The ropax vessel was moving in a southeasterly direction, and it has a speed over ground of around 18kts. The fishing vessel was moving in a southwesterly directing with speed over ground of around 8kts.

By coincidence the encounter was the subject of Mestl et al. (2016), from which the sorting algorithm in step 3 are derived. By comparison the maximum estimated ROT for the ferry is 145.5 deg/min see figure 4.8, while the maximum measured ROT is 194.5deg/min Mestl et al. (2016). The lower maximum value can be attributed by using a moving average and estimating it from COG and time delta between messages. The fishing vessel had a maximum ROT of 252.6 deg/min which probably occurred during the impact.

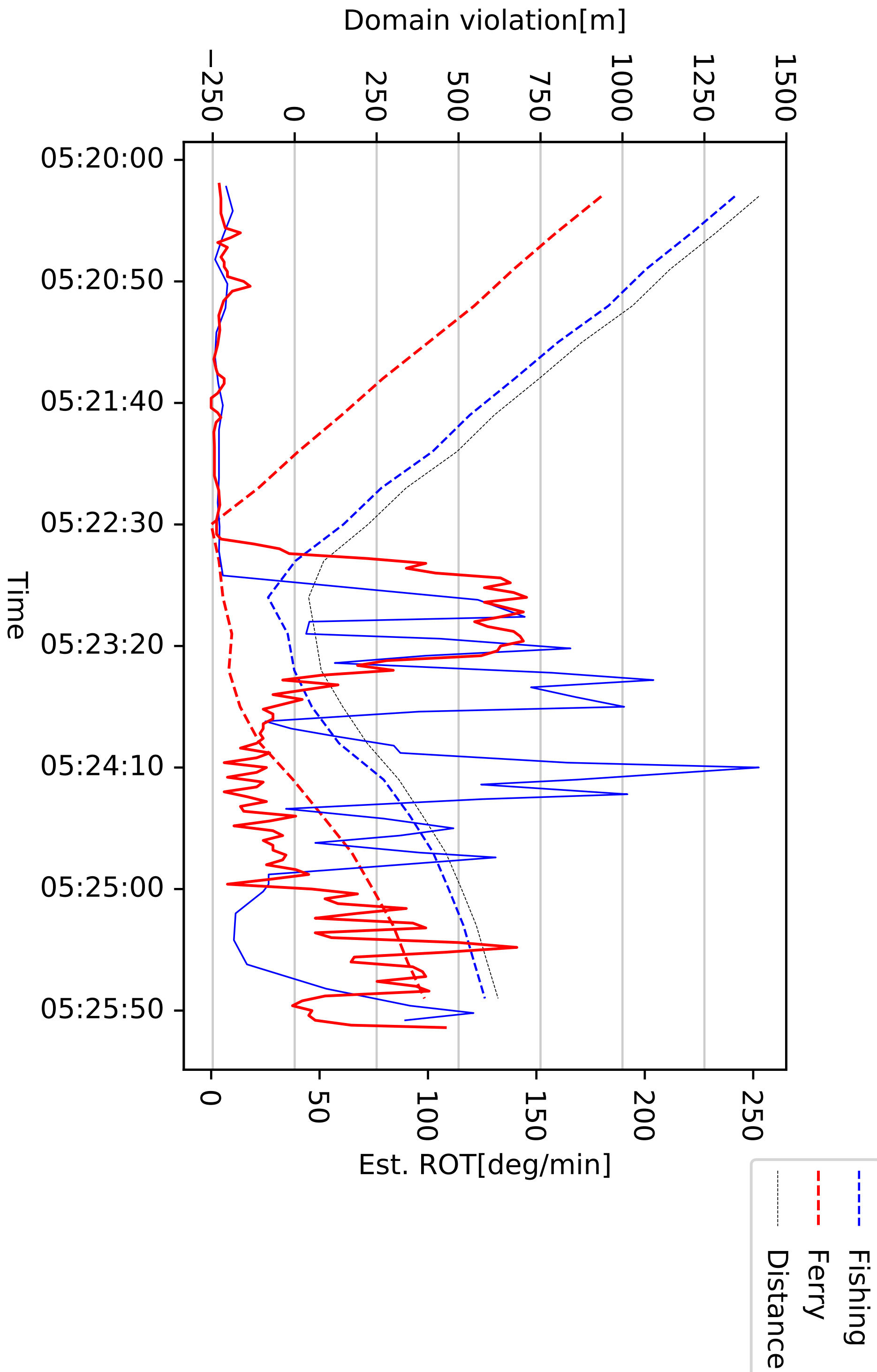


Figure 4.8: Timeseries plot of collision encounter between a fishing vessel and a RoPax ferry

The thin dashed line shows the distance between the vessels 3 minutes before until 2 minutes after CPA. The thick dashed lines shows the distance from ship domain boundary to the target vessel. This definition means that negative values signifies a domain violation. The solid lines shows the estimated rate of turn. From the graph one can see that the ferry had the fishing vessel around 250m inside the domain before it initiated evasive maneuvers and at this point it was to late. The Fishing vessel shows no intention to avoid a collision and the sudden change increase in estimated ROT are most likely the point of contact.

By looking at the distance from ship domain to target vessel for the ferry i.e. red dashed line figure 4.8, one can see that the domain violation was around 250 meters before initiating an evasive maneuver.

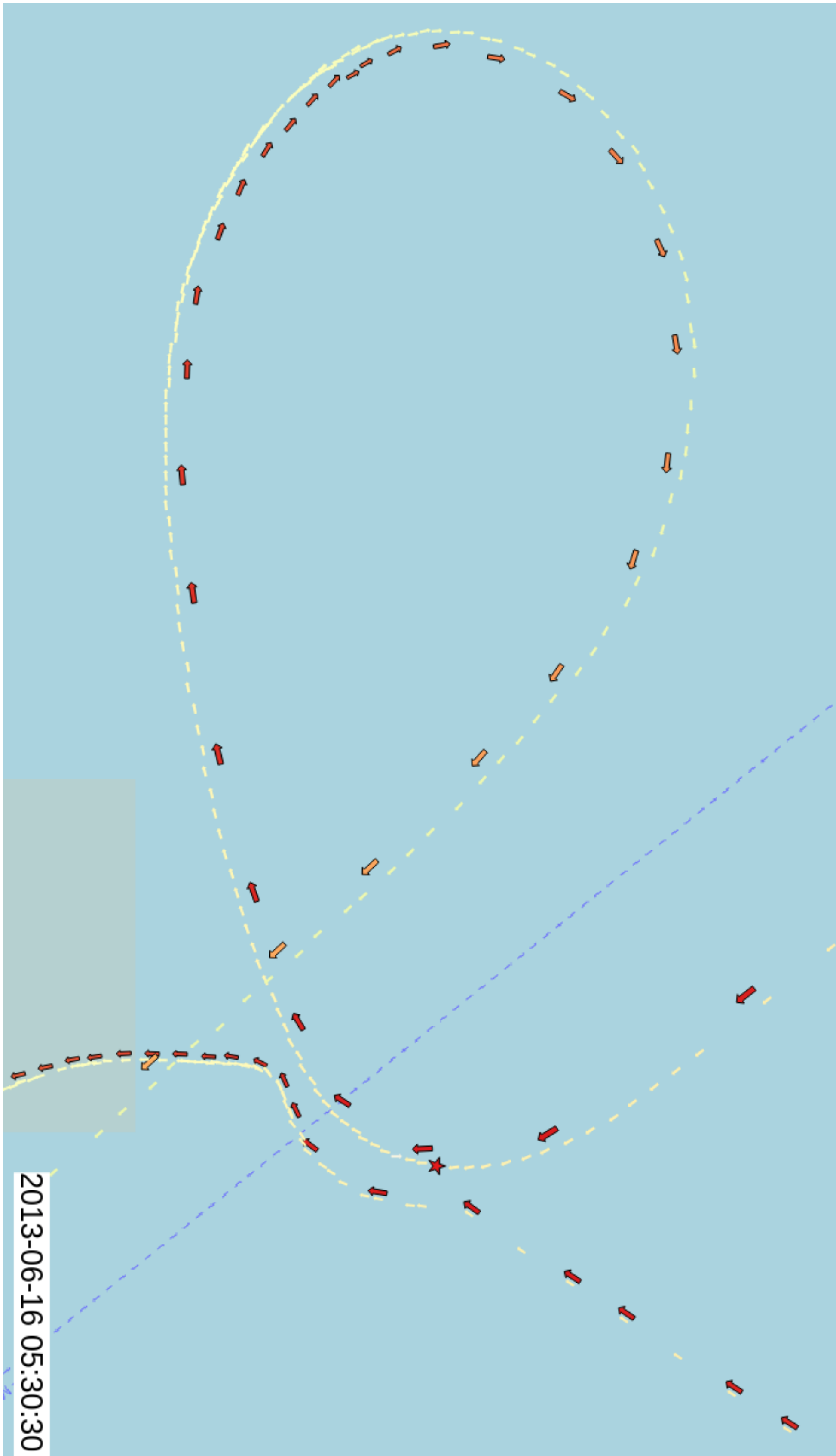


Figure 4.9: Large arrows are the conformed data while the smaller yellow arrows are raw data.

For two vessels on crossing courses like in this encounter, the vessel whose other ship is on their starboard side should keep clear and avoid passing ahead of said ship [3].

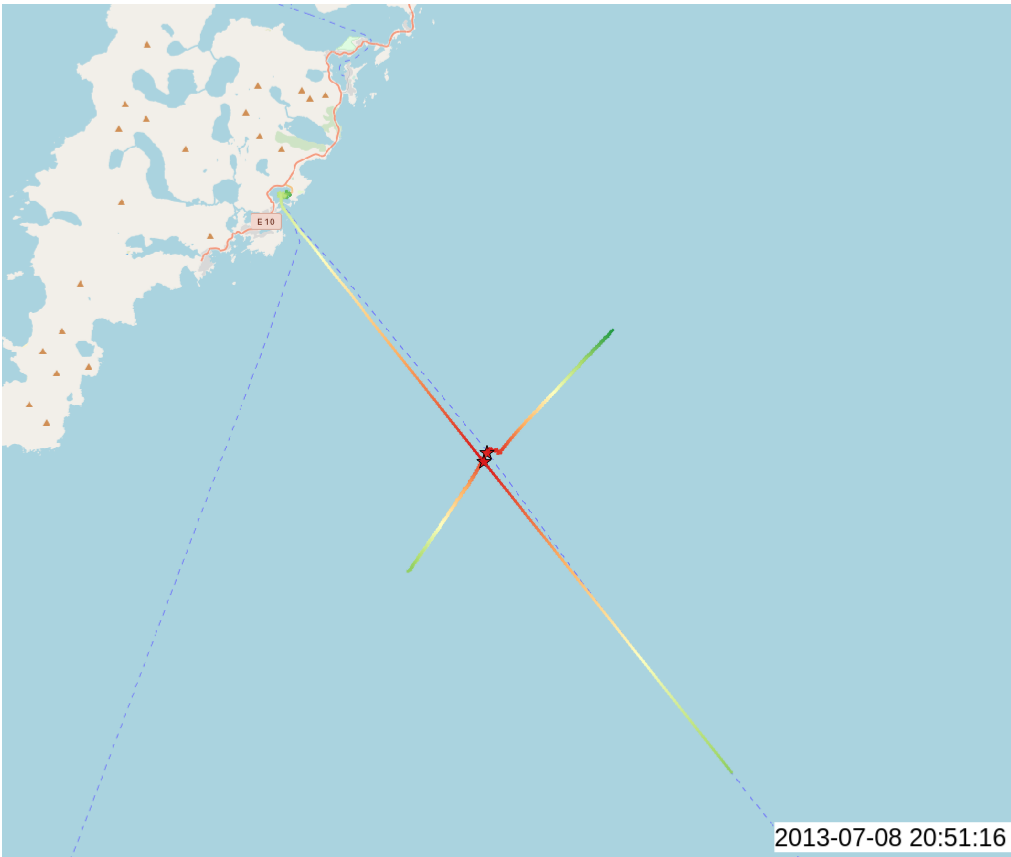
4.3.2 Encounter on the same ferry route

In a domain violation that happened around three weeks after the collision, another fishing vessel were on similar crossing course with a ferry, but closer to the departure port for the ferry see figure 4.10. With the ferry on the starboard side, the fishing vessel made a clear course change adhering to rule 15 and 16[4] in the international regulations for preventing collisions at sea.

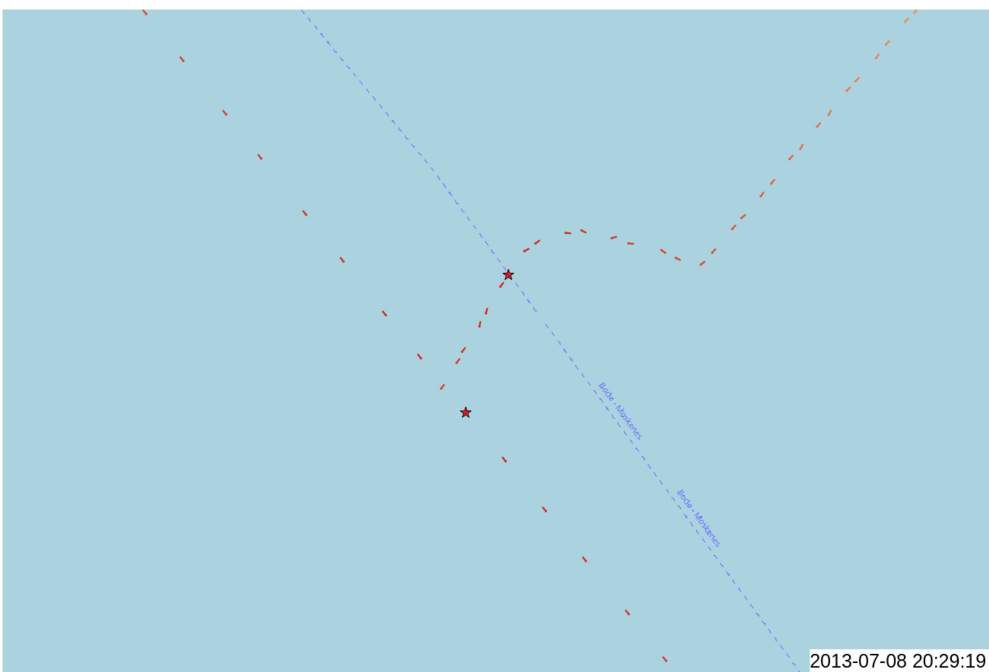
From the plot in figure 4.11 you can see that the domain violation is minimal for the ferry while the fishing vessels domain do not get violated. The solid line showing ROT indicates that it was a sudden and abrupt course change from the fishing vessel that also is visible in the AIS tracks in figure 4.10 (b).

[3] According to rule 15 in the international regulations for preventing collisions at sea https://lovdata.no/dokument/SF/forskrift/1975-12-01-5

[4] Every vessel which is directed to keep out of the way of another vessel shall, so far as possible, take early and substantial action to keep well clear.



(a) Showing the Ais track of an encounter similar to the collision



(b) Close up of the encounter, the star represents closest point

Figure 4.10: AIS tracks of a crossing encounter between a fishing vessel and a RoPax ferry under similar circumstances as the collision

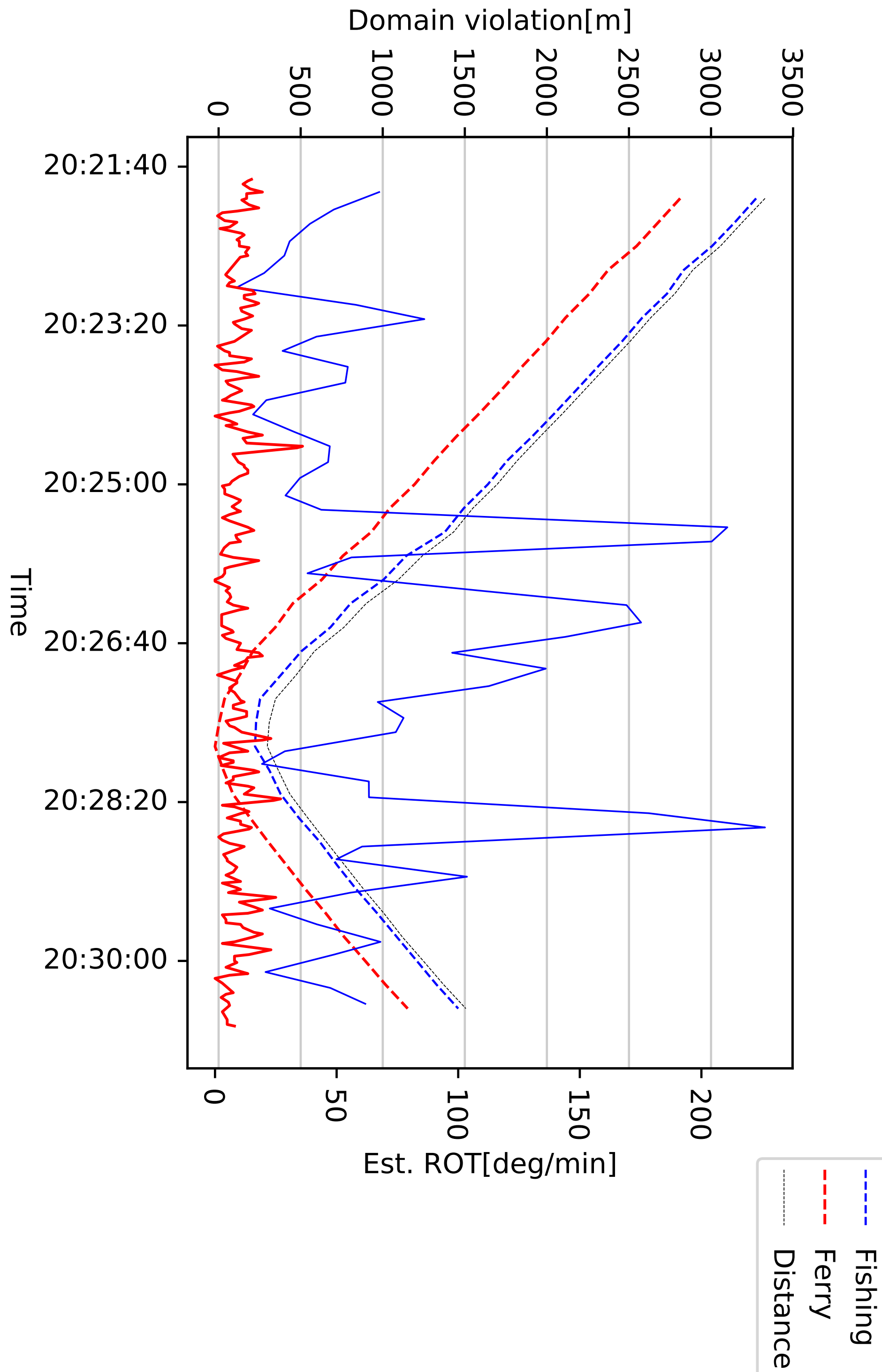


Figure 4.11: Timeseries plot of a crossing encounter between a fishing vessel and a RoPax ferry

4.3.3 Cruise ship crossings

Figure 4.12 shows two encounters between a cruise vessel and a RoPax ferry close to the port of Moskenes shortly after the first departure of the day for the ferry 07:00 am local summer time. The first encounter is marked with green tracks and shows a significant course change for the ferry to pass at the stern of the cruise ship. The red encounter occurred 3 weeks later and shows a more gradual course change compared to the first encounter. This can also be seen in figure 4.13, the red solid lines show the estimated ROT and reach a maximum value 103.5 during the first encounter and 64.5 during the second encounter for the ferry. The solid blue line shows small values of ROT which is consistent the straight tracks seen in figure 4.12. The blue and red dashed lines shows distance away from ship domain boundary to the center of target ship for the cruse ship and ferry respectively. Ship domain size in direction of target vessel can be determined by looking at the difference between the colored dashed line and the thin grey dashed line showing absolute distance between ship centers. It is therefor clear that the cruise vessel's ship domain is significantly larger than for the ferry, this is reasonable considering the cruise vessel are three time longer. Positive values for the red and blue dashed lines signifies no violation of the ship domain, where as negative values indicates domain violations. Looking at the dashed lines it is clear that it is only the cruise vessel that has its domain violated. With the current definition of exceptional encounters, nether of these encounters is classified as exceptional since the cruise vessel is not performing any evasive maneuvers. If the definition was changed to domain violation and either vessel initiating evasive maneuvers the first encounter would then be considered an exceptional encounter due to its maximum ROT exceeding the threshold of 70 deg/min.

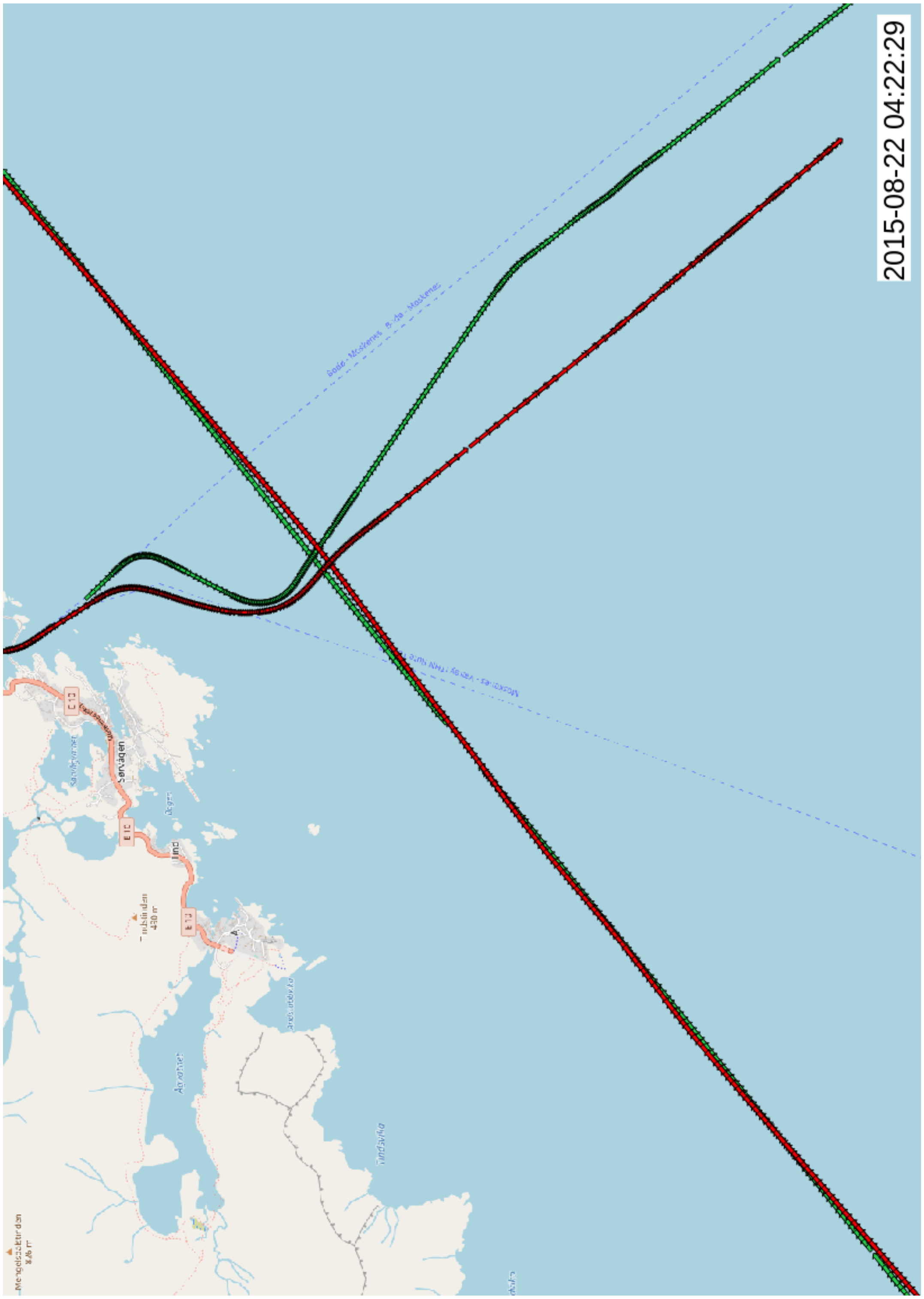
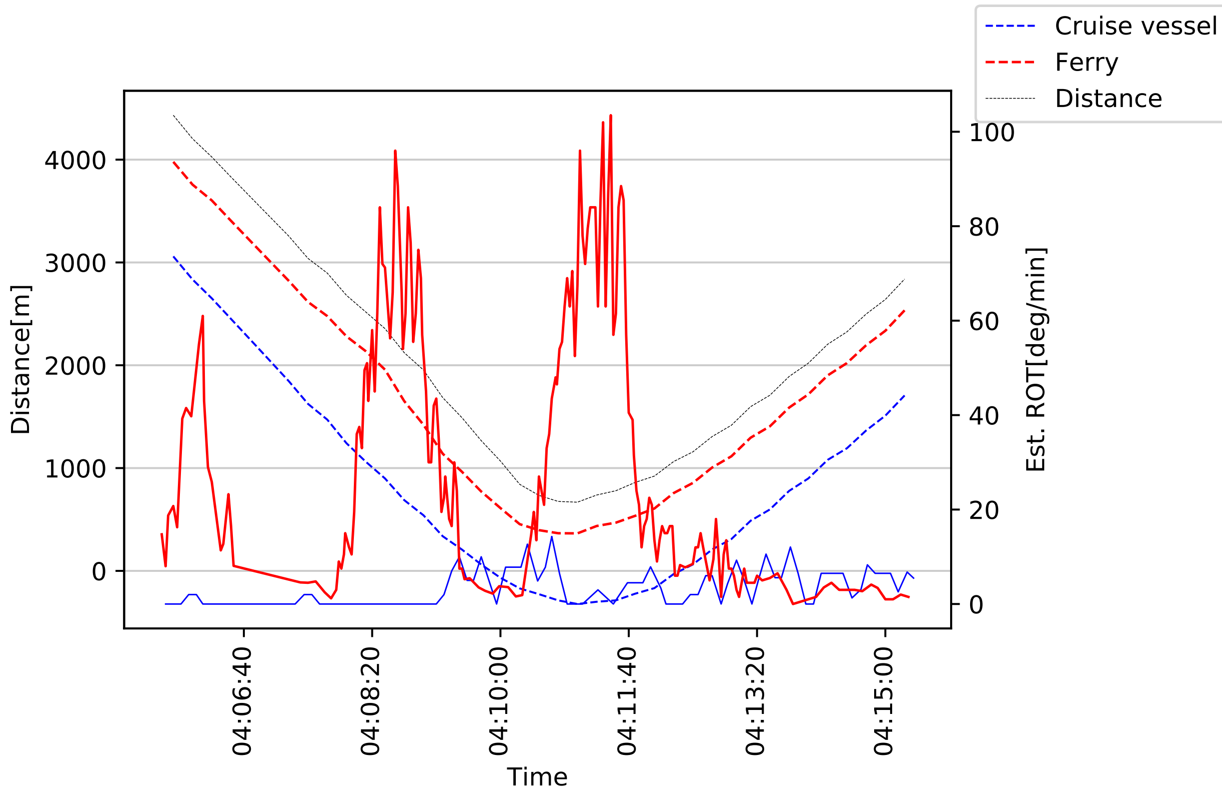
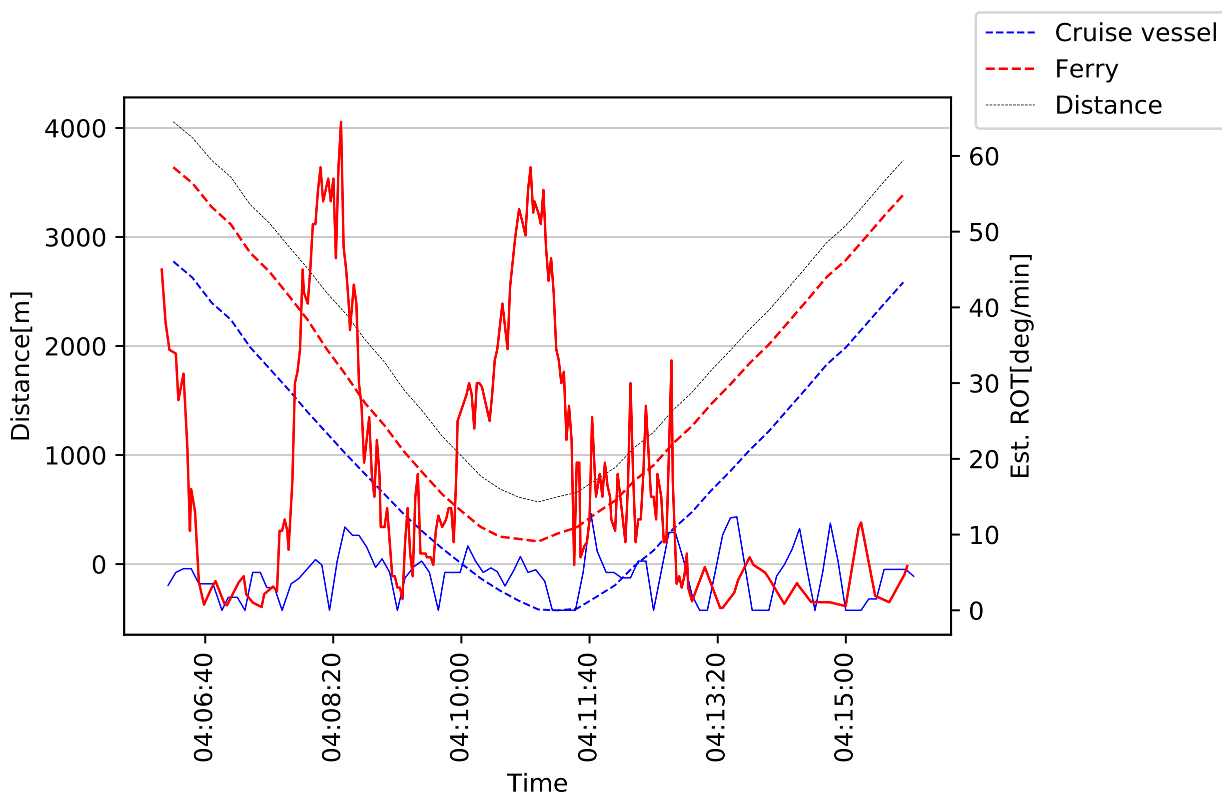


Figure 4.12: Two encounters between a cruise vessel and ferries leaving Moskenes



(a) 10 minute timeseries corresponding to green tracks in figure 4.12



(b) 10 minute timeseries corresponding to red tracks in figure 4.12

Figure 4.13: Timeseries plot of encounters between cruise ship and RoPax ferry

4.3.4 Weird fishing vessel behaviour

From table 4.4 it is clear that most of the sorted encounters are between two fishing vessels. In figures 4.14 - 4.17 four encounters are displayed were the two fishing vessels are moving in patterns that appear to be random. In all the figures darkening of color translate to the passing of one hour, the same color for both tracks in each figure determines the position for both vessels at that time instance.

These are examples of typical vessel movements between fishing vessels which often get classified as exceptional encounters. A timeseries plot of example four can be seen in figure 4.18. It can be seen in the figure that ROT of above the 70 deg/min threshold is exceeded several times in this 20 min period centered at the time of domain violation. The colored dashed lined showing distance away from ship domain boundary shows that only fishing vessel 1 have its domain violated. In this plot the length of fishing vessel 2 have manually been set to a correct length of 28 meters, in the domain violation search of step 2 a length of 40.001 meters is used due to missing data. With a length of 40.001 meters both vessels have their ship domain violated. Due to several estimations of ROT above the threshold it is classified as an exceptional encounter.

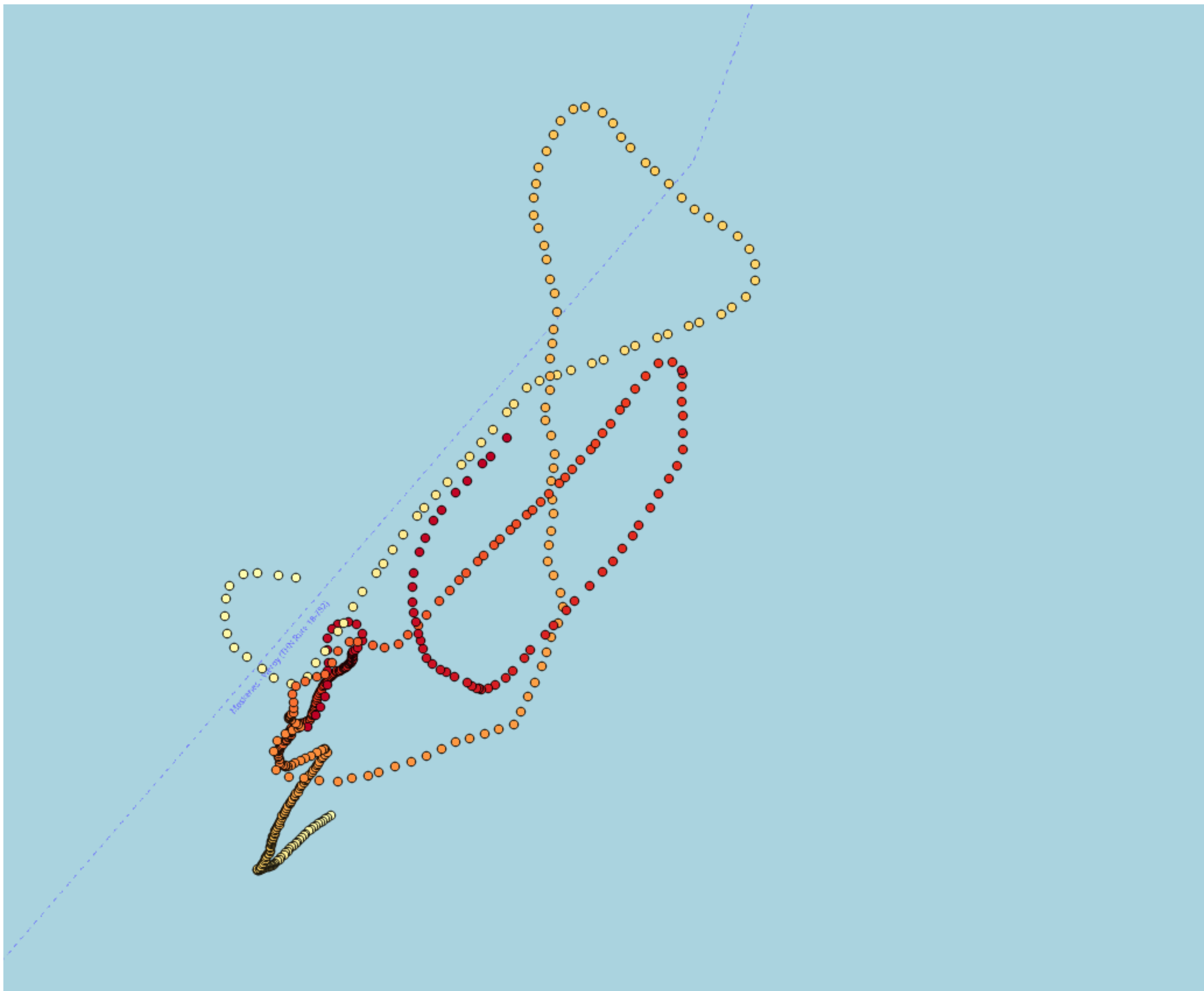


Figure 4.14: Irregular fishing movements example one

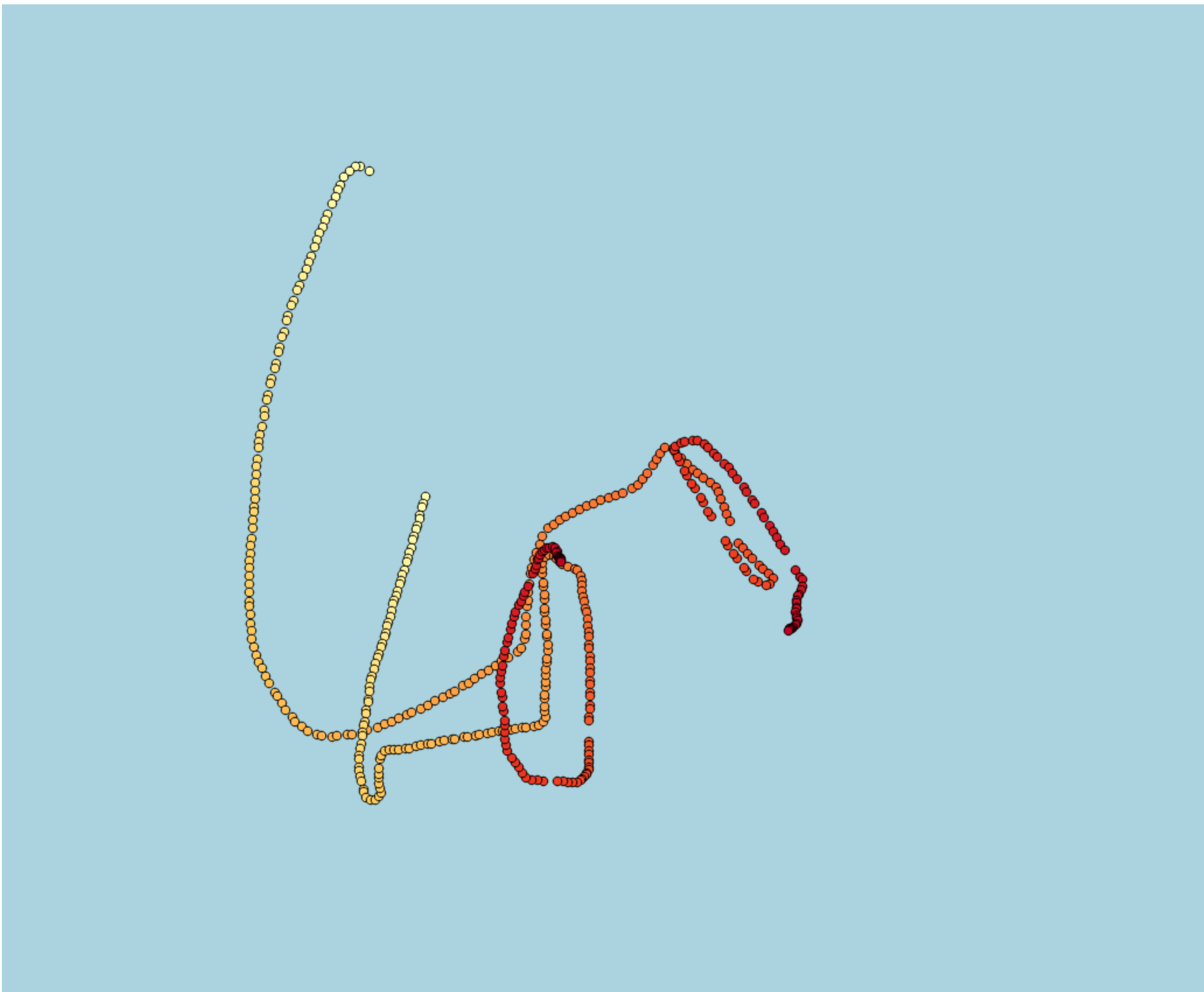


Figure 4.15: Irregular fishing movements example two

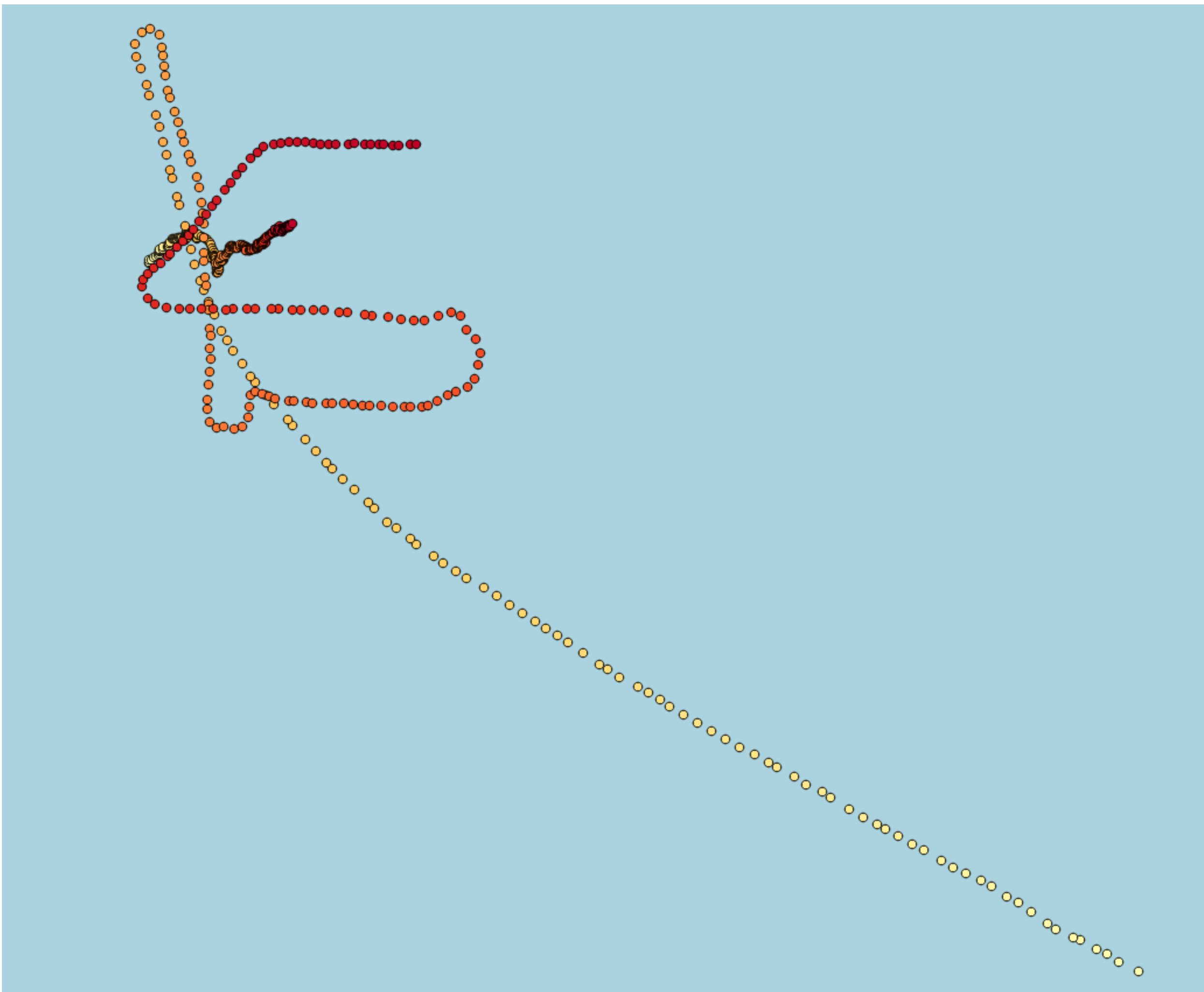


Figure 4.16: Irregular fishing movements example three

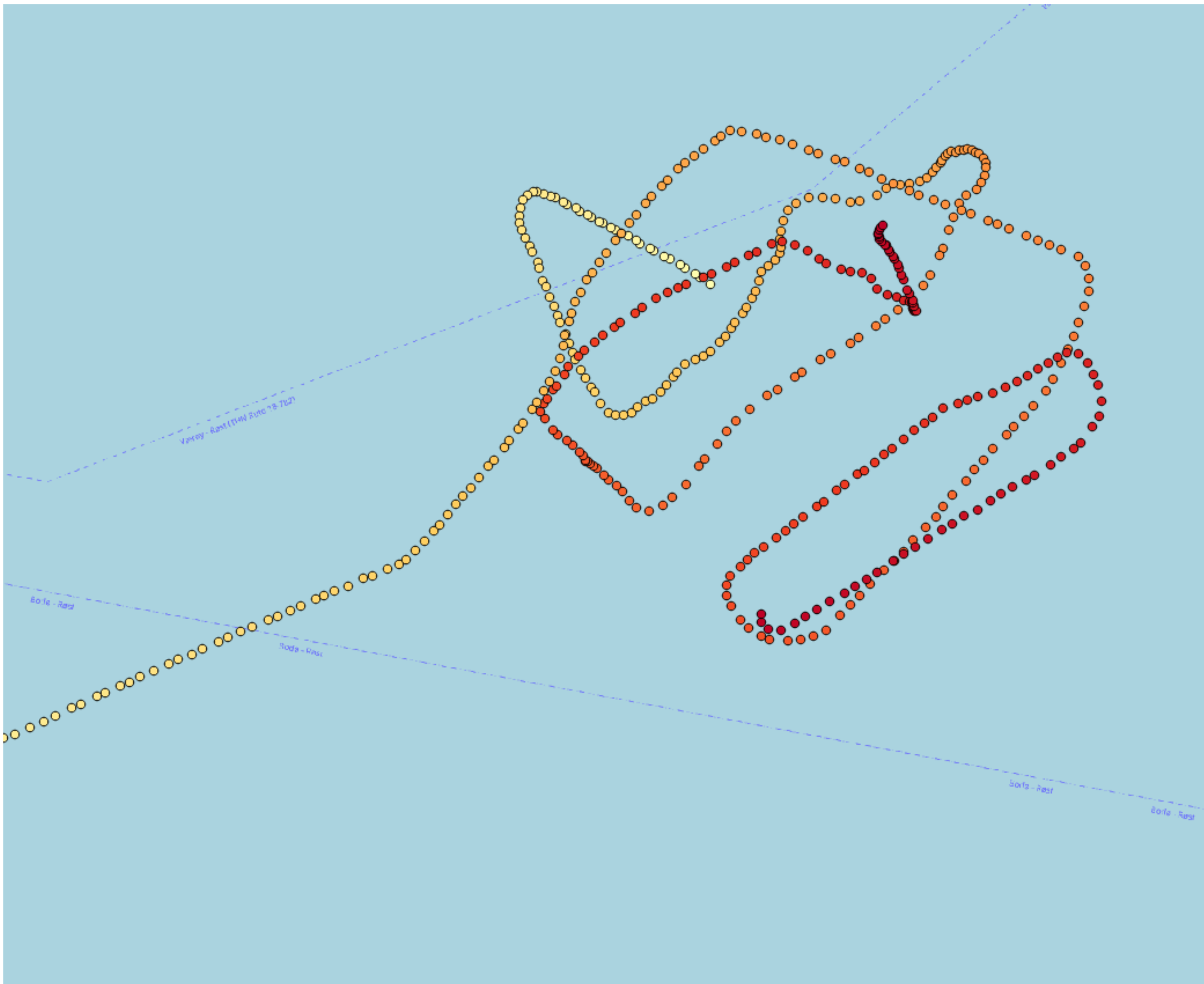


Figure 4.17: Irregular fishing movements example four

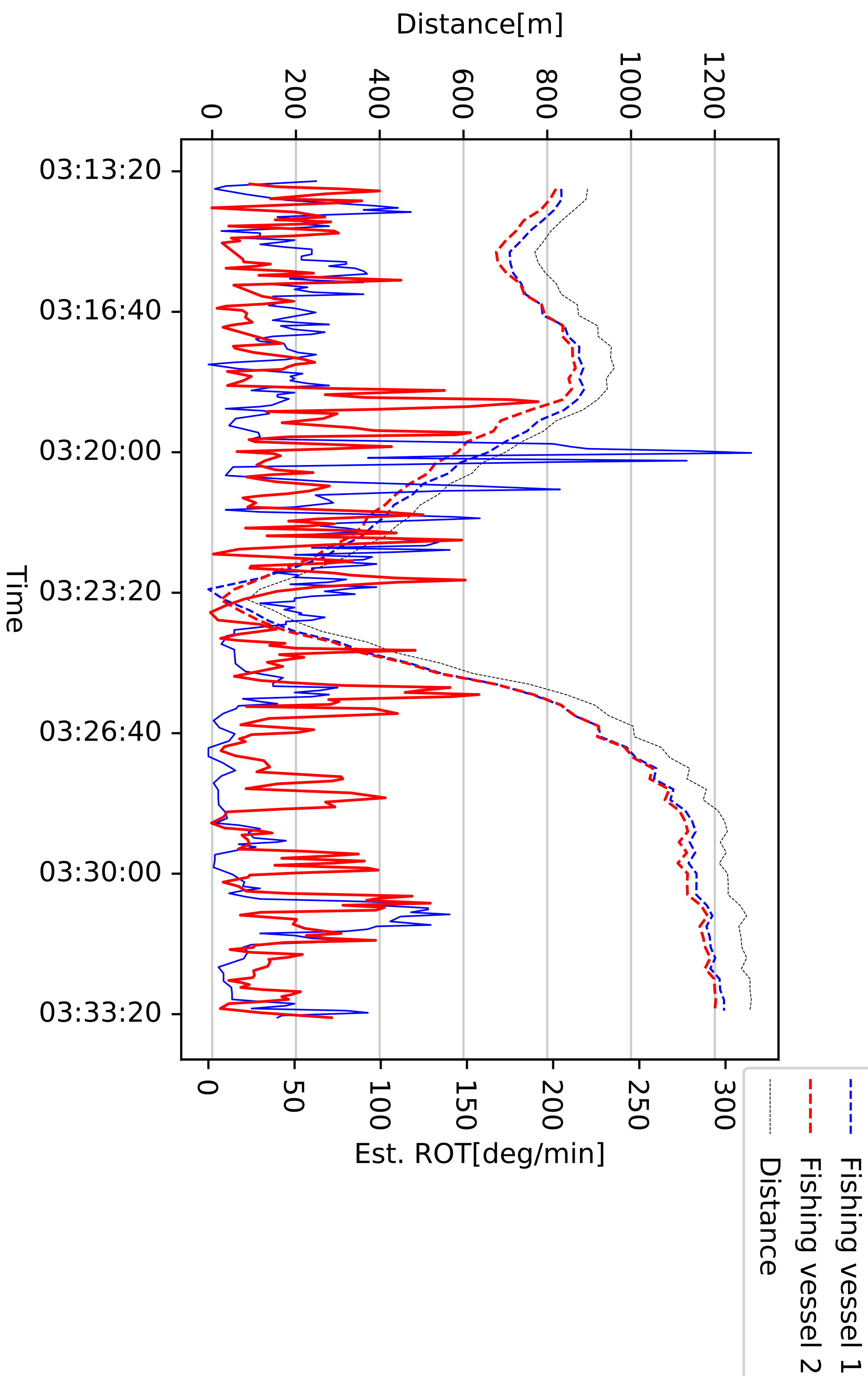


Figure 4.18: Timeseries plot of irregular fishing vessel movements in example four

4.3.5 Erratic AIS data

From the distances between messages shown figure 4.20 it is clear that either the time code or coordinates are erroneous. It is also visible on AIS track(figure 4.19), where the color darkens as time progresses, a timeseries without flaws would show a smooth color gradation. As seen in figure 4.21 there would not have been a violation with flawless data. The general trend in distance shows that the minimum distance away from domain boundary for the cargo vessel is around 1500 meters, with the erratic data a domain violation of 237 meter is detected at 13:59:45.

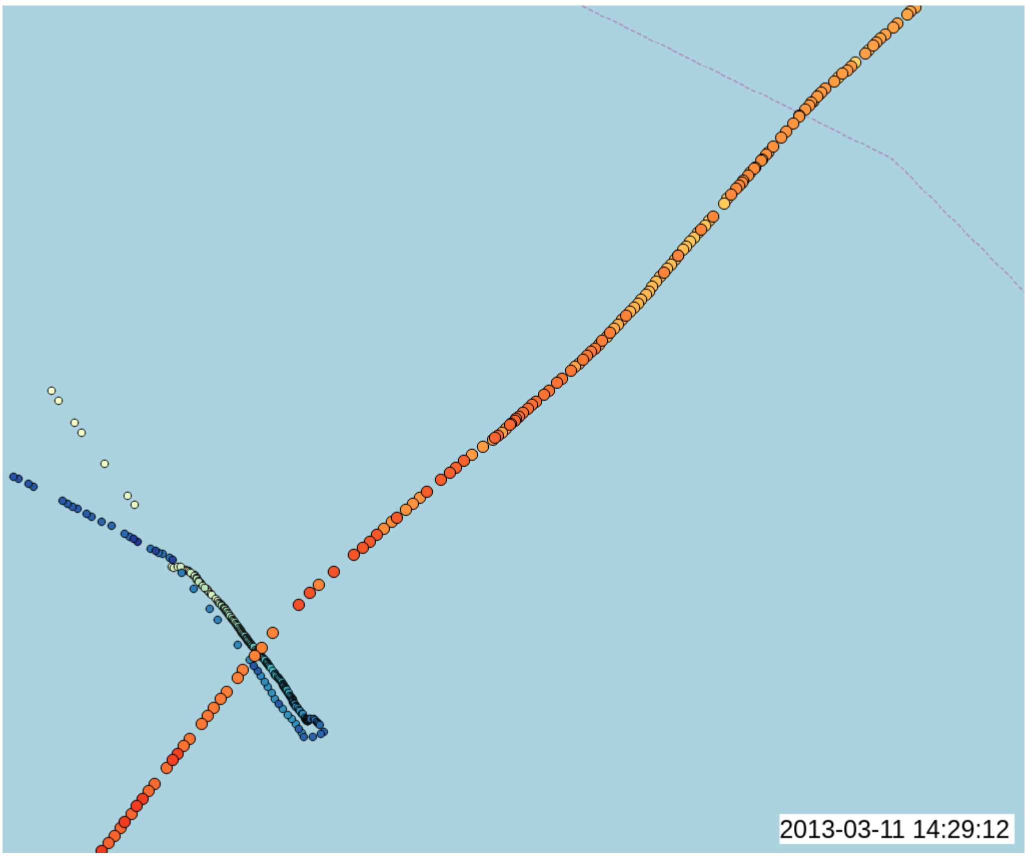


Figure 4.19: AIS tracks

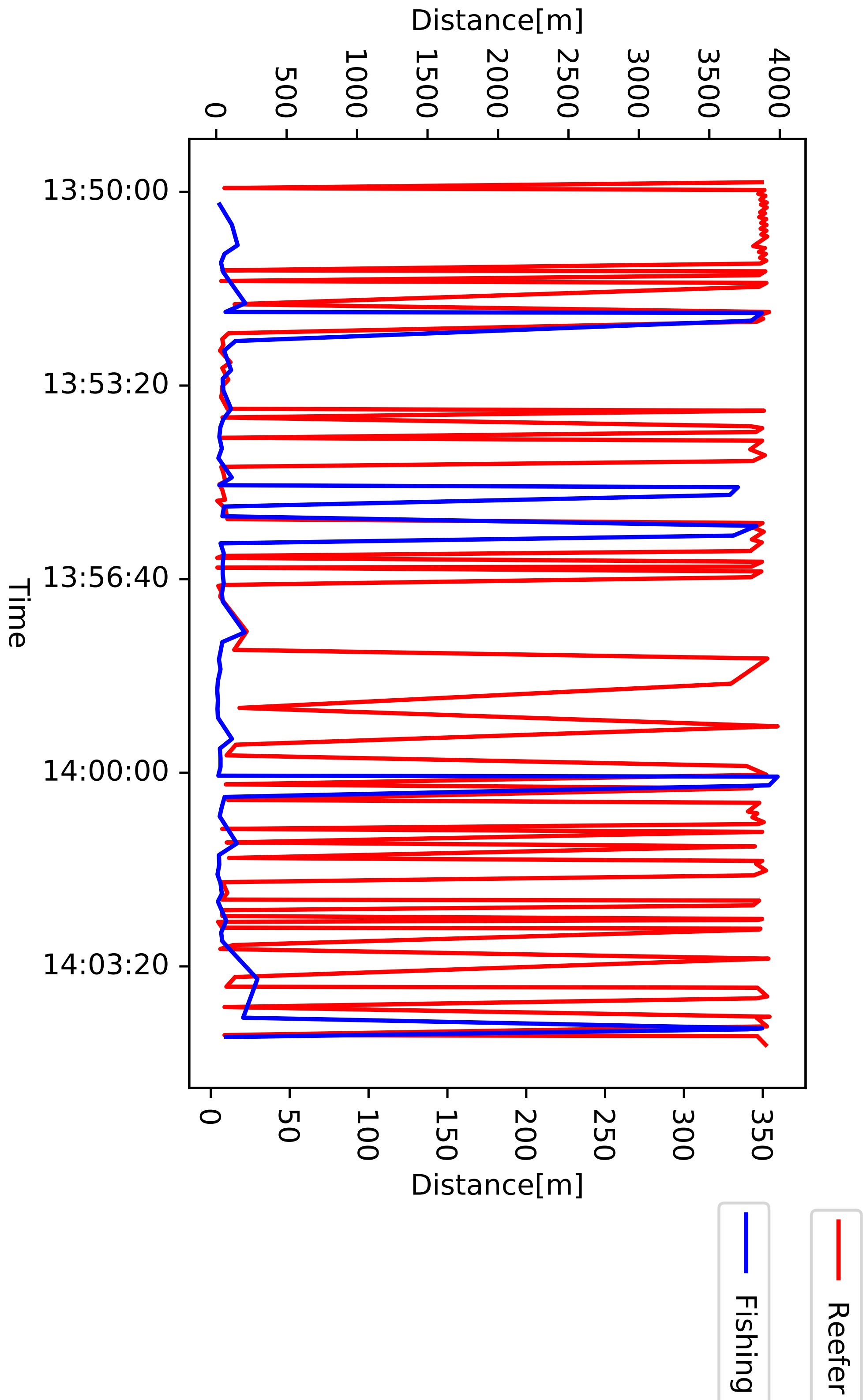


Figure 4.20: Plot of erratic AIS messages showing the distance between each messages, the left scale corresponds to the reefer ship while the right scale corresponds to the fishing vessel

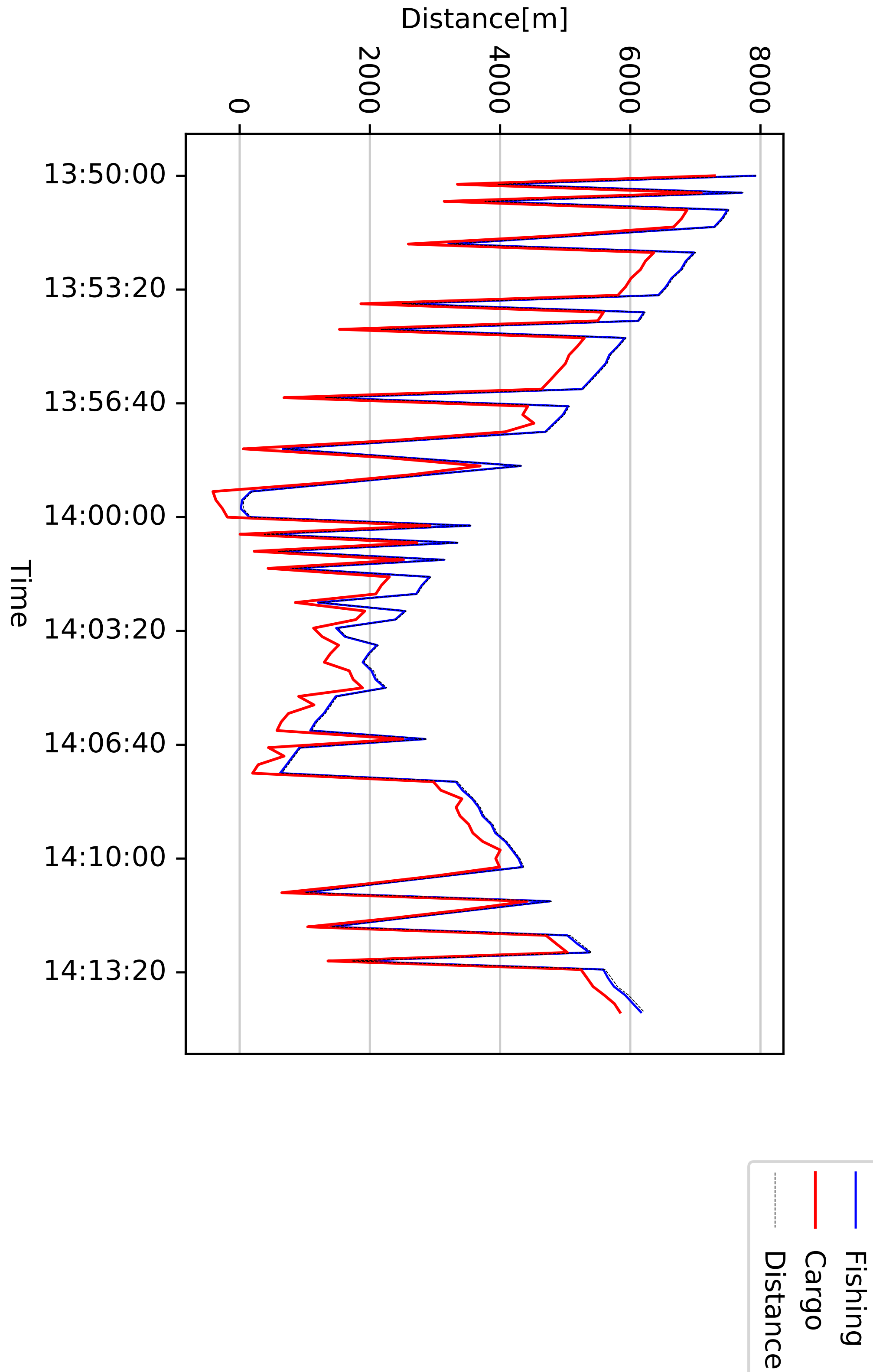


Figure 4.21: Timeseries plot of encounter of erratic data leading to false positive domain violation detection.

4.4 Initial Risk Assessment

In the case study a total of 707 ship domain violations are detected, from this list 381 encounters are classified as exceptional by passing the sorting algorithm described in section 3.5, see table4.4 for yearly distribution.

Table 4.4: Model output from Vestfjorden 2013-2016

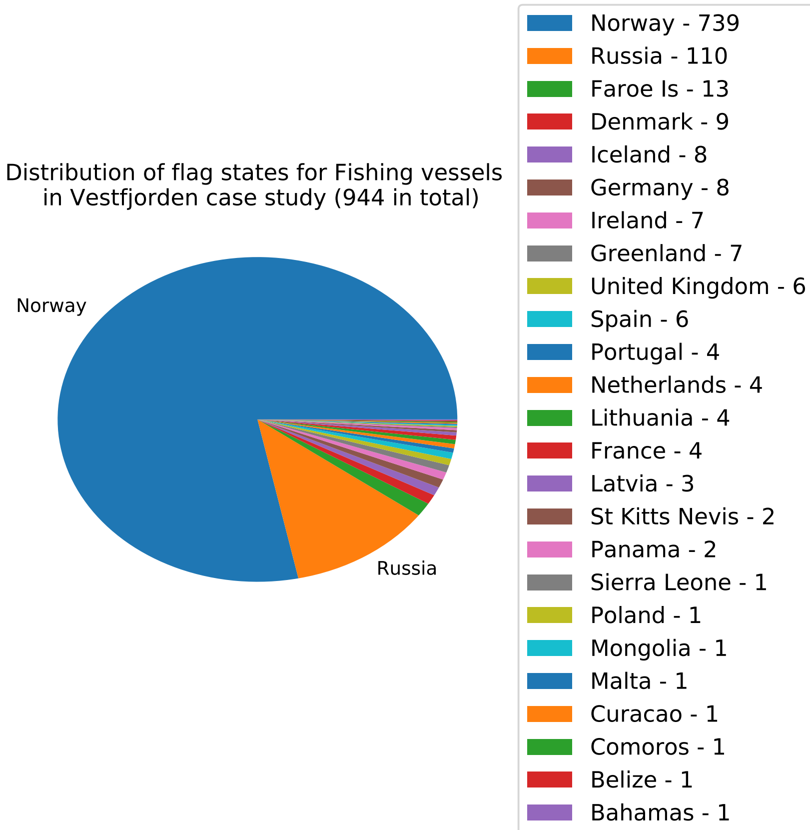
Ship domain violations				
Year	2013	2014	2015	2016
Time for domain violation search [hours]	9.4	10.2	12.8	15.4
Number of detected domain violations	243	152	210	102
Number of encounters passing the sorting algorithm	133	70	115	63
Number of encounters passing sorting algorithm and where both are fishing vessels	83	58	71	56

From table 4.4 it is clear that most encounters are between two fishing vessels. It is therefore reasonable to differentiate between all encounters and sub groups where both vessels are fishing vessels when normalizing the results. It is further found to be interesting to compare frequencies for Norwegian and Non-Norwegian vessels. In this respect excluding encounters where both vessels are fishing vessels become more important. Looking at the share of fishing vessels in both number and distance(see figure 4.22), the frequency of encounters would be unreasonably high for Norwegian vessels due to the higher share of fishing vessels.

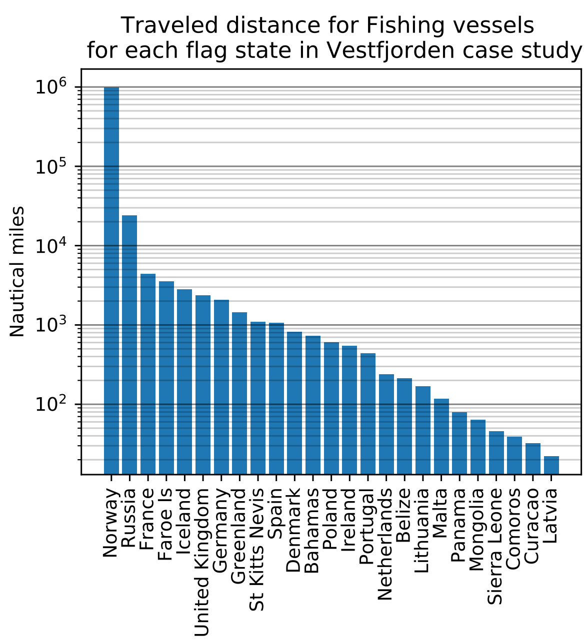
From table 4.6 it is clear that the difference in normalized frequency for Norwegian and Non-Norwegian vessels is significant. The normalized frequency is calculated by counting the number of vessels from each group involved in encounters and diving by the total length or number of trips that group have combined. From the three groups of encounters in table 4.8, encounters between two Norwegian vessels will be counted twice for Norwegian vessels while encounters between a Norwegian vessel and a Non-Norwegian vessel will be count once in each group. Similarly encounters between to Non-Norwegian vessels will be counted twice for the Non-Norwegian vessels.

Table 4.5: Model output from Vestfjorden 2013-2016

Type of domain violation				
Year	2013	2014	2015	2016
Number of Head-on encounters	95	52	105	16
Number of Overtaking encounters	57	28	18	27
Number of Crossing passing at bow	55	38	49	33
Number of Crossing passing at stern	36	34	40	26



(a) Number



(b) Distance

Figure 4.22: Distribution of fishing vessels

Table 4.6: Normalized ship domain violations Vestfjorden 2013-2016

Normalized ship domain violations		
	Per 10^5 nautical miles	per 10^3 trip
Violations for Norwegian Vessels	66.28	7.49
Violations for Non-Norwegian vessels	7.17	3.05
Sorted encounters for Norwegian Vessels	36.12	4.08
Sorted encounters for Non-Norwegian vessels	2.84	1.21
Excluding encounters where both vessels are fishing vessels		
Violations for Norwegian Vessels	39.83	6.49
Violations for Non-Norwegian vessels	4.98	2.29
Sorted encounters for Norwegian Vessels	9.63	1.57
Sorted encounters for Non-Norwegian vessels	2.10	0.96
Excluding encounters between vessels on the ferry service and where both vessels are fishing vessels		
Sorted encounters for Norwegian Vessels	3.77	0.63

Table 4.7: Model output from Vestfjorden 2013-2016

Ship domain violations	
Ship domain violations between Norwegian and Norwegian	657
Ship domain violations between Norwegian and Non-Norwegian	38
Ship domain violation between Non-Norwegian and Non-Norwegian	12
Sorted encounters between Norwegian and Norwegian	360
Sorted encounters between Norwegian and Non-Norwegian	15
Sorted encounters between Non-Norwegian and Non-Norwegian	6

When normalizing for all encounters see top part of table 4.6, the difference in frequency between Norwegian and Non-Norwegian vessels is around an order of magnitude greater when normalizing for distance, for both sorted and unsorted encounters. When normalizing for number of trips the frequency ratio between Norwegian and Non-Norwegian decreases. This is the result of Norwegian and especially Norwegian fishing vessels having significantly sorter average trip length (see table 4.9). The large difference in average length is probably caused by Non-Norwegian vessel traveling trough the area in order to reach Narvik harbor, while Norwegian vessel are to a large degree moving inside the area.

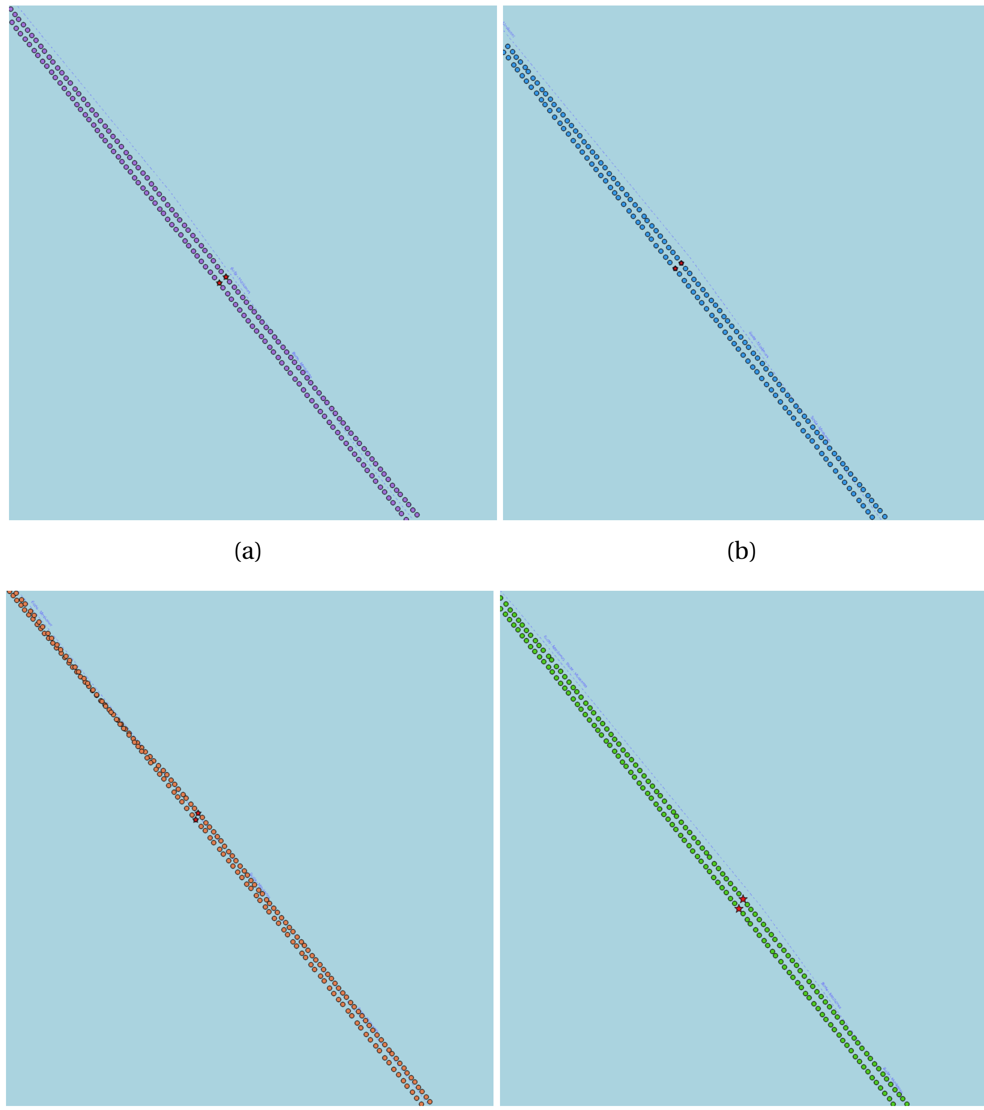
Table 4.8: Model output from Vestfjorden 2013-2016

Ship domain violations excluding those with two fishing vessels	
Ship domain violations between Norwegian and Norwegian	202
Ship domain violations between Norwegian and Non-Norwegian	18
Ship domain violation between Non-Norwegian and Non-Norwegian	11
Sorted encounters between Norwegian and Norwegian	47
Sorted encounters between Norwegian and Non-Norwegian	8
Sorted encounters between Non-Norwegian and Non-Norwegian	5

Table 4.9: Average length per trip for different groups in Vestfjorden 2013-2016

Average length per trip for different groups	
	Nautical miles
Average trip length Norwegian vessels excluding fishing vessels	16.3
Average trip length Non-Norwegian vessels excluding fishing vessels	45.9
Average trip length for fishing vessels	8.7
Average trip length excluding fishing vessels	22.3

The high values for Norwegian vessels in table 4.6 after excluding fishing vessels comes from a significant number of encounters where RoPax ferry's meet between ports. There are 186 encounters where 185 are categorized as head-on, the last one is 1.2° under the head-on threshold see table 3.3. The sorting algorithm that estimates ROT during encounters reduces the number to 38. Looking at AIS tracks of the sorted encounters reveals that they are predominantly on parallel tracks with small or none course alterations see examples in figure 4.23. These are result of noise in the data, with the relatively high speed and corresponding low reporting interval (IMO, 2002a) the inaccuracies in COG are exaggerated by small time-delta between messages.



(a) (b) (c) (d)

Figure 4.23: Four examples of meeting ferries falsely registered with high rate of turn

If these encounters are omitted the sorted frequency is reduced to 3.77 encounters per 10^5 Nautical mile and 0.63 per 10^3 trip that are more in line with Non-Norwegain vessels see bottom row of table 4.6.

Figure 4.24 show all domain violations detected in the case study and there are some special things to take note of. The line of green symbols between Bodø and Moskenes are predominantly between ferries operating the same route, and is assumed to be a result of overestimation of domain size during good sea conditions. The string of yellow symbols follow the ferry route Værøy - Røst – Moskenes and shows that encounters between the ferry service and fishing vessels are quite common on this part of the route. The blue symbols which are encounters

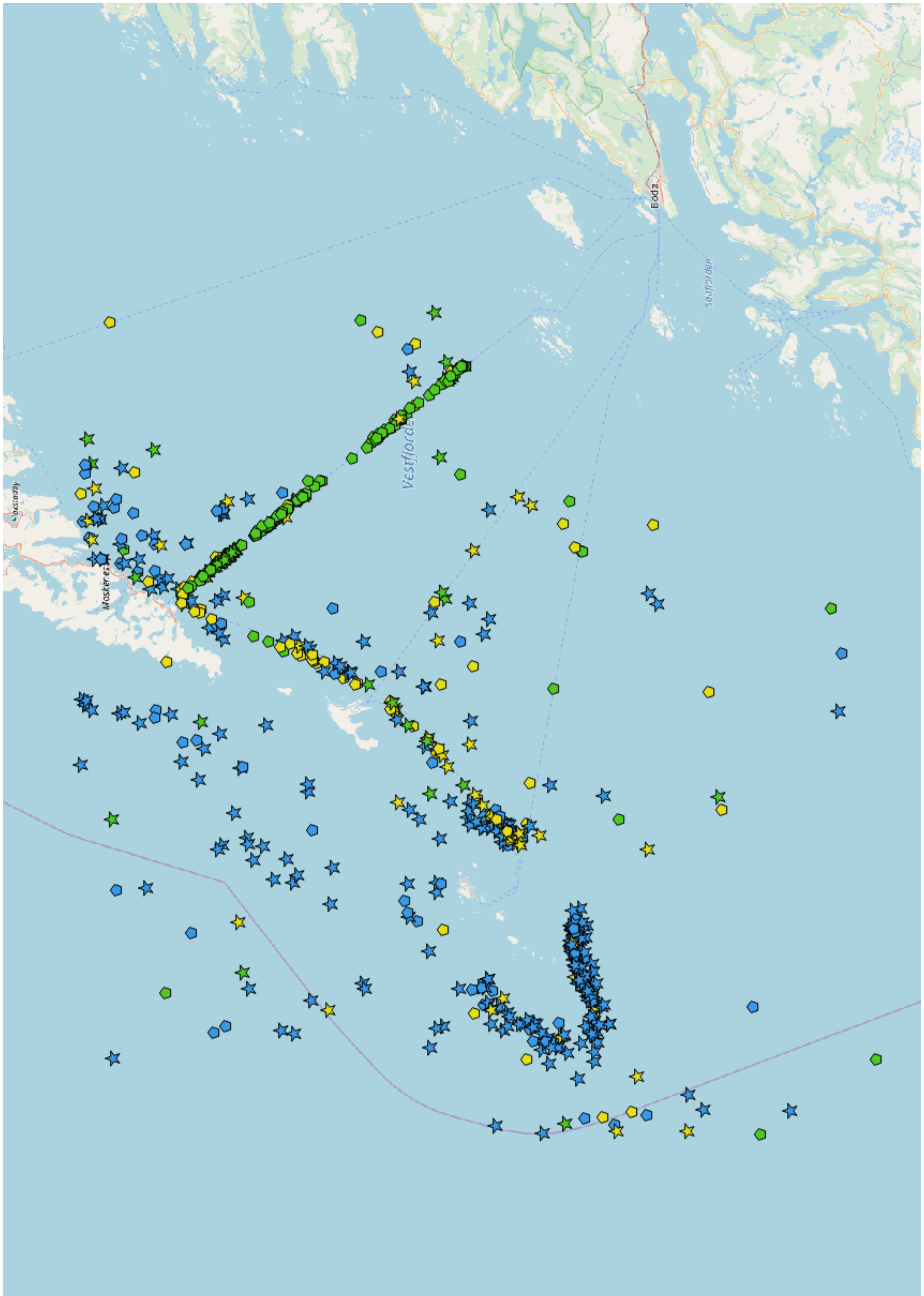


Figure 4.24: All registered violations in the case study, blue symbols are between two fishing vessels, green are without fishing vessels and yellow are with one fishing vessel, lastly stars represent encounters with high ROT.

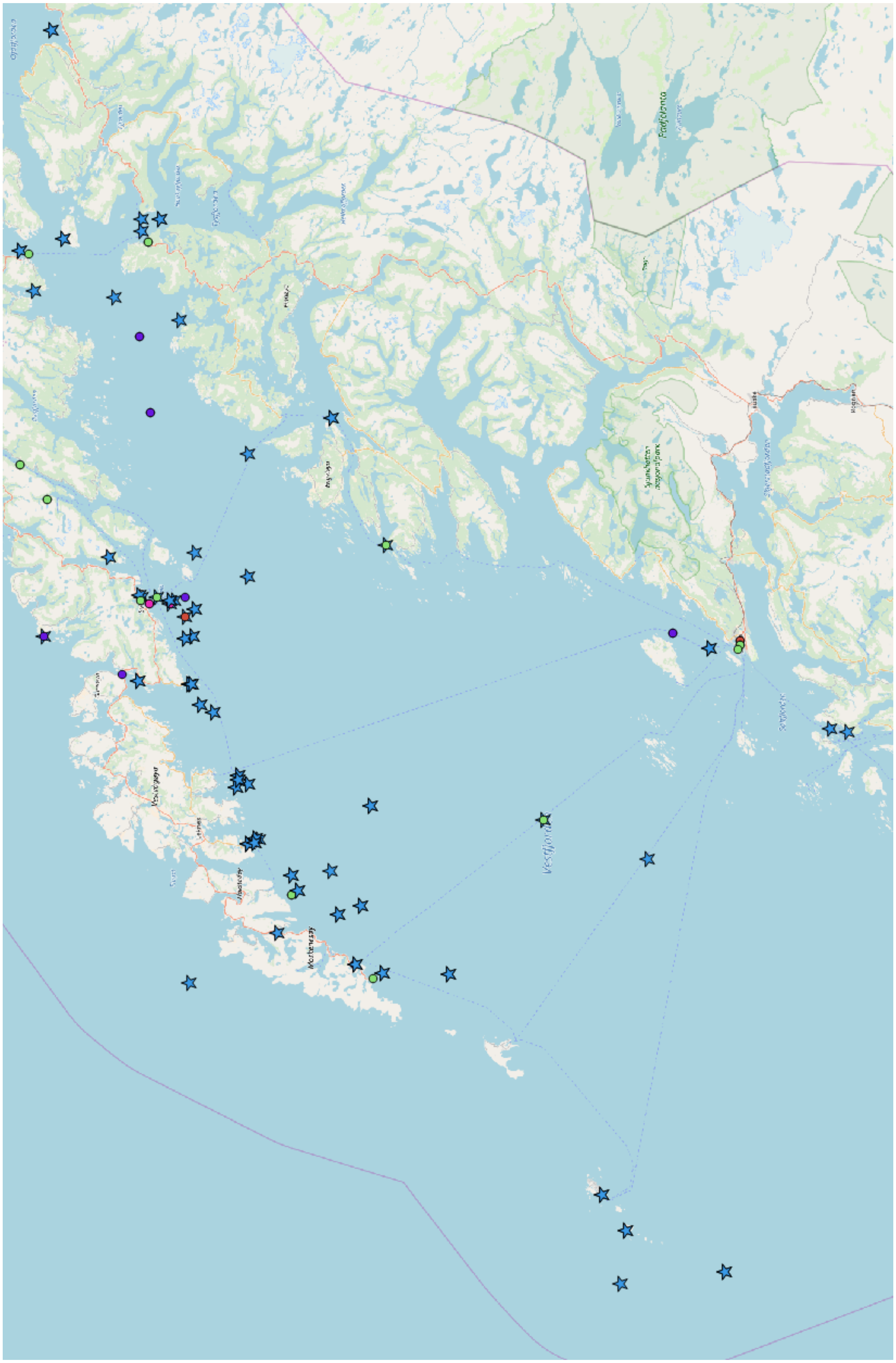


Figure 4.25: All collisions regardless of area type registered by the Norwegian Coastal Administration

between fishing vessels are scattered over large parts of the case study area. Some hot spot can be found concentrated around Lofoten and especially Røst, where there presumably are good fishing areas(see figure 4.26).

Figure 4.25 shows all collisions since 1981 where fishing vessels are marked with blue stars. From the figure it is clear that most of the registered collisions are between fishing vessels, and it is therefor reasonable that most of the registered domain violations from the model includes at least one fishing vessel. Further it seams that the intensity of violations at the tip of Lofoten (see figure 4.26) does not translate to a large number of collisions.

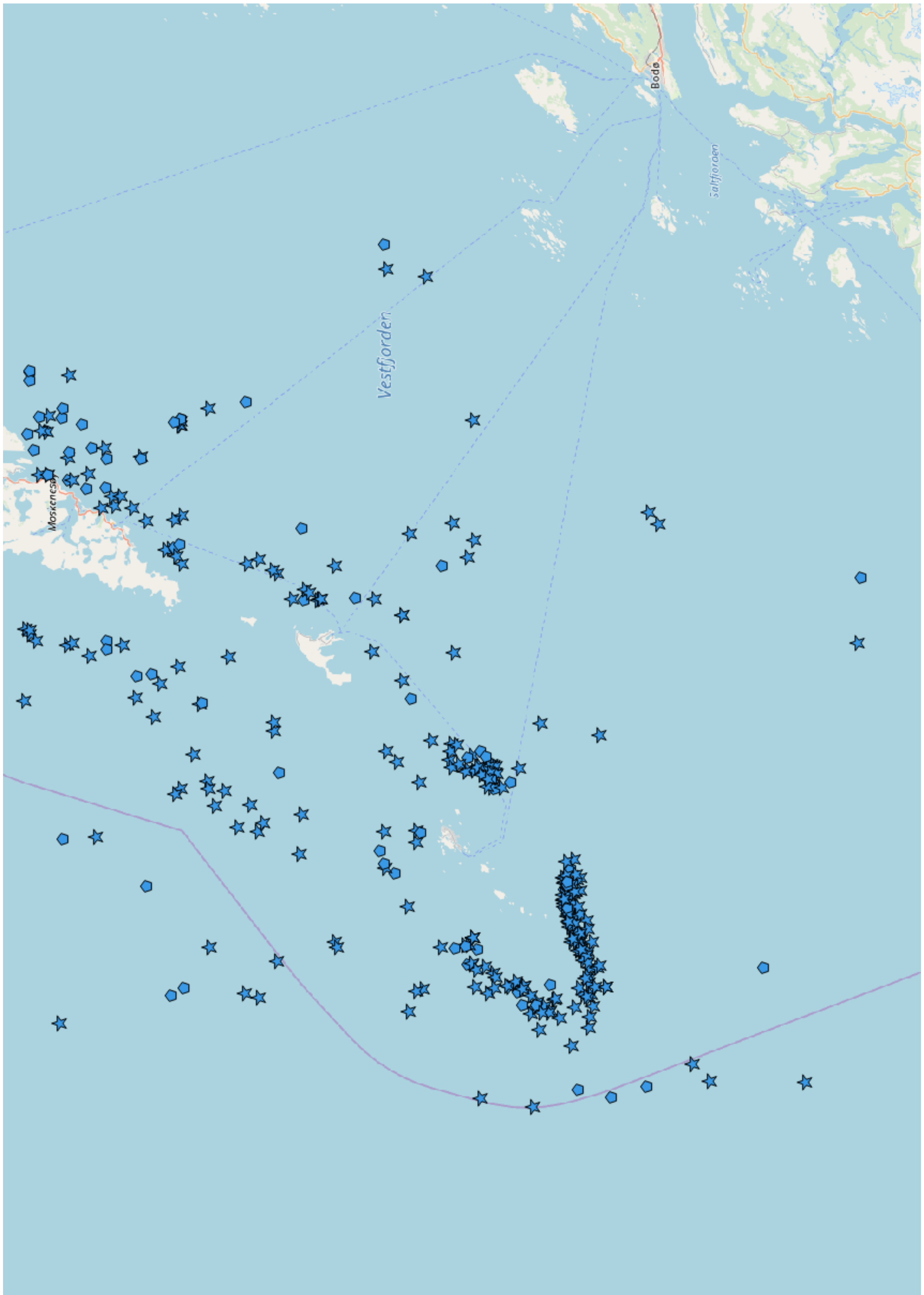
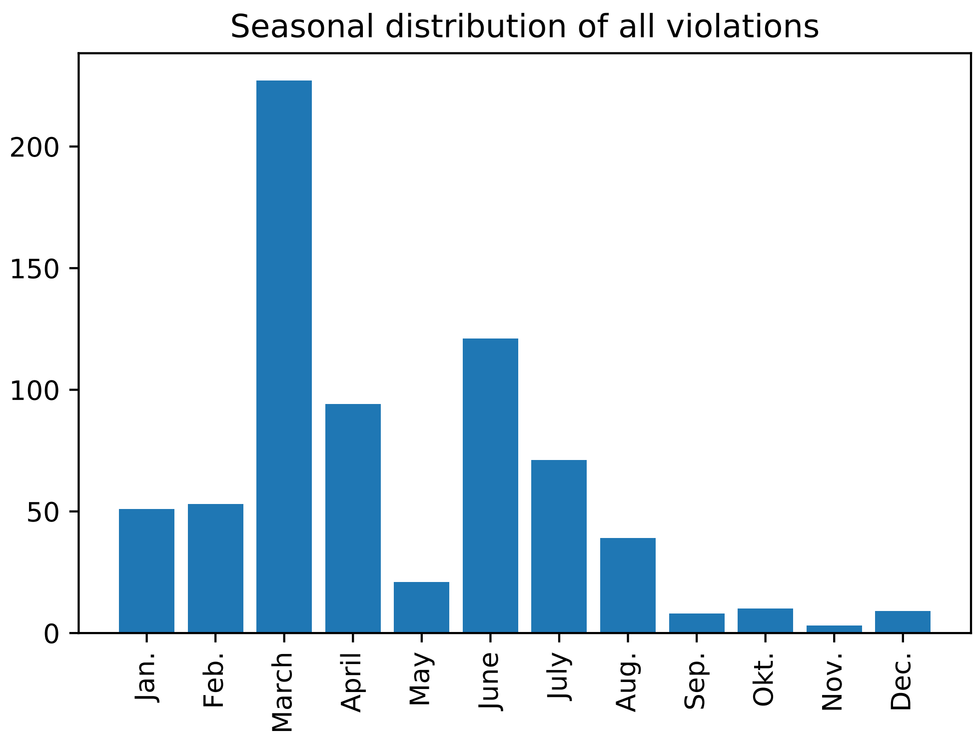


Figure 4.26: All registered violations in the case study between two fishing vessels, stars represent encounters with high ROT.

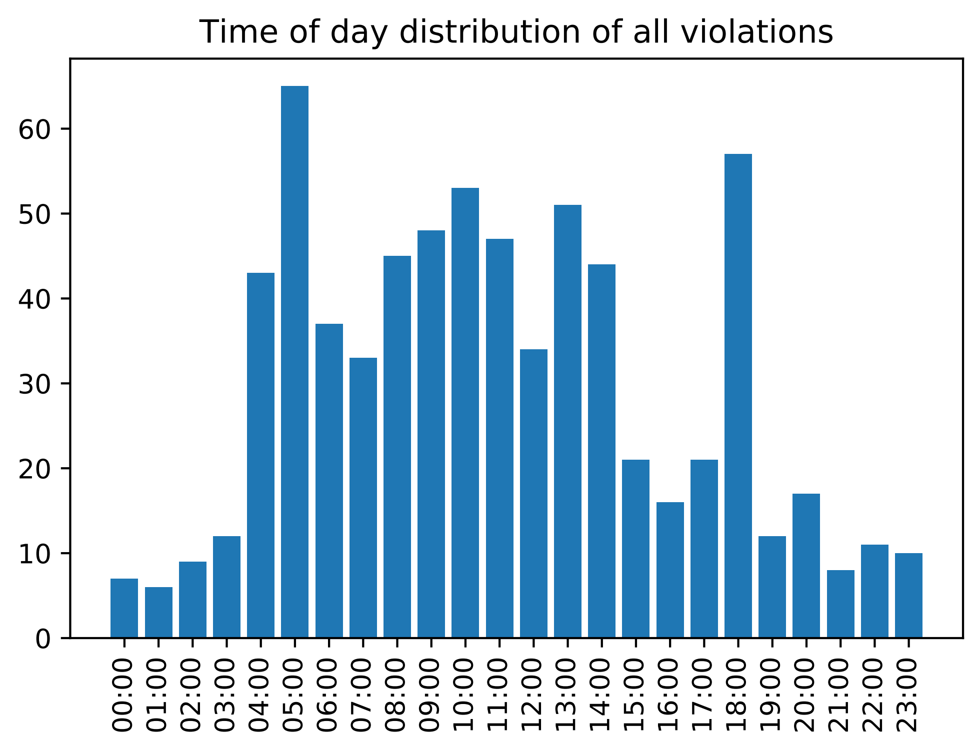
4.4.1 Temporal distributions of domain violations

From figure 4.27 (a) it is clear that the distribution of domain violation is not uniform during the year. Looking at figure 4.28 (a) and 4.29 (a) it is also clear that there is an absolute elevated frequency for domain violation where at least one of the vessels are a fishing vessel in March and April. This period coincide with the traditional cod fishing("Lofotfisket") in Lofoten. From early February until late April the normal regulations for fisheries are replaced and a special inspectorate are instated to monitor the fishing activities both on land and by sea. Normal regulations are reinstated once the temporary inspectorate are dissolved (Bjørge and Hallenstvedt, 2018).

Figure 4.30 (a) shows the seasonal distribution for domain violations where neither vessel are a fishing vessel and there is a significant increase during the summer months. The largest contribution to domain violations between two non fishing vessels comes from the ferry service between Bodø and Moskenes. This is not an indication of poor navigation, but is rather an indication that the ship domain may be overestimated for good sea and weather conditions. The distinct time of day distribution are presumed to follow the route table for this ferry service as a consequence, and does not reflect the higher and lower risk during the day.

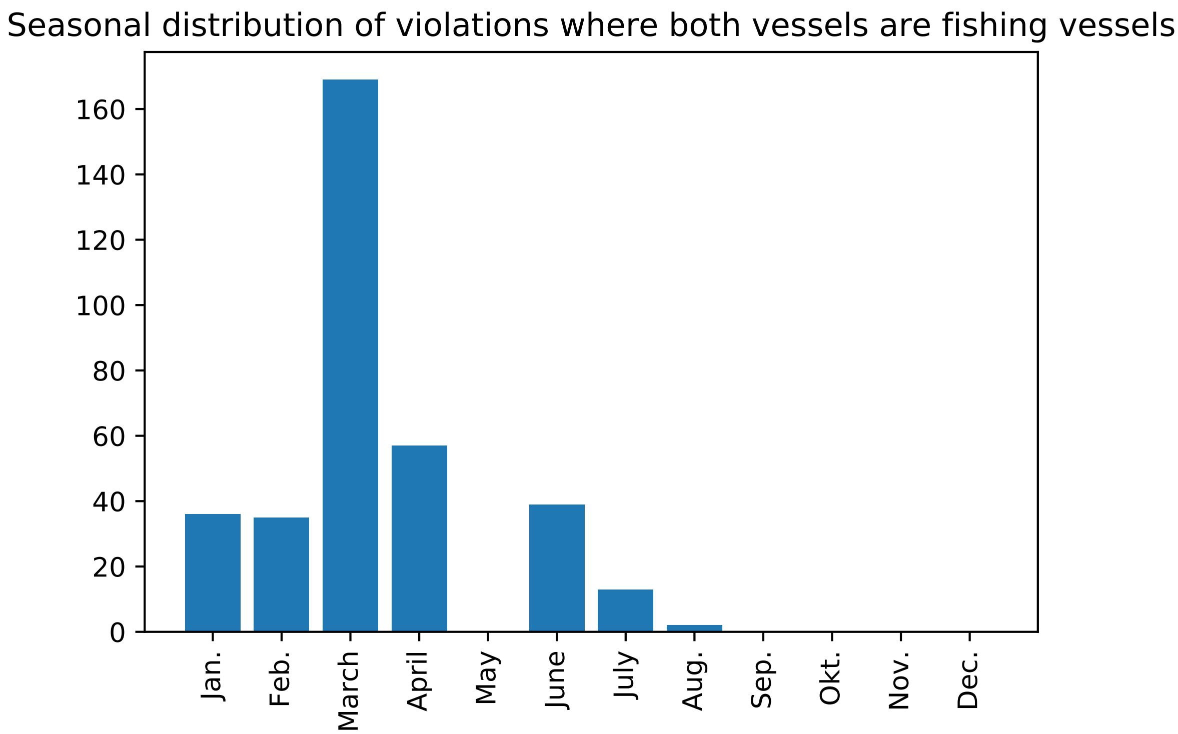


(a) Seasonal distribution

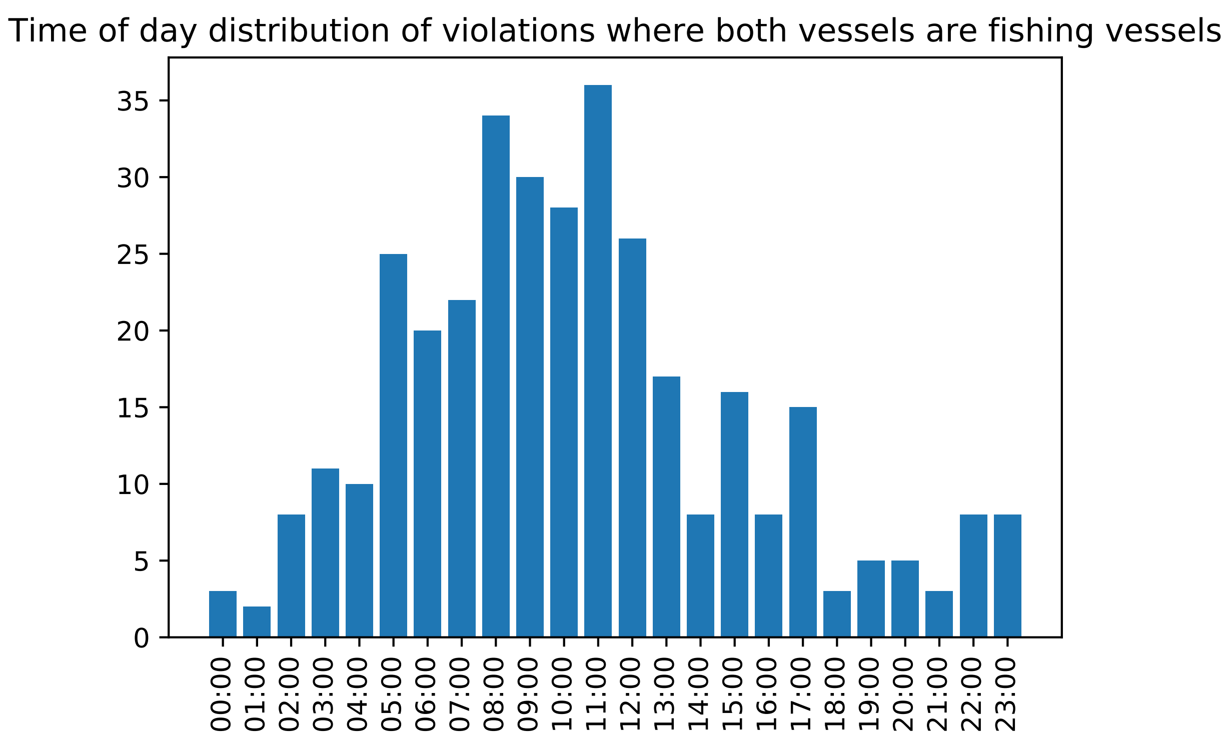


(b) Time of day distribution

Figure 4.27: The figure shows the seasonal and time of day distribution for ship domain violations

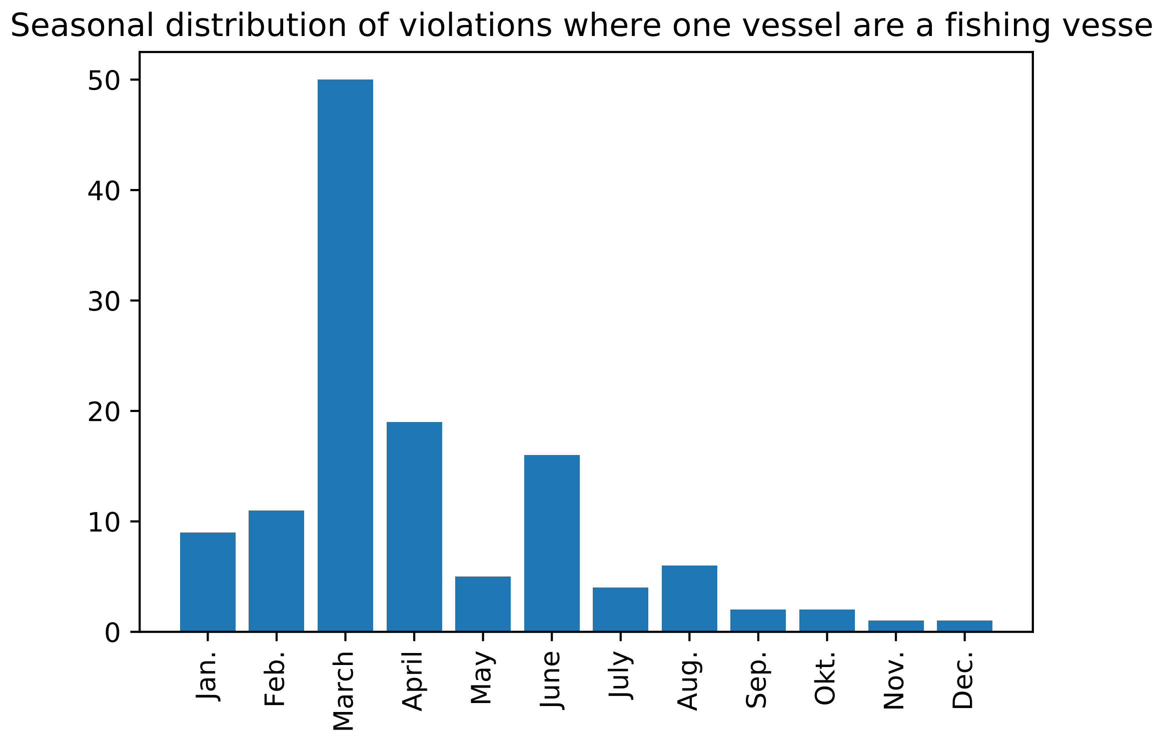


(a) Seasonal distribution

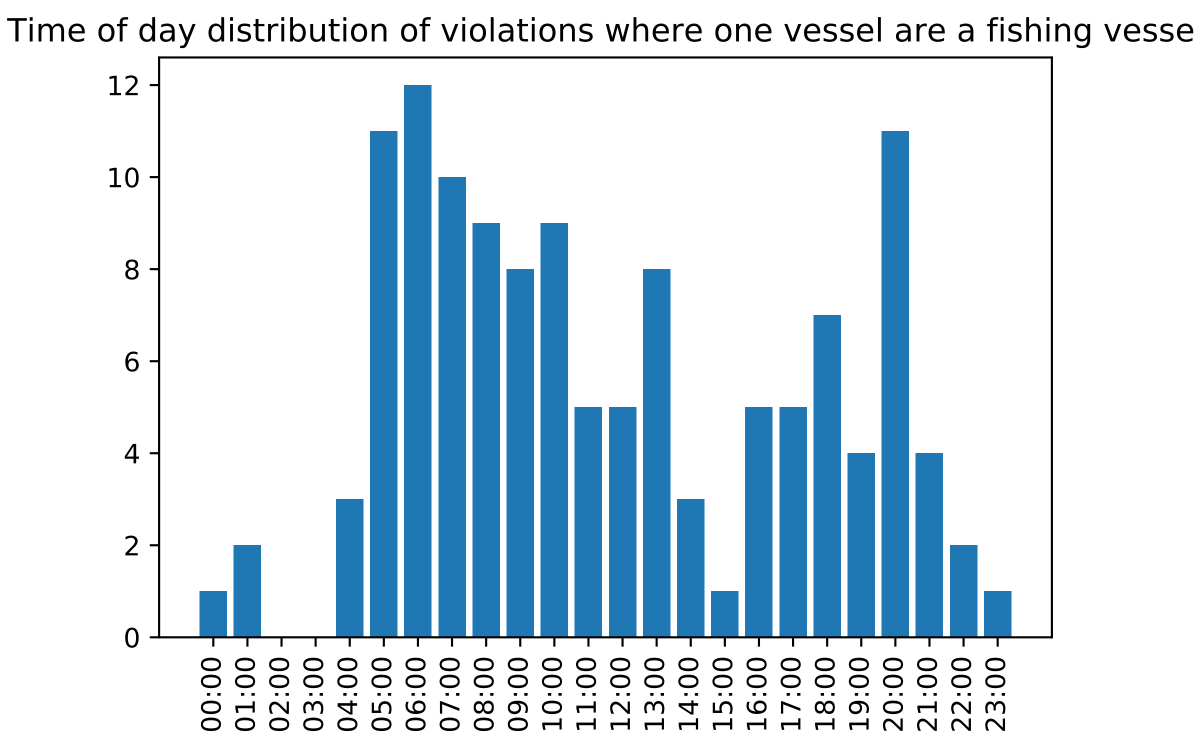


(b) Time of day distribution

Figure 4.28: The figure shows the seasonal and time of day distribution for ship domain violations where both vessels are fishing vessels

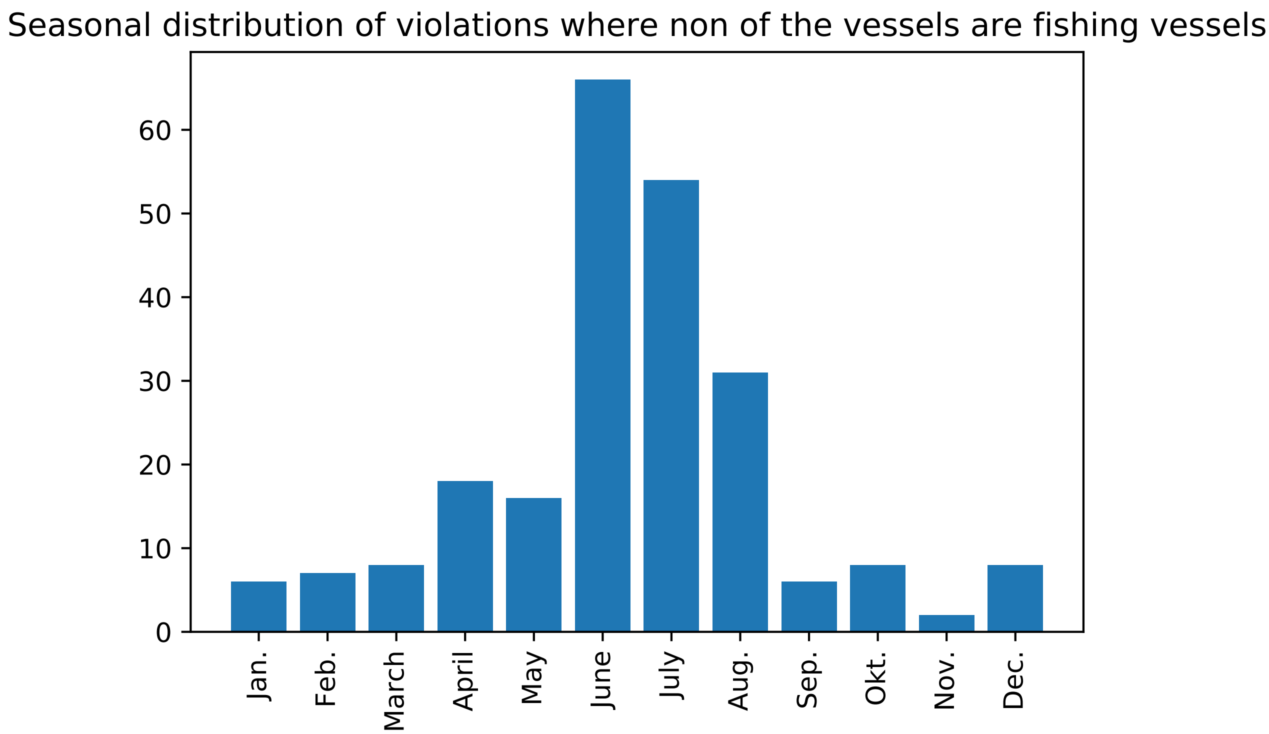


(a) Seasonal distribution

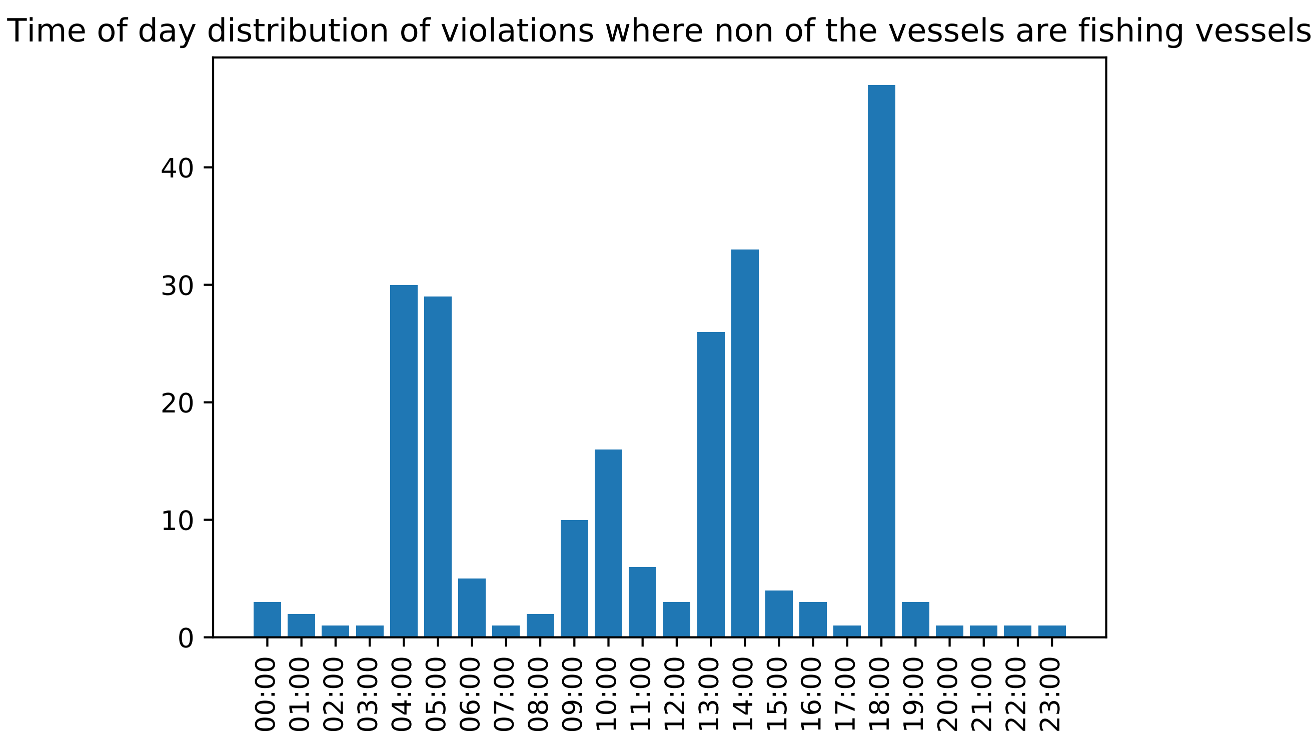


(b) Time of day distribution

Figure 4.29: The figure shows the seasonal and time of day distribution for ship domain violations where one of the vessels are a fishing vessel



(a) Seasonal distribution



(b) Time of day distribution

Figure 4.30: The figure shows the seasonal and time of day distribution for ship domain violations where neither vessel are a fishing vessel

4.4.2 Frequency of domain violations between fishing vessels

The list of encounters contain 351 domain violations between fishing vessels and are caused by 227 unique MMSI numbers, this means that some vessels are involved in several violations. Figure 4.31 shows the frequency for each unique MMSI number with at least on violation normalized for distance.

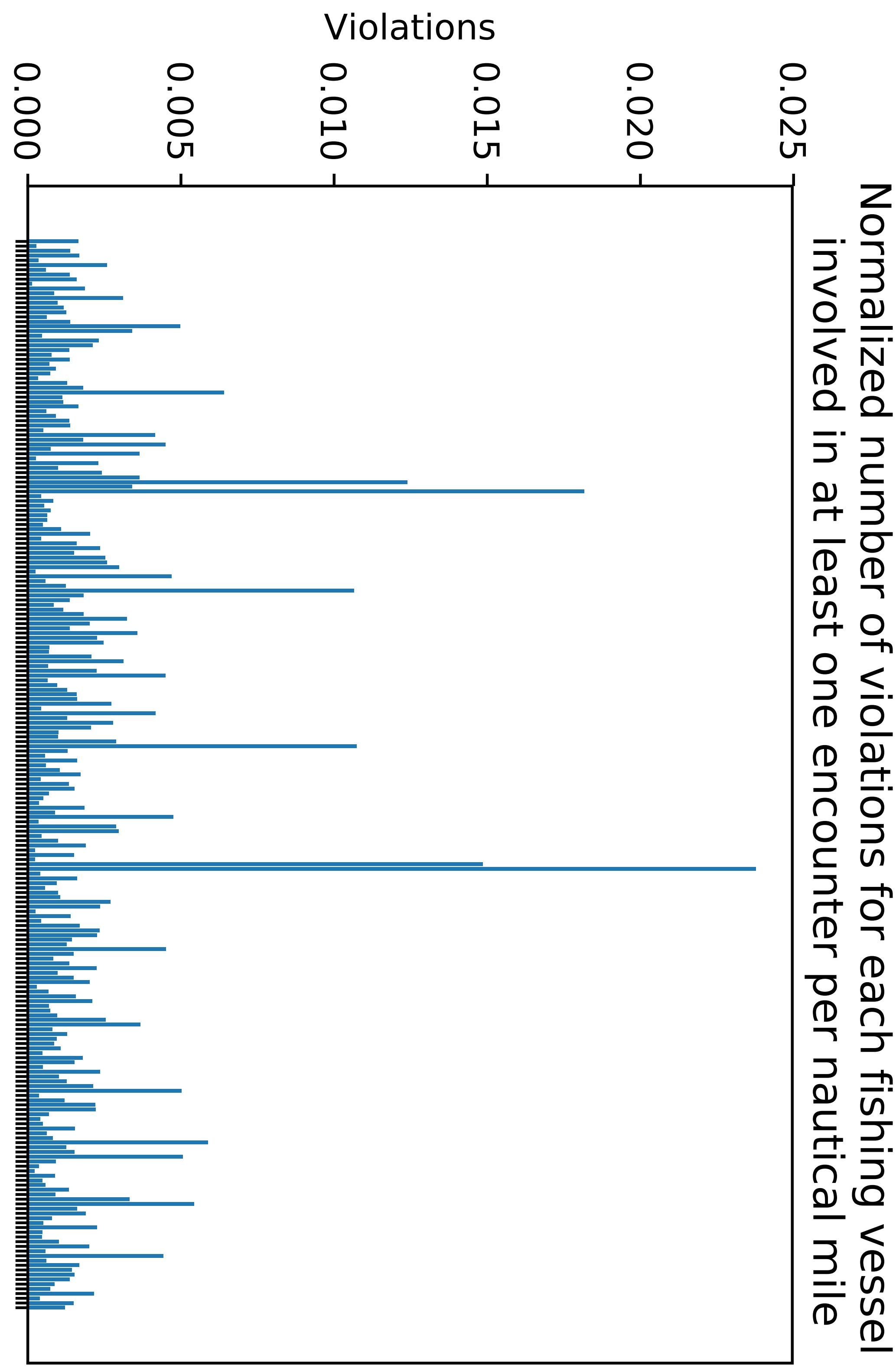


Figure 4.31: The figure shows how some fishing vessels have a significantly higher frequency of domain violations than others

5. Validity and Discussion

5.1 AIS data reliability

As it has been observed and recognized by (Iphar et al., 2015; Harati-Mokhtari et al., 2007) errors are not uncommon in AIS messages, more so for static than dynamic data. There model does not include pre- processing and cleaning of dynamic data with the exception of re-indexing, this leads to some false encounters as a result of errors in the dynamic data, see example scenario 4.3.5. For the current version of the model, these types of erroneous data are seemingly increasing the number violations, meaning that few if any exceptional encounters are undetected. With all models there will be need for human interaction and these types of errors would be relatively easy to dismiss by the analyst. The erratic dynamic data will also lead to an increase in overall traveled distance affecting the normalized frequencies, assuming that these errors are randomly distributed they should not affect the overall result.

5.2 Uncertainty

The algorithm calculating distance for each vessel may be overly aggressive when it comes to discarding data points for fishing vessels or other vessels having reasons to slow down for prolonged periods out at sea. The total distance will only have smaller errors, but it can lead to over estimating the number of total trips per vessel. Having artificially high number of trips can lead to underestimation of normalized risk for certain groups.

In the sorting algorithm of step 3, estimated ROT is utilized to sort out unexceptional encounters, where as Mestl et al. (2016) used recorded ROT from the AIS data. ROT is dynamic information that may be available in AIS data, if it is configured to transmit this data. From a training sett outside Bergen in 2016, 489 out of 2276 unique MMSI numbers had values of ROT above zero indicating that these vessels had activated ROT transmissions. Further more only one of 197 fishing vessel had recorded ROT. For Vestfjorden with a larger percentage of fishing vessel the share of vessels transmitting ROT are likely lower, so it is necessary to create algorithm estimating this value. From the accident scenario in section 4.3.1 the estimated ROT had a lower value than the more accurate reported ROT, this is however not a consistent trend. Among the sorted encounters the highest estimated ROT is 7827.0 deg/min, that translate to almost a full revolution in 3 seconds which is obviously erroneous. These high value mostly occur for smaller vessels during low speed periods which are more likely to have higher ROT.

It has been discovered many pairwise movements between fishing vessel that appear to be random. Some om these encounters could be normal operating procedure, and thus explain why a few vessels have a very high frequency of being involved in domain violations (see figure 4.31). This may be interesting for further work

The crude solution to solve unlikely high speeds in section 4.1.1 is a source of error and could been solved in a smarter way. The effect of this simplification is small if not negligible, the share of messages affected is small and many of these are erroneous. This is backed up by looking at domain violations, the maximum speed in any encounter is 21.1 kts when rescue vessels are omitted. The rescue vessels have recorded SOG values to just under 26, that being said during the 4 year period the rescue vessels have spent 5.28 hours above 26 kts and 24 min above 27 considering a 3 sec reporting interval. There might have been situations where the reduced speed and thereby ship domain of these rescue vessels have lead to violations not being detected.

The ship domain search in step two uses conformed data where the data points often will be offset by some seconds from the "nearest" method of fitting the data. Considering a typical speed of 10 kts, a 4 second offset equates to an positional error of approximately 20 meters. This error is considered to have negligible effect on the results.

As it is mentioned in section 4.1.2 true center is not calculated when either dimension to bow or aft is missing and the length is set to an fixed value. This will lead to some uncertainty in the accuracy of the domain violation search and it is unknown if this is a conservative solution to the problem.

The solution for handling missing course over ground is assumed to be too conservative, from the over 44 million AIS messages in the data set none of the detected encounters are categorized as unknown indicating an encounter with missing COG at the closest point of approach.

5.3 Structure

The ship domain concept is widely accepted for its application in maritime risk assessment, and the research in ship domains are increasing Baran et al. (2018). Step two in the proposed model which finds ship domain violations have a strong position in the maritime navigational risk field. The research on ship domain are increasing Baran et al. (2018) and are likely to remain an active research area.

The third step of analyzing rate of turn is a less researched topic, but is a promising method for detecting exceptional encounters. The application of risk assessment models will always require human input and evaluation. This would usually involve an expert panel validating that the results correlate with the perceived risk in an area. This third step would lessen the work load for expert panel.

5.4 Predictive

The model's ability to predict frequency of collisions is difficult to determine. In the current work it has only been applied to one area with a single collision, it will be necessary to test it on other areas with more collisions to validate its ability. There are existing models for estimation of annual collision frequency(see section 2.2) and the model can be compared to those to get an indication of the number of exceptional encounters per collision. The number of exceptional encounters can be used as an indication of risk, but as Zhang et al. (2016) noted, the number of near misses are not necessary correlated to number of accidents. The model proposed can be seen on as a diagnostic tool for other applications.

5.5 Discussion

The model has shown that is capable of detecting exceptional encounters which a collision at sea is. The area selected for the case study is not ideal in terms of few registered collisions, but the model did detect the one collision that is known to have occurred.

The model in its current version have some limitations that should be resolved before it can be used in an more professional setting which is described previously in this chapter. The model are probably overestimating encounters between fishing vessels as result of the chosen domain and solution to missing length. Considering that most of the collisions in the area are between fishing vessels(see figure 4.25), it is reasonable that most violations involve at least one fishing vessel. As mentioned in section 4.4, there is a poor correlation between intensity of domain violations and registered collisions at NMA outside Røst. It can be argued that it is limitations in the model and limited contextual data, but due to the amount of violations it could potentially be an interesting area for documenting under reporting.

One of advantages of the proposed model is that it includes the navigators knowledge that other ships are in the area by not simulating the traffic flow, when other ships are present the navigator will likely be more attentive and keep their distance.

6. Conclusion and Recommendations for further work

The method developed in this thesis achieved its purpose of detecting exceptional encounters. The model in its current state have some limitations in dealing with fishing vessels and with noise in ROT estimation. The frequency of encounters between fishing vessels is too large to be explain as noise. More contextual data will be needed to more accurately interpret these encounters. The model's ability of interpreting encounters of vessels en route is better, the hybrid of traditional ship domain methodology and evasive maneuver detection is promising. Estimating rate of turn from course over ground and time deltas is prone to a certain degree of noise. This is evident from the many registered head-on encounters on the ferry service Bodø - Værøy - Røst – Moskenes. Utilizing rate of turn when available will reduce the noise and improve accuracy.

The term exceptional encounter is chosen carefully to not overstating the capabilities of the current model. There are research in the field of near miss detection from AIS data, but it has been difficult to show a strong correlation between near misses and collision frequency. The combination of ship domain and evasive maneuver detection applied in the proposed model could be better at approximating actual collision frequency, but the flaws in the current version would have to be resolved. Further it will require a significantly larger dataset with more documented collisions to verify the model quality.

6.1 Recommendations for further work

The first modification to the model should be to enable recorded ROT to be utilized when available in step 3. The second modification should be to review the ROT estimation algorithm and make improvements. Including coordinates, longer moving averages and message skipping during periods with short transmission interval are some the approaches that could mitigate the extreme and strange ROT values that occur es with the current setup.

Look for more contextual data to implement in step 3 for sorting out exceptional encounters. Factors such as other traffic, waves, sight, current and vessel type can improve the model's ability to classify an encounter as exceptional.

A possible use case for this model is to measure the effectiveness of new sea marks, especially for new markings of rocks and obstacles in fairways. A slightly altered model could output detailed information of which ship types or other ship variables that are passing with the least clearance, highest speed etc. according to what type of information that would be useful in the context. That being said, much of the same information would be accessible by traffic flow analysis.

The proposed model uses crisp ship domains, for further work one could look into fuzzy domains that are well documented in the literature. For implementation it is advised to run step two with the largest domain size and account for fuzzy domain in post processing. This will reduce total amount of computation. It could be interesting to compare fuzzy domain with step three of the model. A fuzzy domain addition in the model could also mitigate the challenges with having a critical situation without evasive maneuvers.

The the present state of the model can only give an indication of frequency of exceptional encounters. A common definition of risk is probability/frequency multiplied by consequences. An addition of consequence estimation could make the model to a more complete risk assessment tool. Since the model finds encounters from historic data it can find the "true" frequency of encounters between certain types of vessels. It could hypothetical know that tankers usually arrive during morning and cruise ships in the evening, this would impact both collision frequency and consequence estimation.

The code is made to be scalable with processing power, but there are plenty of code optimization for speed improvements. The domain violation search are currently a type brute-force algorithm measuring all concurrent data points. For a given timestep with n vessels the program will check all possible pairwise combinations which equates to:

$$\sum_{i=2}^{n_{ships}} (i-1)$$

A method for reducing computation is to reduce the search area, an algorithm that can efficiently section the search space into smaller pieces could reduce computation dramatically.

In the next iteration of the model more pre-processing of data should be applied to remove erratic AIS messages as is shown in section 4.3.5.

7. Acronyms

AIS Automatic Information System

COG Course over ground

COLREGS Convention on the International Regulations for Preventing Collisions at Sea

CPA Closest Point of Approach

DCPA Distance at Closest Point of Approach

ETA Estimated time of arravial

IALA International Association of Lighthouse Authorities

IMO International Maritime Organization

ITU International Telecommunication Union

MMSI Maritime Mobile Service Identity

ROT Rate of Turn

SOG Speed over ground

SOLAS International Convention for the Safety of Life at Sea

TCPA Time to Closest Point of Approach

VHF Very High Frequency

VCRO Vessel Conflict Ranking Operator

VTS Vessel Traffic Service

WGS84 World Geodetic System, same as EPSG4326

Bibliography

Baran, A., Fiskin, R., and Kiźi, H. (2018). A research on concept of ship safety domain. *TransNav: International Journal on Marine Navigation and Safety of Sea Transportation*, 12(1):43–47.

Bjørge, C. and Hallenstvedt, A. (2018). lofotfisket – store norske leksikon https://snl.no/lofotfisket. *Store norske leksikon*. Accessed: 2018-06-17.

Chen, P., Mou, J., and Gelder, P. (2017). Risk assessment methods for ship collision in estuarine waters using ais and historical accident data.

Chen, P., Mou, J., and van Gelder, P. (2018). Risk assessment methods for ship collision in estuarine waters using ais and historical accident data.

EU (2016). Control technologies - fisheries - european commission, https://ec.europa.eu/fisheries/cfp/control/technologies_en. Accessed: 2018-05-18.

Friis-Hansen, P. and Simonsen, B. C. (2002). Gracat: software for grounding and collision risk analysis. *Marine Structures*, 15(4):383 – 401. Ship Collisions and Grounding.

Fujii, Y. (1974). Some factors affecting the frequency of accidents in marine traffic [j]. *Journal of Navigation*, 27:235–252.

Fujii, Y., Yamanouchi, H., and Mizuki, N. (1970). On the fundamentals of marine traffic control. part 1 probabilities of collision and evasive actions. *Electronic Navigation Research Institute Papers*, 2:1–16.

GIS Geography (2018). World geodetic system (wgs84) https://gisgeography.com/wgs84-world-geodetic-system/. Accessed: 2018-05-18.

Goerlandt, F., Montewka, J., Lammi, H., and Kujala, P. (2012). Analysis of near collisions in the gulf of finland. pages 2880–2886.

Goodwin, E. M. (1975). A statistical study of ship domains. *The Journal of navigation*, 28(3):328–344.

Harati-Mokhtari, A., Wall, A., Brooks, P., and Wang, J. (2007). Automatic identification system (ais): data reliability and human error implications. *The Journal of Navigation*, 60(3):373–389.

IALA 2016. Guideline no. 1082 an overview of ais. http://www.iala-aism.org/product/an-overview-of-ais-1082/.

IALA 2017. G1123 – the use of iala waterway risk assessment programme (iwrap mkii) http://www.iala-aism.org/product/g1123-use-iala-waterway-risk-assessment-programme-iwrap-mkii/.

IMO (1972). Convention on the international regulations for preventing collisions at sea, 1972 (colregs) http://www.imo.org/en/About/Conventions/ListOfConventions/Pages/COLREG.aspx.

IMO (2002a). Guidelines for the onboard operational use of shipborne automatic identification systems (ais) resolution a.917(22) . https://www.navcen.uscg.gov/pdf/AIS/IMO_A_917(22)_AIS_OPS_Guidelines.pdf. Accessed: 2018-05-19.

IMO (2002b). Resolution msc.137(76) standards for ship manoeuvrability http://www.imo.org/en/KnowledgeCentre/IndexofIMOResolutions/Maritime-Safety-Committee-(MSC)/Documents/MSC.137(76).pdf. Accessed: 2018-05-19.

IMO (2014). *SOLAS, Consolidated Edition 2014: Consolidated Text of the International Convention for the Safety of Life at Sea, 1974, and Its Protocol of 1988 : Articles, Annexes and Certificates*. IMO publication. International Maritime Organization.

Iperen, W. (2015). 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.

Iphar, C., Napoli, A., and Ray, C. (2015). Data quality assessment for maritime situation awareness. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, 2.

ITU (2014). Technical characteristics for an automatic identification system using time division multiple access in the vhf maritime mobile frequency ban https://www.itu.int/dms_pubrec/itu-r/rec/m/R-REC-M.1371-5-201402-I!!PDF-E.pdf.

Liu, S., Zheng, K., Wu, Z., and Wang, N. (2015). New inland river channel transit capacity evaluation method based on dynamic quaternion ship domain model. In *Society of Instrument and Control Engineers of Japan (SICE), 2015 54th Annual Conference of the*, pages 652–656. IEEE.

Macduff, T. (1974). The probability of vessel collisions. *Ocean Industry*, 9(9).

Mestl, T., Tallakstad, K. T., and Castberg, R. (2016). Identifying and analyzing safety critical maneuvers from high resolution ais data. *TransNav: International Journal on Marine Navigation and Safety of Sea Transportation*, 10.

Montewka, J., Hinz, T., Kujala, P., and Matusiak, J. (2010). Probability modelling of vessel collisions. *Reliability Engineering System Safety*, 95(5):573–589.

NASA (2015). Space station keeps watch on world's sea traffic https://www.nasa.gov/mission_pages/station/research/benefits/sea_traffic.html. Accessed: 2018-05-18.

Pedersen, P. T. et al. (1995). Collision and grounding mechanics. *Proceedings of WEMT*, 95(1995):125–157.

Qu, X., Meng, Q., and Suyi, L. (2011). Ship collision risk assessment for the singapore strait. *Accident Analysis Prevention*, 43(6):2030 – 2036.

Shelmerdine, R. L. (2015). Teasing out the detail: How our understanding of marine ais data can better inform industries, developments, and planning. *Marine Policy*, 54:17–25.

Smestad, B. B. (2015). A Study of Satellite AIS Data and the Global Ship Traffic Through the Singapore Strait. Master's thesis, Norwegian University of Science and Technology, Trondheim.

Szlapczynski, R. and Szlapczynska, J. (2017). Review of ship safety domains: Models and applications. *Ocean Engineering*, 145:277–289.

Vinnem, J. E. (2013). Offshore risk assessment vol 1 - principles modelling and applications of qra studies. pages 323–343.

Wang, N. (2010). An intelligent spatial collision risk based on the quaternion ship domain. *The Journal of Navigation*, 63(4):733–749.

Wang, N., Meng, X., Xu, Q., and Wang, Z. (2009). A unified analytical framework for ship domains. *The Journal of Navigation*, 62(4):643–655.

Zhang, W., Goerlandt, F., Kujala, P., and Wang, Y. (2016). An advanced method for detecting possible near miss ship collisions from ais data. *Ocean Engineering*, 124:141–156.

Zhang, W. B., Goerlandt, F., Montewka, J., and Kujala, P. (2015). A method for detecting possible near miss ship collisions from ais data. *Ocean Engineering*, 107:60–69.

Zhang, W. B., Kopca, C., Tang, J. J., Ma, D. F., and Wang, Y. H. (2017). A systematic approach for collision risk analysis based on ais data. *Journal of Navigation*, 70(5):1117–1132.

Zhou, K., Chen, J., and Liu, X. (2018). Optimal collision-avoidance manoeuvres to minimise bunker consumption under the two-ship crossing situation. *The Journal of Navigation*, 71(1):151–168.

A. Code

A.1 Conform data

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Mon Apr 23 10:11:50 2018

@author: haakon
"""
#python3.6 /home/haakon/Documents/par_reindex.py

from joblib import Parallel, delayed
import numpy as np
import pandas as pd
from datetime import datetime
from datetime import timedelta
import sqlite3 as lite
import time
import multiprocessing

def reindex10(mmsi):
    t=time.time()
    multiprocessing.Lock().acquire()
    conn = lite.connect('/home/haakon/Documents/Vestfjorden2014.db')
    c = conn.cursor()
    c.execute(f"select Time,MMSI,SOG,Longitude,Latitude,COG,IMO,Vessel_name,Bow,Aft,Port
    entries = pd.DataFrame(c.fetchall(), columns=header)
    entries['Time']= entries['Time'].astype('datetime64[s]')
    if len(entries.index) >1:
        entries=entries.drop_duplicates(subset='Time',keep='first')
        entries.set_index('Time', inplace=True)
        entries = entries.dropna(axis=0, subset=['SOG', 'Longitude', 'Latitude', 'COG'],
#        entries = entries.drop_duplicates(keep='first')
        entries = entries.reindex(date_index, method='nearest', tolerance=timedelta(seco
        entries[['MMSI','COG','IMO','Vessel_name','Bow','Aft','Port','Starb']]=entries[[
        entries[['SOG','Longitude','Latitude']] = entries[['SOG','Longitude','Latitude']
        entries = entries.dropna(subset=['MMSI'])
        entries = entries.reset_index()
        entries.rename(columns={'index': 'Time'}, inplace=True)
        entries['MMSI']= entries['MMSI'].astype('int64')
        try:
            entries['IMO']= entries['IMO'].astype('int64')
        except:
            pass
    else:
        pass
    multiprocessing.Lock().acquire()
    entries.to_csv('reindexed_vest_2014.csv', mode='a', header=False, index=False)

header = ('Time','MMSI','SOG','Longitude','Latitude','COG','IMO','Vessel_name','Bow','Af
#c.execute("CREATE TABLE reindexed_2013 (Time DATETIME, MMSI    INT,SOG  float, Longitu
conn = lite.connect('/home/haakon/Documents/Vestfjorden2014.db')
c = conn.cursor()

t_1=time.time()
c.execute("select distinct MMSI from raw_2014")
unique_mmsi = [int(i[0]) for i in c.fetchall()]
date_index = pd.date_range(start='01/01/2014', end='01/01/2015', freq='15S')
par = Parallel(n_jobs=16, verbose=10)
do_something = delayed(reindex10)
par(do_something(mmsi) for mmsi in unique_mmsi)
print(time.time()-t_1)
```

```
c.execute("CREATE TABLE All_violations (dom_id INTEGER, time_CPA DATETIME,distance_c
                                        cog_2_C Float,IMO_1 INTEGER,IMO_2 INTEGER,Flag_1

c.execute("select Count(*) from All_violations where Flag_1 = 'Norway' AND Flag_2 = 'Nor
c.fetchall()
```

A.2 Ship domain violation search

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Apr 25 18:06:34 2018
@author: haakon
"""
##### READ THIS###############
# change startyear in the bottom to the year the dataset starts
# change endyear if the datasett is over several years
# Change the csv save filename for each dataset, currently dom_vio16
# you can use a similar comand to run the script from the terminal in linux
#       python3.6 /home/haakon/Documents/par_ship_domain.py


from joblib import Parallel, delayed
import numpy as np
from datetime import datetime
from datetime import timedelta
from dateutil.relativedelta import relativedelta
import sqlite3 as lite
import time
import multiprocessing
import csv
from geographiclib.geodesic import Geodesic

#angle from heading of own ship to position of other ship
def alpha(lon_0, lat_0, lon_2, lat_2, cog_0):
    azi1=Geodesic.WGS84.Inverse(lat_0,lon_0, lat_2,lon_2)['azi1']
    if type(cog_0) == str:
        if azi1<0:
            return np.radians(360+azi1)
        else:
            return np.radians(azi1)
    else:
        if azi1<0:
            azi1=360+azi1
        if azi1<=cog_0:
            return np.radians(360-(cog_0-azi1))
        else:
            return np.radians(azi1-cog_0)
#length from own ship center to ship domain boundary in direction of target ship center
def l_a(length_0,lat_0, lon_0, lat_2, lon_2, cog_0,v_0):
    k_AD=10**(0.3591*np.log10(v_0)+0.0952)
    k_DT=10**(0.5441*np.log10(v_0)-0.0795)
    #course unknown cicular domain
    if type(cog_0) == str:
        R_fore=R_aft=R_starb=R_port=(0.2+k_DT)*length_0
    else:
        R_fore=(1+1.34*np.sqrt(k_AD**2+(k_DT/2)**2))*length_0
        R_aft=(1+0.67*np.sqrt(k_AD**2+(k_DT/2)**2))*length_0
        R_starb=(0.2+k_DT)*length_0
        R_port=(0.2+0.75*k_DT)*length_0
    #Coordinates to true center
    if alpha(lon_0, lat_0, lon_2, lat_2, cog_0)<=np.pi/2:
        l =(((1+np.tan(alpha(lon_0, lat_0, lon_2, lat_2, cog_0))**2)/((1/R_fore**2)\
             +(np.tan(alpha(lon_0, lat_0, lon_2, lat_2, cog_0))**2)/R_starb**2))**0.5)

    elif np.pi/2 < alpha(lon_0, lat_0, lon_2, lat_2, cog_0) and \
    alpha(lon_0, lat_0, lon_2, lat_2, cog_0)<=np.pi:
        l =(((1+np.tan(alpha(lon_0, lat_0, lon_2, lat_2, cog_0))**2)/((1/R_aft**2)+\
```

```
                (np.tan(alpha(lon_0, lat_0, lon_2, lat_2, cog_0))**2)/R_starb**2))**0.5)

    elif np.pi < alpha(lon_0, lat_0, lon_2, lat_2, cog_0) and \
    alpha(lon_0, lat_0, lon_2, lat_2, cog_0)<=(3/2)*np.pi:
        l =(((1+np.tan(alpha(lon_0, lat_0, lon_2, lat_2, cog_0))**2)/((1/R_aft**2)+\
              (np.tan(alpha(lon_0, lat_0, lon_2, lat_2, cog_0))**2)/R_port**2))**0.5)

    elif (3/2)*np.pi < alpha(lon_0, lat_0, lon_2, lat_2, cog_0):
        l =(((1+np.tan(alpha(lon_0, lat_0, lon_2, lat_2, cog_0))**2)/((1/R_fore**2)+\
              (np.tan(alpha(lon_0, lat_0, lon_2, lat_2, cog_0))**2)/R_port**2))**0.5)
    return l
#dir_x = 1 towards bow and 2 towards aft, dir_y = 1 towards starboard and 2 towards port
def True_center(lat_0,lon_0,d_bow,d_aft,d_starb,d_port,cog_0):
    if type(cog_0) == str:
        return(lat_0,lon_0)
    elif type(d_bow) == str:
        return(lat_0,lon_0)
    elif type(d_aft) == str:
        return(lat_0,lon_0)
    elif type(d_starb) == str:
        d_starb=d_port=0
    elif type(d_port) == str:
        d_port=d_starb=0
    else:
        ##hack for 2013/2014 missing d_starb
#        d_starb=d_port
        try:
            if d_bow>d_aft:
                dist_x=(d_bow+d_aft)/2-d_aft
                dir_x=1
                if d_starb>d_port:
                    dist_y=(d_starb+d_port)/2-d_port
                    dir_y=1
                    s_12=np.sqrt(dist_x**2+dist_y**2)
                elif d_starb<d_port:
                    dist_y=(d_starb+d_port)/2-d_port
                    dir_y=2
                    s_12=np.sqrt(dist_x**2+dist_y**2)
                elif d_starb==d_port:
                    dir_y=0
                    dist_y=0
                    s_12=np.sqrt(dist_x**2+dist_y**2)
            elif d_bow<d_aft:
                dist_x=(d_bow+d_aft)/2-d_bow
                dir_x=2
                if d_starb>d_port:
                    dist_y=(d_starb+d_port)/2-d_port
                    dir_y=1
                    s_12=np.sqrt(dist_x**2+dist_y**2)
                elif d_starb<d_port:
                    dist_y=(d_starb+d_port)/2-d_port
                    dir_y=2
                    s_12=np.sqrt(dist_x**2+dist_y**2)
                elif d_starb==d_port:
                    dir_y=0
                    dist_y=0
                    s_12=np.sqrt(dist_x**2+dist_y**2)
            elif d_bow==d_aft:
                dir_x=0
                if d_starb>d_port:
```

```
                    dist_y=(d_starb+d_port)/2-d_port
                    dir_y=1
                    s_12=np.sqrt(dist_x**2+dist_y**2)
                elif d_starb<d_port:
                    dist_y=(d_starb+d_port)/2-d_port
                    dir_y=2
                    s_12=np.sqrt(dist_x**2+dist_y**2)
                elif d_starb==d_port:
                    dir_y=0
                    dist_y=0
                    s_12=np.sqrt(dist_x**2+dist_y**2)
            if dir_x==0 and dir_y==0:
                t_lon_0=lon_0
                t_lat_0=lat_0
            elif dir_x==1 and dir_y==0:
                vector=Geodesic.WGS84.Direct(lat_0,lon_0,cog_0,s_12,outmask=1929)
                t_lon_0=vector['lon2']
                t_lat_0=vector['lat2']
            elif dir_x==2 and dir_y==0:
                vector=Geodesic.WGS84.Direct(lat_0,lon_0,(cog_0+180),s_12,outmask=1929)
                t_lon_0=vector['lon2']
                t_lat_0=vector['lat2']
            elif dir_x==0 and dir_y==1:
                vector=Geodesic.WGS84.Direct(lat_0,lon_0,(cog_0+90),s_12,outmask=1929)
                t_lon_0=vector['lon2']
                t_lat_0=vector['lat2']
            elif dir_x==0 and dir_y==2:
                vector=Geodesic.WGS84.Direct(lat_0,lon_0,(cog_0-90),s_12,outmask=1929)
                t_lon_0=vector['lon2']
                t_lat_0=vector['lat2']
            elif dir_x==1 and dir_y==1:
                vector=Geodesic.WGS84.Direct(lat_0,lon_0,(cog_0+np.degrees(np.arctan(dis
                t_lon_0=vector['lon2']
                t_lat_0=vector['lat2']
            elif dir_x==1 and dir_y==2:
                vector=Geodesic.WGS84.Direct(lat_0,lon_0,(cog_0+(360-np.degrees(np.arcta
                t_lon_0=vector['lon2']
                t_lat_0=vector['lat2']
            elif dir_x==2 and dir_y==2:
                vector=Geodesic.WGS84.Direct(lat_0,lon_0,(cog_0+(180+np.degrees(np.arcta
                t_lon_0=vector['lon2']
                t_lat_0=vector['lat2']
            elif dir_x==2 and dir_y==1:
                vector=Geodesic.WGS84.Direct(lat_0,lon_0,(cog_0+(180-np.degrees(np.arcta
                t_lon_0=vector['lon2']
                t_lat_0=vector['lat2']
            return (t_lat_0,t_lon_0)
        #if dimensions are missing
        except:
            return(lat_0,lon_0)
def violations(i):
    print(i)
    if i.date().day == 1:
        d=i+timedelta(days=3)
    elif i.date().day == 4:
        d=i+timedelta(days=3)
    elif i.date().day == 7:
        d=i+timedelta(days=3)
    elif i.date().day == 10:
        d=i+timedelta(days=3)
```

```
elif i.date().day == 13:
    d=i+timedelta(days=3)
elif i.date().day == 16:
    d=i+timedelta(days=3)
elif i.date().day == 19:
    d=i+timedelta(days=3)
elif i.date().day == 22:
    d=i+timedelta(days=3)
elif i.date().day == 25:
    d=i+timedelta(days=3)
elif i.date().day == 28:
    #first of next month
    d = i + relativedelta(months=+1, day=1)
multiprocessing.Lock().acquire()
conn = lite.connect('/home/haakon/Documents/Vestfjorden2016.db')
c = conn.cursor()
#select all entries in one month starting with date i
c.execute(f"select * from reindexed_2016 WHERE Time >= '{i}' AND Time < '{d}' order
entries=c.fetchall()
#joblib is releasing the lock, otherwise multiprocessing.Lock().release()
for entry_0 in range(len(entries)):
    timestamp_0 = datetime.strptime(entries[entry_0][0], "%Y-%m-%d %H:%M:%S")
    mmsi_0 = entries[entry_0][1]
    lon_0 = entries[entry_0][3]
    lat_0 = entries[entry_0][4]
    try:
        imo_0 = entries[entry_0][6]
    except:
        imo_0 = 9
    #hack for nonetype
    try:
        try:
            v_0 = entries[entry_0][2]
            if v_0>26:
                v_0=10
        except:
            v_0=5.001
        cog_0 = entries[entry_0][5]
        try:
            d_bow_0 = entries[entry_0][8]
            d_aft_0 = entries[entry_0][9]
            length_0=d_bow_0+d_aft_0
        except:
            length_0=40.001
        d_port_0 = entries[entry_0][10]
        if type(d_port_0) == str:
            #To be able to recognize vessels where dimensions are missing they are s
            d_port_0=4.501
        d_starb_0 = entries[entry_0][11]
        if type(d_starb_0) == str:
            d_starb_0=4.501
        try:
            width_0=d_port_0+d_starb_0
        except:
            #dimensions is sett to
            width_0 =9.002
    except:
        cog_0=""
        length_0=40.001
        width_0=8.001
```

```
        v_0=5.001
        d_bow_0 = 0
        d_aft_0 = 0
        d_port_0 = 0
        d_starb_0 = 0
    for entry_2 in range((entry_0+1),len(entries)):
        timestamp_2 = datetime.strptime(entries[entry_2][0], "%Y-%m-%d %H:%M:%S")
        #only cheak ais transmissions at the same time instance
        if timestamp_2>timestamp_0:
            break
        #do not cheak for close encounters close to land
        if (13.4  <= lon_0) and (67.0<=lat_0<=67.52):
            break
        if (11.82 <= lon_0 <= 12.25) and (67.39<=lat_0<=67.55):
            break
        if (12.55 <= lon_0 <=12.78 ) and (67.62<=lat_0<=67.77):
            break
        if (12.78 < lon_0 <=12.91 ) and (67.82<=lat_0<=67.88):
            break
        if (12.91 < lon_0 <= 13.01) and (67.86<=lat_0):
            break
        if (13.01 < lon_0 <= 13.06) and (67.876<=lat_0):
            break
        if (13.06 < lon_0 <= 13.09) and (67.876<=lat_0):
            break
        if (13.09 < lon_0 <=13.13 ) and (67.92<=lat_0):
            break
        if (13.13 < lon_0 <=13.156 ) and (67.937<=lat_0):
            break
        if (13.156< lon_0 <= 13.189) and (67.953<=lat_0):
            break
        if (13.189< lon_0 <=13.243) and (67.985<=lat_0):
            break
        if (13.243< lon_0 <=13.293) and (67.996<=lat_0):
            break
        t_lat_0=True_center(lat_0,lon_0,d_bow_0,d_aft_0,d_starb_0,d_port_0,cog_0)[0]
        t_lon_0=True_center(lat_0,lon_0,d_bow_0,d_aft_0,d_starb_0,d_port_0,cog_0)[1]
        mmsi_2 = entries[entry_2][1]
        lon_2 = entries[entry_2][3]
        lat_2 = entries[entry_2][4]
        try:
            imo_2 = entries[entry_2][6]
        except:
            imo_2=9
        try:
            try:
                v_2 = entries[entry_2][2]
                if v_2>26:
                    v_2=10
            except:
                v_2=5.001
            cog_2 = entries[entry_2][5]
            try:
                d_bow_2 = entries[entry_2][8]
                d_aft_2 = entries[entry_2][9]
                length_2 = d_bow_2+d_aft_2
            except:
                length_2 = 40.001
            d_port_2 = entries[entry_2][10]
            if type(d_port_2) == str:
```

```
            d_port_2=4.501
        d_starb_2 = entries[entry_2][11]
        if type(d_port_2) == str:
            d_port_2=4.501
        try:
            width_2=d_port_2+d_starb_2
        except:
            width_2 =9.001
    except:
        cog_2=""
        length_2=40.001
        width_2=8.001
        v_2=5.001
        d_bow_2 = 0
        d_aft_2 = 0
        d_port_2 = 0
        d_starb_2 = 0
    t_lat_2=True_center(lat_2,lon_2,d_bow_2,d_aft_2,d_starb_2,d_port_2,cog_2)[0]
    t_lon_2=True_center(lat_2,lon_2,d_bow_2,d_aft_2,d_starb_2,d_port_2,cog_2)[1]
    #domain length in direction from ship 0 to ship 2 [meter]
    dom_len_0_2=l_a(length_0, t_lat_0, t_lon_0, t_lat_2, t_lon_2, cog_0,v_0)
    #domain length in direction from ship 2 to ship 0 [meter]
    dom_len_2_0=l_a(length_2, t_lat_2, t_lon_2, t_lat_0, t_lon_0, cog_2,v_2)
    #distance center to center [meter]
    distance_center=Geodesic.WGS84.Inverse(t_lat_0,t_lon_0, t_lat_2,t_lon_2)['s1
        #if distance between true ship centers are less than safety domain of sh
    if dom_len_0_2>distance_center and mmsi_0 != mmsi_2:
        if mmsi_0 < mmsi_2:
            if dom_len_2_0<distance_center:
                Violated_domain=mmsi_0
            else:
                Violated_domain=2
            ID = int(str(mmsi_0)+str(mmsi_2))
            domain_overlap_0_2=distance_center-dom_len_0_2
            domain_overlap_2_0=distance_center-dom_len_2_0
            Vessel_name_0 = entries[entry_0][7]
            Vessel_name_2 = entries[entry_2][7]
            fields=[ID,timestamp_0,mmsi_0,mmsi_2,imo_0,imo_2,Violated_domain,dis
            multiprocessing.Lock().acquire()
            with open(r'dom_vio16.csv', 'a') as f:
                writer = csv.writer(f)
                writer.writerow(fields)

        else:
            if dom_len_2_0<distance_center:
                Violated_domain=mmsi_0
            else:
                Violated_domain=2
            ID = int(str(mmsi_2)+str(mmsi_0))
            domain_overlap_0_2=distance_center-dom_len_0_2
            domain_overlap_2_0=distance_center-dom_len_2_0
            Vessel_name_0 = entries[entry_0][7]
            Vessel_name_2 = entries[entry_2][7]
            fields=[ID,timestamp_0,mmsi_2,mmsi_0,imo_2,imo_0,Violated_domain,dis
            multiprocessing.Lock().acquire()
            with open(r'dom_vio16.csv', 'a') as f:
                writer = csv.writer(f)
                writer.writerow(fields)
          #if distance between true ship centers are less than safety domain of
    elif dom_len_2_0>distance_center and mmsi_0 != mmsi_2:
```

```
                if mmsi_0 < mmsi_2:
                    if dom_len_0_2<distance_center:
                        Violated_domain=mmsi_2
                    else:
                        Violated_domain=2
                    ID = int(str(mmsi_0)+str(mmsi_2))
                    domain_overlap_0_2=distance_center-dom_len_0_2
                    domain_overlap_2_0=distance_center-dom_len_2_0
                    Vessel_name_0 = entries[entry_0][7]
                    Vessel_name_2 = entries[entry_2][7]
                    fields=[ID,timestamp_0,mmsi_0,mmsi_2,imo_0,imo_2,Violated_domain,dis
                    multiprocessing.Lock().acquire()
                    with open(r'dom_vio16.csv', 'a') as f:
                        writer = csv.writer(f)
                        writer.writerow(fields)
                else:
                    if dom_len_0_2<distance_center:
                        Violated_domain=mmsi_2
                    else:
                        Violated_domain=2
                    ID = int(str(mmsi_2)+str(mmsi_0))
                    domain_overlap_0_2=distance_center-dom_len_0_2
                    domain_overlap_2_0=distance_center-dom_len_2_0
                    Vessel_name_0 = entries[entry_0][7]
                    Vessel_name_2 = entries[entry_2][7]
                    fields=[ID,timestamp_0,mmsi_2,mmsi_0,imo_2,imo_0,Violated_domain,dis
                    multiprocessing.Lock().acquire()
                    with open(r'dom_vio16.csv', 'a') as f:
                        writer = csv.writer(f)
                        writer.writerow(fields)
startyear = 2016
startmonth = 1
endyear = startyear
endmonth = 12
d_1=[datetime(m//12, m%12+1, 1) for m in range(startyear*12+startmonth-1, endyear*12+end
d_4=[datetime(m//12, m%12+1, 4) for m in range(startyear*12+startmonth-1, endyear*12+end
d_7=[datetime(m//12, m%12+1, 7) for m in range(startyear*12+startmonth-1, endyear*12+end
d_10=[datetime(m//12, m%12+1, 10) for m in range(startyear*12+startmonth-1, endyear*12+e
d_13=[datetime(m//12, m%12+1, 13) for m in range(startyear*12+startmonth-1, endyear*12+e
d_16=[datetime(m//12, m%12+1, 16) for m in range(startyear*12+startmonth-1, endyear*12+e
d_19=[datetime(m//12, m%12+1, 19) for m in range(startyear*12+startmonth-1, endyear*12+e
d_22=[datetime(m//12, m%12+1, 22) for m in range(startyear*12+startmonth-1, endyear*12+e
d_25=[datetime(m//12, m%12+1, 25) for m in range(startyear*12+startmonth-1, endyear*12+e
d_28=[datetime(m//12, m%12+1, 28) for m in range(startyear*12+startmonth-1, endyear*12+e

#creates list of dates
d_1.extend(d_4)
d_1.extend(d_7)
d_1.extend(d_10)
d_1.extend(d_13)
d_1.extend(d_16)
d_1.extend(d_19)
d_1.extend(d_22)
d_1.extend(d_25)
d_1.extend(d_28)
d_1.sort()

conn = lite.connect(f'/home/haakon/Documents/Vestfjorden{startyear}.db')
c = conn.cursor()
c.execute("""CREATE TABLE IF NOT EXISTS Domain_violation_2 (ID INT,Time DATETIME, mmsi_0
```

```
                        distance_center   float, domain_overlap_0_2   float, domain_over
                        cog_2 float,t_lat_0 float,t_lon_0 float,t_lat_2   float,t_lon_2
                        width_0  float,width_2 float,Vessel_name_0    TEXT,Vessel_name_2
index1= (f"CREATE INDEX IF NOT EXISTS mmsi_index_reindex ON reindexed_{startyear}(MMSI);
c.execute(index1)
index2= (f"CREATE INDEX IF NOT EXISTS time_index_reindex ON reindexed_{startyear}(Time);
c.execute(index2)

write_column_names=['ID','timestamp_0','mmsi_0','mmsi_2','imo_0','imo_2','Violated_domai
                    'domain_overlap_0_2','domain_overlap_2_0','v_0','v_2','cog_0','cog_2
                    't_lat_2','t_lon_2','length_0','length_2','width_0','width_2','Vesse
                    'Vessel_name_2']
with open(r'dom_vio16.csv', 'a') as f:
    writer = csv.writer(f)
    writer.writerow(write_column_names)
par = Parallel(n_jobs=16, verbose=10)
do_something = delayed(violations)
par(do_something(i) for i in d_1)
```

A.3 Sorting algorithm

```
# -*- coding: utf-8 -*-
"""
Created on Mon Apr 30 18:07:11 2018
@author: haano
"""
import sqlite3 as lite #sql
import numpy as np
import csv
import pandas as pd
from datetime import datetime
from datetime import timedelta
from geographiclib.geodesic import Geodesic
from formulas import l_a, True_center, cog_delta, alpha

def rate_of_turn(entries,threshold):
    if len(entries)<4:
        return 0
    else:
        rot=[]
        header=('time','mmsi','lat','lon','sog','cog')
        entries_df = pd.DataFrame(entries, columns=header)
        entries_df=entries_df.drop_duplicates(subset='time',keep='first')
        entries_df = entries_df.reset_index(drop=True)
        for i in range(1,len(entries)-1):
            try:
                cog_next=entries_df.loc[i+1][5]
                cog=entries_df.loc[i][5]
                cog_prev=entries_df.loc[i-1][5]
#                cog_prev_p=entries[i-2][5]
                time_next=datetime.strptime(entries_df.loc[i+1][0], "%Y-%m-%d %H:%M:%S")
                time=datetime.strptime(entries_df.loc[i][0], "%Y-%m-%d %H:%M:%S")
                time_prev=datetime.strptime(entries_df.loc[i-1][0], "%Y-%m-%d %H:%M:%S")
#                time_prev_p=datetime.strptime(entries[i-2][0], "%Y-%m-%d %H:%M:%S")
#                time_delta_1=(time_prev-time_prev_p).seconds/60
                time_delta_2=(time-time_prev).seconds/60
                time_delta_3=(time_next-time).seconds/60
#                rot_MA=((cog_delta(cog_prev,cog_prev_p)/time_delta_1)+(cog_delta(cog,co
                rot_MA=((cog_delta(cog,cog_prev)/time_delta_2)+(cog_delta(cog_next,cog)/
                rot.append(rot_MA)
            except:
                pass
    try:
        if max(rot)> threshold:
            return (1,max(rot))
        else:
            return (0,max(rot))
    except:
        return(-2,101)
def speed_mean(entries):
    sogs=[]
    for i in entries:
        try:
            sogs.append(float(i[4]))
        except:
            pass
    return np.nanmean(sogs).astype(np.float)
def format_timedelta(td):
    if td < timedelta(0):
        return '-' + format_timedelta(-td)
    else:
```

```
        return str(td)
def give_way(lat_1_t0,lon_1_t0,lat_2_t0,lon_2_t0,cog_1_t0,mmsi_1,mmsi_2):
    if 0 <= alpha(lon_1_t0,lat_1_t0,lon_2_t0,lat_2_t0,cog_1_t0)<=np.pi:
        return(mmsi_1)
    else:
        return(mmsi_2)
def interaction(lat_1,lon_1,lat_2,lon_2,cog_1,cog_2,lat_1_t0,lon_1_t0,lat_2_t0,lon_2_t0,
    try:
        if abs(cog_1-cog_2)<25 or (min(cog_1,cog_2)+(360-max(cog_1,cog_2)))<25:
            interaction='Overtaking'
            vessel_give= 0
        elif 165<abs(cog_1-cog_2)<195:
            interaction='Head_on'
            vessel_give=2
        else:
            if give_way(lat_1_t0,lon_1_t0,lat_2_t0,lon_2_t0,cog_1_t0,mmsi_1,mmsi_2)==mms
                vessel_give=mmsi_1
            #angle form stand on vessel to give way vessel if mmsi_1 gives way
                if np.pi/2<alpha(lon_2, lat_2, lon_1, lat_1, cog_2)<(3/2)*np.pi:
                    interaction='Crossing passing at stern'
                else:
                    interaction='Crossing passing at bow'
            else:
                vessel_give=mmsi_2
                if np.pi/2<alpha(lon_1, lat_1, lon_2, lat_2, cog_1)<(3/2)*np.pi:
                    interaction='Crossing passing at stern'
                else:
                    interaction='Crossing passing at bow'
    except:
        interaction = 'unknown'
        vessel_give = 0
    return (interaction,vessel_give)


def dom(dom_id, dom_start, dom_end,j):
    c.execute(f"SELECT *, min(distance_center) FROM Domain_violation_2 WHERE ID = {dom_i
    entry=c.fetchone()
    try:
        time_CPA=datetime.strptime(entry[1], "%Y-%m-%d %H:%M:%S")
    except:
        print(dom_id,'  ',entry[1])
    start_init=str(time_CPA-timedelta(minutes=3))
    end_init=str(time_CPA+timedelta(minutes=2))
    mmsi_1=entry[2]
    mmsi_2=entry[3]
    dom_vio=entry[6]
    c.execute(f"SELECT * FROM Statistics_{year} WHERE MMSI = {mmsi_1};")
    ship_1_info=c.fetchone()
    c.execute(f"SELECT * FROM Statistics_{year} WHERE MMSI = {mmsi_2};")
    ship_2_info=c.fetchone()
    c.execute(f"select Time, MMSI, Latitude, Longitude,SOG,COG  from raw_{year} WHERE MM
    entries_init_1=c.fetchall()
    lat_1_t0=entries_init_1[0][2]
    lon_1_t0=entries_init_1[0][3]
    cog_1_t0=entries_init_1[0][5]
    N_entries_1=len(entries_init_1)
    if N_entries_1 < 10:
            r_1=(0,0)
            s_1=0
    else:
```

```
    r_1=rate_of_turn(entries_init_1,70)
    s_1=speed_mean(entries_init_1)
c.execute(f"select Time, MMSI, Latitude, Longitude,SOG,COG  from raw_{year} WHERE MM
entries_init_2=c.fetchall()
lat_2_t0=entries_init_2[0][2]
lon_2_t0=entries_init_2[0][3]
N_entries_2=len(entries_init_2)
if N_entries_2 < 10:
    r_2=(0,0)
    s_2=0
else:
    r_2=rate_of_turn(entries_init_2,70)
    s_2=speed_mean(entries_init_2)
sorting=0
if ((r_1[0]==1 and dom_vio == mmsi_1) or (r_2[0]==1 and dom_vio == mmsi_2) or ((r_1[
    sorting=1

    if ship_1_info[4]==ship_2_info[4]=='Fishing':
        with open(f'ff_{dom_id}_{j}.csv', 'w', newline='', encoding='utf-8') as out:
            csv_out = csv.writer(out, delimiter=',')
            csv_out.writerow(top_row)
    else:
        with open(f'{dom_id}_{j}.csv', 'w', newline='', encoding='utf-8') as out:
            csv_out = csv.writer(out, delimiter=',')
            csv_out.writerow(top_row)
start=str(time_CPA-timedelta(minutes=30))
end=str(time_CPA+timedelta(minutes=30))
c.execute(f"select * from reindexed_{year} WHERE (MMSI = '{mmsi_1}' OR MMSI = '{mmsi
entries=c.fetchall()
a=[]
b=[]
if len(entries)>5:
    for idx in range(len(entries)-1):
        if entries[idx][0] == entries[idx+1][0]:
            cog_1=entries[idx][5]
            if type(cog_1) == str:
                try:
                    int(entries[idx-2][5])
                    cog_1=entries[idx-2][5]
                except:
                    cog_1=0.1
            sog_1=entries[idx][2]
            if type(sog_1) == str:
                try:
                    int(entries[idx-2][2])
                    sog_1=entries[idx-2][2]
                except:
                    sog_1=0.1
            time_0=entries[idx][0]
            cog_2=entries[idx+1][5]
            if type(cog_2) == str:
                try:
                    int(entries[idx-1][5])
                    cog_2=entries[idx-1][5]
                except:
                    cog_2=0.1
            sog_2=entries[idx+1][2]
            if type(sog_2) == str:
                try:
                    int(entries[idx-2][2])
```

```
        sog_2=entries[idx-2][2]
    except:
        sog_2=0.1
t_lat_1=True_center(entries[idx][4],entries[idx][3],entries[idx][8],entr
t_lon_1=True_center(entries[idx][4],entries[idx][3],entries[idx][8],entr
t_lat_2=True_center(entries[idx+1][4],entries[idx+1][3],entries[idx+1][8
t_lon_2=True_center(entries[idx+1][4],entries[idx+1][3],entries[idx+1][8
distance=Geodesic.WGS84.Inverse(t_lat_1,t_lon_1,t_lat_2,t_lon_2)['s12']
try:
    length_1=(entries[idx][8]+entries[idx][9])
except:
    length_1=40.001
try:
    length_2=(entries[idx+1][8]+entries[idx+1][9])
except:
    length_2=40.001
#domain length in direction from ship 1 to ship 2 [meter]
dom_len_1_2=l_a(length_1, t_lat_1, t_lon_1, t_lat_2, t_lon_2, cog_1,sog_
#domain length in direction from ship 2 to ship 0 [meter]
dom_len_2_1=l_a(length_2, t_lat_2, t_lon_2, t_lat_1, t_lon_1, cog_2,sog_
#distance center to center [meter]
distance=Geodesic.WGS84.Inverse(t_lat_1,t_lon_1, t_lat_2,t_lon_2)['s12']
#negative values indicate domain violation
overlap_ship2_into_shipdomain_1=distance-dom_len_1_2
overlap_ship1_into_shipdomain_2=distance-dom_len_2_1
if str(time_CPA) == time_0:
    CPA=1
    lat_1_C=t_lat_1
    lon_1_C=t_lon_1
    lat_2_C=t_lat_2
    lon_2_C=t_lon_2
    cog_1_C=cog_1
    cog_2_C=cog_2
    sog_1_C=sog_1
    sog_2_C=sog_2
    overlap_c_2_1=overlap_ship2_into_shipdomain_1
    overlap_c_1_2=overlap_ship1_into_shipdomain_2
    distance_c=distance
    dom_len_1_c=l_a(length_1, t_lat_1, t_lon_1, t_lat_2, t_lon_2, cog_1,
    dom_len_2_c=l_a(length_2, t_lat_2, t_lon_2, t_lat_1, t_lon_1, cog_2,

    try:
        max_rot_1=r_1[1]
    except:
        max_rot_1=-1
    try:
        max_rot_2=r_2[1]
    except:
        max_rot_2=-1
    encounter=interaction(lat_1_C,lon_1_C,lat_2_C,lon_2_C,cog_1_C,cog_2_
    give_way_vessel=interaction(lat_1_C,lon_1_C,lat_2_C,lon_2_C,cog_1_C,
    if encounter == 'Overtaking':
        if s_1>s_2:
            give_way_vessel=mmsi_1
        else:
            give_way_vessel=mmsi_2
else:
    CPA=0
TCPA=format_timedelta(datetime.strptime(time_0, "%Y-%m-%d %H:%M:%S")-tim
fields_1=(dom_id,time_0,mmsi_1,distance,dom_len_1_2,overlap_ship2_into_s
```

```
                fields_2=(dom_id,time_0,mmsi_2,distance,dom_len_2_1,overlap_ship1_into_s
                a.append(fields_1)
                b.append(fields_2)
        a.extend(b)

        c.execute(f"INSERT INTO sorted_70_{year} VALUES(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,
                      (dom_id,time_CPA,distance_c,mmsi_1,mmsi_2,dom_len_1_c,dom_len_2_c,
                       give_way_vessel,max_rot_1,max_rot_2,lat_1_C,lon_1_C,lat_2_C,lon_2
                       ship_1_info[1],ship_2_info[1],ship_1_info[3],ship_2_info[3],ship_
                       ship_2_info[5],ship_1_info[6],ship_2_info[6],length_1,length_2,so
        conn.commit()
        if sorting==1:
            if ship_1_info[4]==ship_2_info[4]=='Fishing':
                with open(f'ff_{dom_id}_{j}.csv', 'a', newline='', encoding='utf-8') as
                    csv_out = csv.writer(out, delimiter=',')
                    csv_out.writerows(a)
            else:
                with open(f'{dom_id}_{j}.csv','a', newline='', encoding='utf-8') as f:
                    writer = csv.writer(f)
                    writer.writerows(a)

year = input("input year form 2013-2016: ")
conn = lite.connect(f'/home/haakon/Documents/Vestfjorden{year}.db')
c = conn.cursor()
c.execute(f"CREATE TABLE IF NOT EXISTS sorted_70_{year} (dom_id int,time_CPA datetime,di
          encounter text,give_way_vessel int,max_rot_1 float,max_rot_2 float,lat_1_C flo
          cog_2_C float, IMO_1 int,IMO_2 int,Flag_1 text,Flag_2 text,Vessel_type_1 text,
          Sailed_distance_2 float,N_trips_1 int,N_trips_2 int,Length_1 int,Length_2 int,
top_row=('Identifyier','Time','mmsi','Distance','Domain_length','domain_violation','CPA'

c.execute("SELECT distinct ID FROM Domain_violation_2 ORDER BY ID")
violation_ID=[int(i[0]) for i in c.fetchall()]
for x in violation_ID:
#    print(x)
    c.execute(f"SELECT * FROM Domain_violation_2 WHERE ID = {x};")
    entries=c.fetchall()
    dates=[]
    dates.append(datetime.strptime(entries[0][1],"%Y-%m-%d %H:%M:%S"))
    for i in range(1,len(entries)):
        g=datetime.strptime(entries[i][1],"%Y-%m-%d %H:%M:%S")
        f=datetime.strptime(entries[i-1][1],"%Y-%m-%d %H:%M:%S")
        hours=(g-f).seconds/3600
        days=(g-f).days
        if days>1 or hours>4:
#            dates.append(datetime.strptime(dom_vios[i][1],"%Y-%m-%d %H:%M:%S"))
            dates.append(g)
#    print(x,len(dates))
    j=0
    for i in dates:
        j+=1
        dom_start=str(i-timedelta(minutes=60))
        dom_end=str(i+timedelta(minutes=60))
        dom(x, dom_start, dom_end,j)
```

A.4 Statistics

```
# -*- coding: utf-8 -*-
"""
Created on Sun May  6 18:42:42 2018

@author: haano
"""
#python3.6 /home/haakon/Documents/par_statistics.py
from joblib import Parallel, delayed
import numpy as np
import pandas as pd
from datetime import datetime
from datetime import timedelta
from dateutil.relativedelta import relativedelta
import sqlite3 as lite
#import time
import multiprocessing
import csv
from geographiclib.geodesic import Geodesic
ship_type=['Reserved', 'Reserved', 'Reserved', 'Reserved', 'Reserved', 'Reserved', 'Rese
           'Reserved', 'Reserved', 'Reserved', 'Reserved', 'Wing In Grnd', 'Wing In Grnd
           'Wing In Grnd', 'Wing In Grnd', 'Wing In Grnd', 'Wing In Grnd',\
           'Wing In Grnd', 'Wing In Grnd', 'Wing In Grnd', 'SAR Aircraft', 'Fishing', 'T
           'Tug', 'Dredger', 'Dive Vessel', 'Military Ops', 'Sailing Vessel', 'Pleasure
           'Reserved', 'Reserved', 'High-Speed Craft', 'High-Speed Craft', 'High-Speed C
           'High-Speed Craft', 'High-Speed Craft', 'High-Speed Craft', 'High-Speed Craft
           'High-Speed Craft', 'High-Speed Craft', 'High-Speed Craft', 'Pilot Vessel', '
           'Tug', 'Port Tender', 'Anti-Pollution', 'Law Enforce', 'Local Vessel', 'Local
           'Medical Trans', 'Special Craft', 'Passenger', 'Passenger', 'Passenger', 'Pas
           'Passenger', 'Passenger', 'Passenger', 'Passenger', 'Passenger', 'Passenger',
           'Cargo - Hazard A (Major)', 'Cargo - Hazard B', 'Cargo - Hazard C (Minor)',\
           'Cargo - Hazard D (Recognizable)', 'Cargo', 'Cargo', 'Cargo', 'Cargo', 'Cargo
           'Tanker', 'Tanker - Hazard A (Major)', 'Tanker - Hazard B',\
           'Tanker - Hazard C (Minor)', 'Tanker - Hazard D (Recognizable)', 'Tanker', 'T
           'Tanker','Tanker', 'Tanker', 'Other', 'Other', 'Other',\
           'Other', 'Other', 'Other', 'Other', 'Other', 'Other', 'Other']
country=['Albania', 'Andorra', 'Austria', 'Portugal', 'Belgium', 'Belarus', 'Bulgaria',
         'Cyprus', 'Germany', 'Cyprus', 'Georgia', 'Moldova', 'Malta', 'Armenia', 'Germa
         'Spain', 'Spain', 'France', 'France', 'France', 'Malta', 'Finland', 'Faroe Is',
         'United Kingdom', 'United Kingdom', 'Gibraltar', 'Greece', 'Croatia', 'Greece',
         'Hungary', 'Netherlands', 'Netherlands', 'Netherlands', 'Italy', 'Malta', 'Malt
         'Luxembourg', 'Monaco', 'Portugal', 'Malta', 'Norway', 'Norway', 'Norway', 'Pol
         'Sweden', 'Sweden', 'Slovakia', 'San Marino', 'Switzerland', 'Czech Republic',
         'Latvia', 'Estonia', 'Lithuania', 'Slovenia', 'Serbia', 'Anguilla', 'USA', 'Ant
         'Aruba', 'Bahamas', 'Bahamas', 'Bermuda', 'Bahamas', 'Belize', 'Barbados', 'Can
         'Dominica', 'Dominican Rep', 'Guadeloupe', 'Grenada', 'Greenland', 'Guatemala',
         'St Kitts Nevis', 'St Lucia', 'Mexico', 'Martinique', 'Montserrat', 'Nicaragua'
         'Panama', 'Panama', 'Panama', 'Puerto Rico', 'El Salvador', 'St Pierre Miquelon
         'USA', 'USA', 'USA', 'Panama', 'Panama', 'Panama', 'Panama', 'Panama', 'St Vinc
         'St Vincent Grenadines', 'British Virgin Is', 'US Virgin Is', 'Afghanistan', 'S
         'China', 'China', 'China', 'Taiwan', 'Sri Lanka', 'India', 'Iran', 'Azerbaijan'
         'Kazakhstan', 'Uzbekistan', 'Jordan', 'Korea', 'Korea', 'Palestine', 'DPR Korea
         'Maldives', 'Mongolia', 'Nepal', 'Oman', 'Pakistan', 'Qatar', 'Syria', 'UAE', '
         'Bosnia and Herzegovina', 'Antarctica', 'Australia', 'Myanmar', 'Brunei', 'Micr
         'Christmas Is', 'Cook Is', 'Fiji', 'Cocos Is', 'Indonesia', 'Kiribati', 'Laos',
         'New Caledonia', 'Niue', 'Nauru', 'French Polynesia', 'Philippines', 'Papua New
         'Samoa', 'Singapore', 'Singapore', 'Singapore', 'Singapore', 'Thailand', 'Tonga
         'Wallis Futuna Is', 'South Africa', 'Angola', 'Algeria', 'St Paul Amsterdam Is'
         'Cen Afr Rep', 'Cameroon', 'Congo', 'Comoros', 'Cape Verde', 'Antarctica', 'Ivo
         'Eritrea', 'Gabon', 'Ghana', 'Gambia', 'Guinea-Bissau', 'Equ. Guinea', 'Guinea'
         'Liberia', 'Libya', 'Lesotho', 'Mauritius', 'Madagascar', 'Mali', 'Mozambique',
```

```
         'Reunion', 'Rwanda', 'Sudan', 'Senegal', 'Seychelles', 'St Helena', 'Somalia',
         'Togo', 'Tunisia', 'Tanzania', 'Uganda', 'DR Congo', 'Tanzania', 'Zambia', 'Zim
         'Colombia', 'Ecuador', 'UK', 'Guiana', 'Guyana', 'Paraguay', 'Peru', 'Suriname'
Mids=[201, 202, 203, 204, 205, 206, 207, 208, 209, 210, 211, 212, 213, 214, 215, 216, 21
      233, 234, 235, 236, 237, 238, 239, 240, 241, 242, 243, 244, 245, 246, 247, 248, 24
      262, 263, 264, 265, 266, 267, 268, 269, 270, 271, 272, 273, 274, 275, 276, 277, 27
      312, 314, 316, 319, 321, 323, 325, 327, 329, 330, 331, 332, 334, 336, 338, 339, 34
      357, 358, 359, 361, 362, 364, 366, 367, 368, 369, 370, 371, 372, 373, 374, 375, 37
      416, 417, 419, 422, 423, 425, 428, 431, 432, 434, 436, 437, 438, 440, 441, 443, 44
      470, 472, 473, 475, 477, 478, 501, 503, 506, 508, 510, 511, 512, 514, 515, 516, 51
      546, 548, 553, 555, 557, 559, 561, 563, 564, 565, 566, 567, 570, 572, 574, 576, 57
      615, 616, 617, 618, 619, 620, 621, 622, 624, 625, 626, 627, 629, 630, 631, 632, 63
      655, 656, 657, 659, 660, 661, 662, 663, 664, 665, 666, 667, 668, 669, 670, 671, 67
      735, 740, 745, 750, 755, 760, 765, 770, 775]
mids_country=np.column_stack((Mids,country))


code=['A11','A12','A13','A14','A21','A22','A23','A24','A31','A32','A33','A34','A35','A36
vessel_types=['Tanker','Tanker','Tanker','Tanker','Bulk','Bulk','Bulk','Bulk','Cargo', '
              'Fishing','Fishing','Offshore','Offshore','Research','Tug','Dredger','Othe
statco=np.column_stack((code,vessel_types))
def stat_type(s_code):
    n=str(s_code)[2:5]
    try:
        index=int(np.where(statco==f'{n}')[0])
        return str(statco[index][1])
    except:
        raise Exception
def vessel_type(n):
    return(ship_type[n-9])
def flag(n):
    n=int(str(n)[:3])
    try:
        index=int(np.where(mids_country==f'{n}')[0])
        return str(mids_country[index][1])
    except:
        return "not valid"
def stats(mmsi):
    #set year
    ship=int(mmsi[0])
    year=2016
    multiprocessing.Lock().acquire()
    conn = lite.connect(f'/home/haakon/Documents/Vestfjorden{year}.db')
    c = conn.cursor()
   c.execute(f"SELECT Time,MMSI,IMO, Longitude,Latitude,SOG,Vessel_name FROM raw_{year}
    entries=c.fetchall()
    c.execute(f"SELECT IMO,MMSI, AIS_vessel_type, Vessel_name  From static_web_{year} WH
    vessel_info=c.fetchone()
    distance=0.0
    N_trips=0

    for index in range(1,len(entries)):
        #Prevent distance being calculated from GPS inaccuracies(at quay) and while anch
        a=entries[index-2][5]
        if type(a)==str:
            a=float('NaN')
        b=entries[index-1][5]
        if type(b)==str:
            b=float('NaN')
```

```
        d=entries[index][5]
        if type(d)==str:
            d=float('NaN')
        if np.nanmean([a,b,d]).astype(np.float) < 0.2:
            pass
        else:
            if N_trips == 0:
                #indicating first time in area
                N_trips+=1
                last_measurement=datetime.strptime(entries[index][0], "%Y-%m-%d %H:%M:%S
            time_b=datetime.strptime(entries[index-1][0], "%Y-%m-%d %H:%M:%S")
            time=datetime.strptime(entries[index][0], "%Y-%m-%d %H:%M:%S")
            lon_b=entries[index-1][3]
            lat_b=entries[index-1][4]
            lon=entries[index][3]
            lat=entries[index][4]
            #One hour between transmissions is interpreted as leaving the case study are
            if (time-time_b)>=timedelta(hours=1):
#                N_trips+=1
#                print(time,"hour")
#                removes distance from leaving and entering area at different coordinate
                distance-=Geodesic.WGS84.Inverse(lat,lon,lat_b,lon_b)['s12']
            #will account for not sailing during at quay or anchored, e.g. ropax ferry t
            if (time-last_measurement)>=timedelta(minutes=9):
                N_trips+=1
            #sums distances between all AIS messages
            distance+=Geodesic.WGS84.Inverse(lat,lon,lat_b,lon_b)['s12']
            last_measurement=time=datetime.strptime(entries[index][0], "%Y-%m-%d %H:%M:%

    IMO=vessel_info[0]
    MMSI=ship
    Flag=flag(MMSI)
    Vessel_type=vessel_info[2]
    Vessel_name=vessel_info[3]
    fields=[MMSI,IMO,Vessel_name,Flag,Vessel_type,distance,N_trips]
    multiprocessing.Lock().acquire()
    with open(f'Statistics_{year}.csv', 'a') as f:
        writer = csv.writer(f)
        writer.writerow(fields)
#Set year
year=2016
write_column_names=['MMSI','IMO','Vessel_name','Flag_state','Vessel_type','Distance','Nu
conn = lite.connect(f'/home/haakon/Documents/Vestfjorden{year}.db')
c = conn.cursor()
c.execute(f"CREATE TABLE IF NOT EXISTS Statistics_{year}(MMSI int, IMO int,Vessel_name t
c.execute(f"SELECT DISTINCT MMSI FROM raw_{year}")
unique_mmsi=c.fetchall()
with open(f'Statistics_{year}.csv', 'w') as f:
    writer = csv.writer(f)
    writer.writerow(write_column_names)
par = Parallel(n_jobs=16, verbose=10)
do_something = delayed(stats)
par(do_something(mmsi) for mmsi in unique_mmsi)
```

A.5 Web scraping

```
# -*- coding: utf-8 -*-
"""
Created on Tue Mar 13 12:44:58 2018

@author: haano
"""

import sqlite3 as lite #sql
import numpy as np
import pandas as pd
#web
import requests
from bs4 import BeautifulSoup
year=input("write year: ")
conn = lite.connect(f'/home/haakon/Documents/Vestfjorden{year}.db')
c = conn.cursor()
c.execute(f"select distinct mmsi from raw_{year}")
unique_mmsi=c.fetchall()

c.execute(f"CREATE TABLE IF NOT EXISTS static_web_{year} (IMO int, MMSI int,Flag text,Gr
#c.execute("drop table static_web_data")
mmsi_list=pd.DataFrame(unique_mmsi)
#ensure same type for all entries
mmsi_list=mmsi_list.apply(lambda x: pd.to_numeric(x, errors='coerce'))
mmsi_list=mmsi_list.values.tolist()

mmsi_list=[i[0]for i in mmsi_list]

user = {'User-Agent':'Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.36 (KHTML, like Gecko

for mmsi in mmsi_list:
#    mmsi=str(t)
    c.execute(f"SELECT imo, vessel_name FROM raw_{year} WHERE mmsi = '{mmsi}'")
    imo=c.fetchone()[0]
    try:
        vessel_name=c.fetchone()[1]
    except:
        vessel_name="none"
    try:
        try:
            MMSI=int(mmsi)
            IMO=int(imo)
            url=f"https://www.marinetraffic.com/en/ais/details/ships/imo:{imo}/"
            r = requests.get(url, headers=user)
            soup= BeautifulSoup(r.text,'html.parser')
            result= soup.find_all('h1')
            web_name=result[0].text
            if web_name != vessel_name:
                print("error imo: ",web_name," ",vessel_name," ",mmsi)
            url=f"https://www.marinetraffic.com/en/ais/details/ships/imo:{imo}/"
            r = requests.get(url, headers=user)
            soup= BeautifulSoup(r.text,'html.parser')
            result= soup.find_all('b')
            IMO_web=np.int(result[4].text)
            Flag=result[7].text
            AIS_vessel_type=result[8].text
            GT=np.int(result[9].text)
            Dimensions=result[11].text
            Year_built=np.int(result[12].text)
            result= soup.find_all('a')
```

```
                category=result[39].text[18:]
                c.execute('''INSERT INTO static_web_{year} VALUESVALUES(?,?,?,?,?,?,?,?,?,?)
                conn.commit()

            except:
                url=f"https://www.marinetraffic.com/en/ais/details/ships/mmsi:{mmsi}/"
                r = requests.get(url, headers=user)
                soup= BeautifulSoup(r.text,'html.parser')
                result= soup.find_all('h1')
                web_name=result[0].text
                if web_name != vessel_name:
                    print("error mmsi: ",web_name," ",vessel_name," ",mmsi)
#                    raise Exception
                url=f"https://www.marinetraffic.com/en/ais/details/ships/mmsi:{mmsi}/"
                r = requests.get(url, headers=user)
                soup= BeautifulSoup(r.text,'html.parser')
                result= soup.find_all('b')
                IMO=imo
#                MMSI=int(mmsi)
                try:
                    Flag=result[7].text
                except:
                    Flag='not found'
                try:
                    AIS_vessel_type=result[8].text
                except:
                    AIS_vessel_type='not found'
                try:
                    GT=np.int(result[9].text)
                except:
                    GT=0
                try:
                    Dimensions=result[11].text
                except:
                    Dimensions='not found'
                try:
                    Year_built=np.int(result[12].text)
                except:
                    Year_built=0
                result= soup.find_all('a')
                try:
                    category=result[39].text[18:]
                except:
                    category='not found'
                c.execute(f'''INSERT INTO static_web_{year} VALUES(?,?,?,?,?,?,?,?,?,?)''',(
                conn.commit()
        except:
            try:
                IMO=int(imo)
                url=f"https://www.fleetmon.com/vessels/?s={IMO}"
                r = requests.get(url, headers=user)
                soup= BeautifulSoup(r.text,'html.parser')
                res=soup.find_all('a')
                web_name=res[54].text
                if web_name != vessel_name:
                    if web_name != vessel_name:
                        print("error fleetmon imo: ",web_name," ",vessel_name," ",mmsi)
                url=f"https://www.fleetmon.com/vessels/?s={IMO}"
                r = requests.get(url, headers=user)
                soup= BeautifulSoup(r.text,'html.parser')
```

```
                res=soup.find_all('td')
                flag=res[0].contents
                *other,flag=str(flag[1]).split()
                Flag=flag[:-3]
                Dimensions=res[6].contents
                Dimensions=str(Dimensions)[4:-4]
                res=soup.find_all('span')
                category=res[16].text
#                MMSI=int(mmsi)
                AIS_vessel_type='not found'
                Year_built=0
                GT=0
                c.execute('''INSERT INTO static_web_{year} VALUES(?,?,?,?,?,?,?,?,?,?)''',(I
                conn.commit()
            except:
                print("error : ",  mmsi)
                Year_built=9999
                IMO=9999
#                MMSI=int(mmsi)
                Dimensions='not found'
                Flag='not found'
                category='not found'
                AIS_vessel_type='not found'
                GT=9999
                c.execute(f'''INSERT INTO static_web_{year} VALUES(?,?,?,?,?,?,?,?,?,?)''',(
                conn.commit()
                667001255
```

A.6 Formulas

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri Jun  1 17:15:28 2018

@author: haakon
"""
import matplotlib.pyplot as plt
import sqlite3 as lite
import numpy as np
import pandas as pd
from datetime import datetime
from datetime import timedelta
from geographiclib.geodesic import Geodesic
def cog_delta(cog_1,cog_2):
    if abs(cog_1-cog_2)<=180:
        return abs(cog_1-cog_2)
    else:
        return (min(cog_1,cog_2)+(360-max(cog_1,cog_2)))
def rate_of_turn(entries):
    rot=[]
    header=('time','mmsi','sog','cog','lon','lat')
    entries_df = pd.DataFrame(entries, columns=header)
    entries_df=entries_df.drop_duplicates(subset='time',keep='first')
    entries_df = entries_df.reset_index(drop=True)
    for i in range(1,len(entries)-1):
        cog_next=entries_df.loc[i+1][3]
        cog=entries_df.loc[i][3]
        cog_prev=entries_df.loc[i-1][3]
        time_next=datetime.strptime(entries_df.loc[i+1][0], "%Y-%m-%d %H:%M:%S")
        time=datetime.strptime(entries_df.loc[i][0], "%Y-%m-%d %H:%M:%S")
        time_prev=datetime.strptime(entries_df.loc[i-1][0], "%Y-%m-%d %H:%M:%S")
        time_delta_2=(time-time_prev).seconds/60
        time_delta_3=(time_next-time).seconds/60
        rot_MA=((cog_delta(cog,cog_prev)/time_delta_2)+(cog_delta(cog_next,cog)/time_del
        rot.append((time.time(),rot_MA))
    return(rot)
def alpha(lon_1, lat_1, lon_2, lat_2, cog_1):
    azi1=Geodesic.WGS84.Inverse(lat_1,lon_1, lat_2,lon_2)['azi1']
    if type(cog_1) == str:
        if azi1<0:
            return np.radians(360+azi1)
        else:
            return np.radians(azi1)
    else:
        if azi1<0:
            azi1=360+azi1
        if azi1<=cog_1:
            return np.radians(360-(cog_1-azi1))
        else:
            return np.radians(azi1-cog_1)
def l_a(length_1,lat_1, lon_1, lat_2, lon_2, cog_1,v_1):
    k_AD=10**(0.3591*np.log10(v_1)+0.0952)
    k_DT=10**(0.5441*np.log10(v_1)-0.0795)
    #course unknown cicular domain
    if type(cog_1) == str:
        R_fore=R_aft=R_starb=R_port=(0.2+k_DT)*length_1
    else:
        R_fore=(1+1.34*np.sqrt(k_AD**2+(k_DT/2)**2))*length_1
        R_aft=(1+0.67*np.sqrt(k_AD**2+(k_DT/2)**2))*length_1
```

```
        R_starb=(0.2+k_DT)*length_1
        R_port=(0.2+0.75*k_DT)*length_1
    if alpha(lon_1, lat_1, lon_2, lat_2, cog_1)<=np.pi/2:
        l =(((1+np.tan(alpha(lon_1, lat_1, lon_2, lat_2, cog_1))**2)/((1/R_fore**2)+(np.
    elif np.pi/2 < alpha(lon_1, lat_1, lon_2, lat_2, cog_1) and alpha(lon_1, lat_1, lon_
        l =(((1+np.tan(alpha(lon_1, lat_1, lon_2, lat_2, cog_1))**2)/((1/R_aft**2)+(np.t

    elif np.pi < alpha(lon_1, lat_1, lon_2, lat_2, cog_1) and alpha(lon_1, lat_1, lon_2,
        l =(((1+np.tan(alpha(lon_1, lat_1, lon_2, lat_2, cog_1))**2)/((1/R_aft**2)+(np.t

    elif (3/2)*np.pi < alpha(lon_1, lat_1, lon_2, lat_2, cog_1):
        l =(((1+np.tan(alpha(lon_1, lat_1, lon_2, lat_2, cog_1))**2)/((1/R_fore**2)+(np.
    return l
def True_center(lat_1,lon_1,d_bow,d_aft,d_starb,d_port,cog_1,year):
    if type(cog_1) == str:
        return(lat_1,lon_1)
    elif type(d_bow) == str:
        return(lat_1,lon_1)
    elif type(d_aft) == str:
        return(lat_1,lon_1)
    elif type(d_starb) == str:
        d_starb=d_port=1
    elif type(d_port) == str:
        d_port=d_starb=1
    else:
        ##hack for 2013/2014 missing d_port
        if year == (2013 or 2014):
            d_starb=d_port
        try:
            if d_bow>d_aft:
                dist_x=(d_bow+d_aft)/2-d_aft
                dir_x=1
                if d_starb>d_port:
                    dist_y=(d_starb+d_port)/2-d_port
                    dir_y=1
                    s_12=np.sqrt(dist_x**2+dist_y**2)
                elif d_starb<d_port:
                    dist_y=(d_starb+d_port)/2-d_port
                    dir_y=2
                    s_12=np.sqrt(dist_x**2+dist_y**2)
                elif d_starb==d_port:
                    dir_y=0
                    dist_y=0
                    s_12=np.sqrt(dist_x**2+dist_y**2)
            elif d_bow<d_aft:
                dist_x=(d_bow+d_aft)/2-d_bow
                dir_x=2
                if d_starb>d_port:
                    dist_y=(d_starb+d_port)/2-d_port
                    dir_y=1
                    s_12=np.sqrt(dist_x**2+dist_y**2)
                elif d_starb<d_port:
                    dist_y=(d_starb+d_port)/2-d_port
                    dir_y=2
                    s_12=np.sqrt(dist_x**2+dist_y**2)
                elif d_starb==d_port:
                    dir_y=0
                    dist_y=0
                    s_12=np.sqrt(dist_x**2+dist_y**2)
            elif d_bow==d_aft:
```

```
            dir_x=0
            if d_starb>d_port:
                dist_y=(d_starb+d_port)/2-d_port
                dir_y=1
                s_12=np.sqrt(dist_x**2+dist_y**2)
            elif d_starb<d_port:
                dist_y=(d_starb+d_port)/2-d_port
                dir_y=2
                s_12=np.sqrt(dist_x**2+dist_y**2)
            elif d_starb==d_port:
                dir_y=0
                dist_y=0
                s_12=np.sqrt(dist_x**2+dist_y**2)
        if dir_x==0 and dir_y==0:
            t_lon_1=lon_1
            t_lat_1=lat_1
        elif dir_x==1 and dir_y==0:
            vector=Geodesic.WGS84.Direct(lat_1,lon_1,cog_1,s_12,outmask=1929)
            t_lon_1=vector['lon2']
            t_lat_1=vector['lat2']
        elif dir_x==2 and dir_y==0:
            vector=Geodesic.WGS84.Direct(lat_1,lon_1,(cog_1+180),s_12,outmask=1929)
            t_lon_1=vector['lon2']
            t_lat_1=vector['lat2']
        elif dir_x==0 and dir_y==1:
            vector=Geodesic.WGS84.Direct(lat_1,lon_1,(cog_1+90),s_12,outmask=1929)
            t_lon_1=vector['lon2']
            t_lat_1=vector['lat2']
        elif dir_x==0 and dir_y==2:
            vector=Geodesic.WGS84.Direct(lat_1,lon_1,(cog_1-90),s_12,outmask=1929)
            t_lon_1=vector['lon2']
            t_lat_1=vector['lat2']
        elif dir_x==1 and dir_y==1:
            vector=Geodesic.WGS84.Direct(lat_1,lon_1,(cog_1+np.degrees(np.arctan(dis
            t_lon_1=vector['lon2']
            t_lat_1=vector['lat2']
        elif dir_x==1 and dir_y==2:
            vector=Geodesic.WGS84.Direct(lat_1,lon_1,(cog_1+(360-np.degrees(np.arcta
            t_lon_1=vector['lon2']
            t_lat_1=vector['lat2']
        elif dir_x==2 and dir_y==2:
            vector=Geodesic.WGS84.Direct(lat_1,lon_1,(cog_1+(180+np.degrees(np.arcta
            t_lon_1=vector['lon2']
            t_lat_1=vector['lat2']
        elif dir_x==2 and dir_y==1:
            vector=Geodesic.WGS84.Direct(lat_1,lon_1,(cog_1+(180-np.degrees(np.arcta
            t_lon_1=vector['lon2']
            t_lat_1=vector['lat2']
        return (t_lat_1,t_lon_1)
    #if dimensions are missing
    except:
        return(lat_1,lon_1)
```

A.7 Print timeseries

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri Jun  1 14:34:36 2018

@author: haakon
"""
#################READ THIS############
#vessel type must be written manually
# length of timeseries are set with start and end variables line (36,37)

import matplotlib.pyplot as plt
import matplotlib.dates as mdates
import sqlite3 as lite
import numpy as np
import pandas as pd
from datetime import datetime
from datetime import timedelta
from geographiclib.geodesic import Geodesic
from formulas import rate_of_turn, l_a, True_center

x = int(input("Encounter ID "))
year = int(input("input year"))
conn = lite.connect(f'/home/haakon/Documents/Vestfjorden{year}.db')
c = conn.cursor()
c.execute(f"Select * from sorted_70_{year} where dom_id = {x};")
entries=c.fetchall()
if len(entries)>1:
    a=int(input(f"which one [1:{len(entries)}]"))-1
else:
    a=0
entry=entries[a]
mmsi_1=entry[3]
mmsi_2=entry[4]
time_CPA=datetime.strptime(entry[1], "%Y-%m-%d %H:%M:%S")
start=str(time_CPA-timedelta(minutes=10))
end=str(time_CPA+timedelta(minutes=10))

c.execute(f"select Time, MMSI, SOG, COG, Longitude, Latitude from raw_{year} \
          WHERE MMSI = '{mmsi_1}' and Time > '{start}' and Time< '{end}' order by Time;")
entries_1=c.fetchall()
c.execute(f"select Time, MMSI, SOG, COG, Longitude, Latitude from raw_{year}\
          WHERE MMSI = '{mmsi_2}' and Time > '{start}' and Time< '{end}' order by Time;")
entries_2=c.fetchall()
c.execute(f"select * from reindexed_{year} WHERE (MMSI = '{mmsi_1}' OR\
          MMSI = '{mmsi_2}') and Time > '{start}' and Time< '{end}' order by Time,MMSI");
entries=c.fetchall()
a=[]
b=[]
c=[]
if len(entries)>10:
    for idx in range(len(entries)-1):
        if entries[idx][0] == entries[idx+1][0]:
            cog_1=entries[idx][5]
            if type(cog_1) == str:
                try:
                    int(entries[idx-2][5])
                    cog_1=entries[idx-2][5]
                except:
                    cog_1=0.1
            sog_1=entries[idx][2]
            if type(sog_1) == str:
                try:
                    int(entries[idx-2][2])
```

```
                    sog_1=entries[idx-2][2]
                except:
                    sog_1=0.1
            time_0=datetime.strptime(entries[idx][0], "%Y-%m-%d %H:%M:%S")
            cog_2=entries[idx+1][5]
            if type(cog_2) == str:
                try:
                    int(entries[idx-1][5])
                    cog_2=entries[idx-1][5]
                except:
                    cog_2=0.1
            sog_2=entries[idx+1][2]
            if type(sog_2) == str:
                try:
                    int(entries[idx-2][2])
                    sog_2=entries[idx-2][2]
                except:
                    sog_2=0.1
            t_lat_1=True_center(entries[idx][4],entries[idx][3],entries[idx][8],entries[idx][9],
            t_lon_1=True_center(entries[idx][4],entries[idx][3],entries[idx][8],entries[idx][9],
            t_lat_2=True_center(entries[idx+1][4],entries[idx+1][3],entries[idx+1][8],entries[id
            t_lon_2=True_center(entries[idx+1][4],entries[idx+1][3],entries[idx+1][8],entries[id
            distance=Geodesic.WGS84.Inverse(t_lat_1,t_lon_1,t_lat_2,t_lon_2)['s12']
            #domain length in direction from ship 1 to ship 2 [meter]
            dom_len_1_2=l_a((entries[idx][8]+entries[idx][9]), t_lat_1, t_lon_1, t_lat_2, t_lon_
            #domain length in direction from ship 2 to ship 0 [meter]
            dom_len_2_1=l_a((entries[idx+1][8]+entries[idx+1][9]), t_lat_2, t_lon_2, t_lat_1, t_
#            dom_len_2_1=l_a(28, t_lat_2, t_lon_2, t_lat_1, t_lon_1, cog_2,sog_2)

            #distance center to center [meter]
            distance=Geodesic.WGS84.Inverse(t_lat_1,t_lon_1, t_lat_2,t_lon_2)['s12']
            #negative values indicate domain violation
            a.append((time_0.time(),(distance-dom_len_1_2)))
            b.append((time_0.time(),(distance-dom_len_2_1)))
            c.append((time_0.time(),distance))

fig, ax1 = plt.subplots()
plt.plot(*zip(*a),'b--', linewidth=0.8)
plt.plot(*zip(*b),'r--', linewidth=1)
plt.plot(*zip(*c),'k--', linewidth=0.3)
#plt.legend(('ship_1','ship_2'))
ax1.set_ylabel('Distance[m]', color='k')
ax1.grid(which='major',color='k',axis='y', alpha=0.2)
ax1.set_xlabel('Time',color='k')
ax1.legend(('Fishing vessel 1','Fishing vessel 2','Distance'),loc='right', bbox_to_anchor=(1.37,
#ax1.legend(('Fishing','Ferry'),loc='upper left')
#ax1.legend(('Fishing','Ferry'),loc='upper right', bbox_to_anchor=(1.35, 1.05))
#box = ax1.get_position()
#ax1.set_position([box.x0, box.y0, box.width * 0.9, box.height])
#ax1.legend(bbox_to_anchor=(1.1, 1.05))
_=plt.xticks(rotation=90)

ax2 = ax1.twinx()

plt.plot(*zip(*rate_of_turn(entries_1)),'-b',markersize=3, linewidth=0.6)
plt.plot(*zip(*rate_of_turn(entries_2)),'-r',markersize=1, linewidth=1)
#ax2.legend(('Fishing','Ferry','Distance'),loc='right', bbox_to_anchor=(1.35, 1.05))
plt.xticks(rotation=90)
plt.xlabel('Time',)
ax2.set_ylabel('Est. ROT[deg/min]', color='k')
plt.savefig(f'/home/haakon/rot_{x}_rot.pdf', format='pdf',bbox_inches='tight')
plt.close()
```