Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Marine Technology

Luis Fernando Sanchez Martin

Online Digital Twin and a Decision Support for Safe Maneuvering of R/V Gunnerus

Master's thesis in Marine Technology
Supervisor: Roger Skjetne
Co-supervisor: Mathias Marley
January 2022

NTNU
Norwegian University of
Science and Technology

Luis Fernando Sanchez Martin

Online Digital Twin and a Decision Support for Safe Maneuvering of R/V Gunnerus

Master's thesis in Marine Technology
Supervisor: Roger Skjetne
Co-supervisor: Mathias Marley
January 2022

Norwegian University of Science and Technology
Faculty of Engineering
Department of Marine Technology

NTNU
Norwegian University of
Science and Technology

NTNU Trondheim
Norwegian University of Science and Technology
Department of Marine Technology

MASTER OF TECHNOLOGY THESIS DEFINITION (30 SP)

Name of the candidate:	Sanchez Martin, Luis Fernando
Field of study:	Marine cybernetics
Thesis title (Norwegian):	Online digital tvilling og beslutningsstøtte for sikker manøvrering av F/F Gunnerus
Thesis title (English):	Online digital twin and a decision support for safe maneuvering of R/V Gunnerus

Background

Simulating and verifying autonomous ships require complex simulation environments capable of modeling the vessel dynamics and ship systems, position, heading, and motion sensors, exteroceptive sensors, and external traffic and environmental conditions, and thereafter running these models in real time in a realistic operational environment. Such simulators are essential in the assurance of autonomous vehicles, where CARLA (https://carla.org) is a prime example from the autonomous automotive industry.

In many cases, embedding information related to safe maneuvering onto the 2D electronic chart can be very valuable. These charts give very good overview of the large-scale situation with multiple target ships (TSs) in the vicinity of the own ship (OS). Gemini is, on the other hand, a Unity-based 3D simulation and visualization platform originating from NTNU, aiming to fill the role of testing and certification of autonomous marine vessels.

Such simulators should be connected and adapted to the real operations of the vessel, in order to enable remote monitoring/control and decision support functions. The goal of this project is to establish a digital twin system of R/V Gunnerus (RVG) and extending this with decision support related to safe maneuvering in marine traffic. This includes establishing an online datastream from the real vessel, visualization of this in 2D (chart) and 3D (Gemini), develop a predictive decision support function based on control barrier functions and a simulation model, and visualization of corresponding collision-related parameters with respect to TS from AIS data. This will be the start of development of a digital twin lab for RVG for use in the department's research and education.

Scope of Work

1. Perform a background and literature review to provide information and relevant references on:
   - Unity, Gemini, CARLA, Ocean Systems Lab (OS-Lab), etc.
   - Relevant background on marine autonomous surface ships and marine digital twin definitions, scope, services, incl. earlier master studies and papers on RVG digital twin development.
   - RVG; its use profile and particulars, and its system for data communication to shore.
   - Relevant communication protocols and setups; UDP, MQTT, etc.
   - Automatic Radar Plotting Aid (ARPA) system/functions/parameters for collision avoidance.
   - Relevant alternative collision avoidance methods, such as Collision Avoidance Dynamic Critical Area (CADCA) and Control Barrier Functions (CBF).

   Write a list with abbreviations and definitions of terms and symbols, relevant to the literature study.
2. Prepare the Gemini simulator to read online data from the real RVG and present them in a 3D environment of Trondheim harbor and its vicinity. Implement an updated 3D object drawing to feature the actual RVG with correct dimensions. Implement also TSs appearing in RVG's AIS messages, as well as other relevant information. Implement a web-application that can feature the 3D visualization on an arbitrary computer screen and demonstrate the solution.
3. Prepare a 2D electronic chart (EC) function that features RVG and TSs with relevant odometry data. Implement it as a web application to be shown on an arbitrary screen. Demonstrate the solution.
4. Study and derive Automatic Radar Plotting Aid (ARPA) parameters such as CPA, DCPA, TCPA. Embed historic and predicted (straight line) paths, as well as the ARPA parameters, for RVG and the relevant TSs in the 2D EC app.

5. Develop a collision-avoidance (CA) decision support system (DSS) as a function to be added to the 2D EC app. It should be based on Control Barrier Functions (CBFs), to be specified for each TS and implemented as a guidance model and simulated from present state of RVG and into the future. The resulting predicted path should be plotted, with additional info, as recommended actions to the officer on watch (OOW). The simulation and path prediction should be updated at sufficient rate. The CBF methodology and resulting CSS design should be presented.

Tentatively:

6. As alternative to reading the real RVG datastream, include a function that simulates an RVG operation as part of its digital twin system. This should use the Python-based high-fidelity 4DOF maneuvering model of RVG in the MCSim_python repo. Based on the data generated through the simulation, the same functions as for the real RVG should be shown, incl. potential TSs.

Specifications

Every weekend throughout the project period, the candidate shall send a status email to the supervisor and co-advisors, providing two brief bulleted lists: 1) work done recent week, and 2) work planned to be done next week.

The scope of work may prove to be larger than initially anticipated. By the approval from the supervisor, described topics may be deleted or reduced in extent without consequences with regard to grading.

The candidate shall present personal contribution to the resolution of problems within the scope of work. Theories and conclusions should be based on mathematical derivations and logic reasoning identifying the steps in the deduction.

The report shall be organized in a logical structure to give a clear exposition of background, problem/research statement, design/method, analysis, and results. The text should be brief and to the point, with a clear language. Rigorous mathematical deductions and illustrating figures are preferred over lengthy textual descriptions. The report shall have font size 11 pts., and it is not expected to be longer than 70 A4-pages, 100 B5-pages, from introduction to conclusion, unless otherwise agreed. It shall be written in English (preferably US) and contain the elements: Title page, project definition, preface (incl. description of help, resources, and internal and external factors that have affected the project process), acknowledgement, abstract, list of symbols and acronyms, table of contents, introduction (project background/motivation, objectives, scope and delimitations, and contributions), technical background and literature review, problem formulation or research question(s), method/design/development, results and analysis, conclusions with recommendations for further work, references, and optional appendices. Figures, tables, and equations shall be numerated. The contribution of the candidate shall be clearly and explicitly described, and material taken from other sources shall be clearly identified. Chapters/sections written together with other students shall be explicitly stated at the start of the chapter/section. Work from other sources shall be properly acknowledged using quotations and a Harvard citation style (e.g. natbib Latex package). The work is expected to be conducted in an honest and ethical manner, without any sort of plagiarism and misconduct, which is taken very seriously by the university and will result in consequences. NTNU can use the results freely in research and teaching by proper referencing, unless otherwise agreed.

The thesis shall be submitted with an electronic copy to the main supervisor and department according to NTNU administrative procedures. The final revised version of this thesis definition shall be included after the title page. Computer code, pictures, videos, data, etc., shall be included electronically with the report.

Start date: 15 January, 2023 **Due date:** As specified by the administration.

Supervisor: Roger Skjetne
Co-advisor(s): Mathias Marley

Signatures: Roger Skjetne Digitally signed by rskjetne Date: 2023.02.15 13:44:57 +01'00'

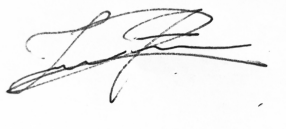
Preface

This paper is a master thesis and a part of the study programme Marine Technology at the Norwegian University of Science and Technology (NTNU). The work was carried out during the spring semester of 2023 for the course TMR4930 Marine Technology, master thesis, at the Department of Marine Systems Design/Machinery. The workload and the weighting are 30 ECTS and the supervisor for this thesis is Roger Sketne.

The focus of this thesis is on the development of an online digital twin for the Research Vessel Gunnerus, and it follows an initiative at NTNU for the development of a digital twin for this vessel. The work presented in this thesis is my contribution to this initiative. This work will serve as a vessel monitoring tool, and decision support system for safe navigation based on real-time data. This opens the possibility of more implementations based on real-time data to be developed down the line that will expand the capabilities of this online digital twin.

Several technologies were involved in the development of the framework for the online digital twin, this spans several programming languages, software frameworks and state-of-the-art implementations. Some of which I was familiar with, and most of which I had to learn. Overall, this has been a very rewarding experience and I hope that this contribution will be a meaningful step forward towards the development of a comprehensive digital twin for Research Vessel Gunnerus.

Trondheim, 11th June 2023



Luis Fernando Sanchez Martin

Acknowledgement

I would like to thank the Erasmus Mundus Joint Masters Degree in Marine and Maritime Intelligent Robotics for giving me the opportunity of partaking in this academic experience. I would also like to thank Prof. Roger Skjetne for providing guidance throughout the course of this project.

To my friends, family, and educators. Thank you all for your support.

Abstract

This thesis will present the development of an online digital twin for NTNU's Research Vessel Gunnerus. Maritime digital twins are becoming a more and more prevalent technology used for the development, maintenance, and evolution of maritime vessels. A digital twin is at its core a virtual representation of a physical asset, it can simulate the interaction of the diverse systems involved in the operation of its real-life counterpart and combine them with available sensor data. As such, a digital twin can accompany a vessel from its early stages of development, going then through the integration of additional systems, testing of new implementations, monitoring of the real vessel, and its maintenance.

The focus of this work is on displaying real-time information coming from Gunnerus for the purposes of vessel monitoring and decision support for safe navigation. To this effect, a backend implementation has been developed for interfacing with the real-time data stream coming from Gunnerus. This backend will then be used to provide processed data to the applications in charge of vessel monitoring and decision support for safe navigation. These applications will be implemented as a 3D visualization based on the Autoferry Gemini simulator, and a 2D electronic chart web application, both conveying information related to Gunnerus and the surrounding vessels. Furthermore, the 2D electronic chart will display information regarding the decision support for safe navigation. This decision support system will use real-time data in order to display parameters from automatic radar plotting aids related to nearby vessels. Additionally, a safe trajectory based on control barrier functions will be displayed in the same chart as a suggestion for safe navigation for the officer on watch.

The capabilities of the resulting framework for an online digital twin of Gunnerus will be demonstrated with examples based on real data collected through its development.

Acronyms

AIS Automatic identification system. 7, 9

ARPA Automatic Radar Plotting Aids. 1, 3, 5, 18

CARLA Car Learning to Act. 17

CBF Control Barrier Functions. 1, 3, 5, 20

COG Center of gravity. 7

COLAV Collision Avoidance. 18, 27

COLREG Convention on the International Regulations for Preventing Collisions at Sea. 15

CRP Coordinate reference point. 6

DM Data Model. 33, 34, 43

DP Dynamic positioning. 6, 17

DSS Decision Support System. 1, 5, 27

DT Digital Twin. 1, 5, 10

EC Electronic Chart. 3, 26

ENU East North Up. 39

FMI Functional Mockup Interface. 13

HOCBF Higher Order Control Barrier Functions. 20

IL Invisible light. 15

IMO International Maritime Organization. 14

JSON Java Script Object Notation. 38

LCG Longitudinal center of gravity. 7

MASS Maritime Autonomous Surface Ship. 14

ML Machine learning. 17

MMSI Maritime Mobile Service Identity. 38, 49

MRU Motion reference unit. 6, 7

NMEA National Marine Electronics Association. xv, 7

OOW Officer On Watch. 27, 32

OSP Open Simulation Platform. 13

PMSM Permanent Magnet Synchronous Motors. 12

ROV Remote operated vehicle. 6

RVG Research Vessel Gunnerus. 5

TCP Transmission Control Protocol. 24

TTEM Time To Evasive Maneuver. 32

UDP User Datagram Protocol. 24

UE4 Unreal Engine 4. 17

UI User Interface. 26, 47

VCG Vertical center fo gravity. 7

VHF Very High Frequency. 9

VL Visible light. 15

VPN Virtual Private Network. 24

Glossary

csv file a text delimited file format that uses commas and new line characters to separate and organize different data entries into fields. Similar to the way data is displayed in Microsoft Excel. 34

WebSocket a computer communications protocol, that provides full-duplex communication channels over a single TCP connection. 37

Contents

Figures

Tables

Chapter 1

Introduction

The goal of this thesis is to develop an online Digital Twin (DT) and Decision Support System (DSS) for safe maneuvering of RVG, based on Automatic Radar Plotting Aids (ARPA) parameters and Control Barrier Functions (CBF). The structure of the report is as follows.

- Technical Background: An overview of the technical background required for the development of the project presented in this thesis.
- Problem Formulation: This chapter will present the methodology through which the objectives of this thesis will be achieved.
- Real-Time Data Interfacing and Processing: This chapter will present the back-end implementation for accessing data from Research Vessel Gunnerus, processing it, and relaying it to other applications.
- Vessel Monitoring and Data Visualization: The implementation for the 2D and 3D visualization for monitoring Research Vessel Gunnerus will be presented in this chapter. This will include the Display for the DSS for Safe Navigation
- Demonstration of RVG Online Digital Twin: Results on the implementation of the different components will be presented and discussed in this chapter.
- Conclusions: Conclusions from the project will be drawn and recommendations for future work will be pointed out.

1.1 Project Background and Motivation

This project is part of an ongoing initiative by NTNU professors, students, and industry partners to develop a digital twin for RVG. This initiative serves multiple purposes.

First, as an educational tool and real-world application of the knowledge acquired through the courses of NTNU which can be later harnessed in a technologically driven work or research environment.

Second, academia and particularly NTNU will benefit from the contributions to this project; as each collaboration further improves the capabilities of the RVG

digital twin.

Third, research partners will benefit from the output of the initiative as multiple facets of the digital twin technology are realized and put to practical use and can be then implemented on different applications or platforms.

Finally, as a student myself this thesis proves to be a valuable learning tool as I employ all the tools and knowledge at my disposal in order to develop the components necessary for the realization of this project.

1.2 Objectives

As outlined in the preliminary thesis description the main objective of this project is to enable the online capabilities of a digital twin system of Research Vessel Gunnerus and extend it with decision support for safe maneuvering in marine traffic. This includes:

- Establishing a real-time, online data stream from the real vessel.
- Visualization of this data in a 2D electronic chart and 3D simulator.
- Developing a predictive decision support function based on control barrier functions and a simulation model.
- Visualization of corresponding collision-related parameters with respect to incoming AIS data

1.3 Scope and Delimitation

In order to achieve the objectives of this project, three main components will come into play, each with its own related challenges and technologies.

Establishing and making available the real-time data stream coming from RVG will require a back-end that will receive the incoming data from the sensors as well as marine traffic and will process it in order to make it available to other components. Additionally, this back-end must be capable of logging and replaying the received data. In this case, the back-end will be created using Python, mostly due to ease of use and availability of resources and libraries that can aid in the development of the project.

The 2D electronic chart shall utilize the data processed by the back-end and clearly display information that will aid with decision support for RVG operations. This includes information related to marine traffic in the area surrounding RVG, as well as a planned path for RVG to follow in order to avoid other vessels within predefined safety parameters. This chart will be developed as a web application using the React.js framework as it allows for the quick development of user interfaces for web applications.

Finally, the 3D simulator Gemini will be modified in order to depict the situation of RVG as well as its environment in real-time; this comprises an accurate representation of the position and orientation of RVG as well as nearby vessels

at sea. Additionally, the output of this visualization will be made available as an online stream which will accompany the 2D chart web application.

It is worth noting that since this project is mainly concerned with the online part of a Digital Twin, simulation and co-simulation of RVG is outside of the scope of this project, i.e.; it won't be possible to render a Dynamic simulation of RVG in a 3D environment and obtain simulated online data from it. This project can only handle historic and real-time data incoming from RVG.

1.4 Contributions

The main contribution of this project is to provide a starting point for collecting and utilizing real-time data from the vessel and provide a comprehensive access point for other implementations to further expand on the digital-twin system at NTNU, starting with the decision support for obstacle avoidance and navigation for RVG. In more concrete terms, the following software solutions will be provided as an output of this project:

- A backend implementation in Python capable of interfacing with the Real-time data stream incoming from RVG, storing it, and relaying it to client applications.
- A 3D visualization of RVG and surrounding vessels in real-time based on Autoferry Gemini simulator.
- A 2D Electronic Chart (EC) with the geographical situation of RVG and the surrounding vessels based on the React Framework. Additionally, this 2D EC will display information for the DSS:
    - Automatic Radar Plotting Aids (ARPA) parameters of pertinent vessels will be displayed in the chart.
    - Visualization of a suggested path based on Control Barrier Functions (CBF) will be displayed in the chart.
    - Telemetry data will also be accessible in the form of plotted time series.
- The implementation will be wrapped in a web application that will provide access to the aforementioned features plus user-selectable settings.
- The web application will be mounted on an Intel Nuc computer and set up for a demonstration on a TV screen at NTNU's Moholt Campus.

Chapter 2

Technical Background

This section will provide the technical background of the components and technologies involved in achieving the objectives of the project. This includes a description of the research vessel involved in the development of the project. An overview of Marine digital twins, their development and applications, and previous work related to the development of a Digital Twin (DT) for Research Vessel Gunnerus (RVG) at NTNU.

Autonomy in the context of marine vessels will also be included in this section to emphasize one of the directions the DT for RVG initiative could follow in future iterations, as the current implementation of this project proposes a Decision Support System (DSS) seeking to aid in the navigation of the vessel.

Vehicle simulators play a key role in this project, therefore examples of vehicle simulators previously used for research will be explored in this section. These simulators include Autoferry Gemini, a simulator used for the development of autonomous vessels at NTNU.

Finally, an introduction to the concepts of Automatic Radar Plotting Aids (ARPA) and Control Barrier Functions (CBF) will be presented along with the general equations involved in the computation of their related parameters.

Figure 2.1: Port view of R/V Gunnerus (Polarkonsult (n.d.)).

2.1 Research Vessel Gunnerus

RVG (Figure 2.1) is an NTNU vessel in operation since 2006. It is fitted with the equipment necessary to conduct a variety of research activities within different fields of maritime sciences; these include an ROV, a crane for its deployment, and a variety of Dynamic Positioning (DP) sensors that allow for its positioning and deployment, these are listed in Table 2.1.

Table 2.1: Gunnerus positioning sensors

Dynamic Positioning system	Kongsberg SDP-11 / cPos
DP - Reference systems	GPS, Kongsberg Seatex DPS 232, HPR, Kongsberg transponders, Kongsberg Seatex RADius
Heading, Attitude and Positioning Sensor	Kongsberg Seapath 300
Acoustic positioning system	Kongsberg HiPAP 500
Motion reference unit (MRU)	Kongsberg Seatex
Compass, magnet	Nautisk service NS 150-A
Compass, gyro	Simrad GC80 / 85
Differential positioning sensor	Kongsberg DPS 232
GPS	Furuno Navigator GP-90
Radar	Furuno FAR 28x7 /FAR 21x7
Echo Sounder	Furuno FCV-1200L. 38, 50, 200 kHz – 2000m
Echo sounder, multibeam	Kongsberg EM 3002s

RVG is fitted with diesel-electric propulsion engines (see Table 2.2) capable of reaching a top speed of 12.6 kn with a cruising speed of 9.4 kn.

Table 2.2: Gunnerus propulsion system

Main electric propulsion	1000 kW (Siemens 2 $\times 500kW$)
Main generators	3 $\times Nogva - Scania 450kW$
Bow tunnel thruster	1 $\times Brunvoll 200kW$

Furthermore, this vessel has the navigation assistance and communications equipment required to conduct typical operations, see Table 2.3.

A reference for the position of some of the sensors involved in the data collection will be provided in this section. The body-fixed reference frame shall originate from the coordinate reference point (CRP) in Appendix A.2. With axis X pointing towards the bow, Y axis pointing towards the starboard, and Z pointing upwards, see Figure 2.2. It is worth noting that RVG has been lengthened by 5 meters in the x direction since the creation of the drawings Appendix A.2. The z-axis is inverted in this frame to match the reference provided in the schematics for RVG.

Table 2.3: RVG navigational assistance and communications equipment.

Chartplotter 1	Telchart (AIS)
Chartplotter 2	Olex (Installed AIS, HT, SB, ITI, MBES)
Chartplotter 3	Olex LT (Office version)
AIS	Furuno FA-150
Navtex	JMC NT900
GMDSS console	
VHF fixed radio	Sailor
VHF handheld radios	Jotron
UHF handheld radios	Icom
Satellite phone	Sailor
Internet in sea ICE	Telenor mobilt bredbånd
Internet at pier Trondheim	Wireless broadband NTNU

The Center of Gravity (COG) for the ship, previous to the extension, was located at the intersection between the vertical center of gravity (VCG), 3.354 meters above the lowest point of the underside of the keel; and the longitudinal center of gravity (LCG), 13.162 meters from the center of the rudderstock. Most of the Motion Reference Unit (MRU) sensors are placed in a central position in the ship, as is customary under rigid body assumptions (Heyn et al. (2017)).

RVG has the necessary equipment for gathering data from itself and its surroundings and then relaying it ashore for further study and analysis. This is necessary in order to develop a number of maritime applications.

2.2 RVG Communications and Equipment

Establishing a real-time, online data stream from the RVG is the first objective of this project. this section will explore the communications equipment, protocols, and data involved in achieving this objective.

As stated in previous sections RVG has the capability of gathering information from its sensors and then relaying it to an onshore server. This project is mainly concerned with the NMEA and Automatic Identification System (AIS) messages being transmitted by the vessel. Furthermore, it is possible to obtain MQTT messages from the onboard equipment, as such, this system will also be explored in this section.

2.2.1 NMEA Data and Telegrams

The NMEA Interface Standard (National Marine Electronics Association (2022)) is a communications protocol for marine electronics: GPS, echo sounders, compass,

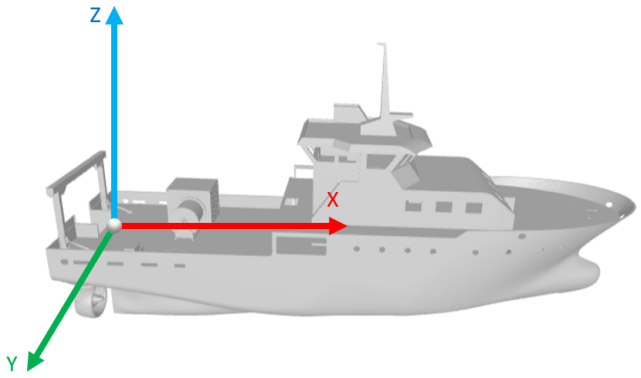


Figure 2.2: Visualization of body fixed frame for RVG. Axis X in red, Y axis in green, and Z axis in blue. The 3D model for RVG displayed here originates from Rølvåg and Stranden (2022).

Table 2.4: Available locations for Gunnerus sensors, all coordinates are in meters.

Sensor	$X(m)$	$Y(m)$	$Z(m)$
Seapath Fwd.	19.250	-0.61	12.228
Seapath Aft	16.755	0.574	12.275
DPS 232	17.811	-0.611	15.599
MRU 5	20.459	-0.578	1.450
MRU 5, 2	18.764	0.443	3.856
MRU 5, 3	18.762	0.583	3.630
MRU 5+	18.037	0.805	3.028
HiPAP	26.472	-0.005	-6.637
RADius	13.077	-1.547	8.127
EM 3002s	23.635	0.951	-4.128

etc. That defines electrical signal requirements, data transmission protocol and time, and specific sentence formats for a 4800-baud serial data bus. This standard is geared towards communications between a single talker and several listeners. The sentences emitted by the talker can be parsed as ASCII characters.

NMEA is the communication standard used to communicate the incoming sensor readings from the onboard sensors to the systems or components that require it. Sensor data can be communicated in a single message, spread across different ones, or combined into proprietary sentences.

The telegrams currently being broadcasted by RVG can be found in Appendix A.1. The message definitions therein are based on Kongsberg (Kongsberg Maritime (2013)) and the NMEA manual (National Marine Electronics Association (2007)).

The NMEA telegrams relayed by RVG are captured and made available by the equipment described in Table 2.1 from Chapter 2. A backend will process and then relay this data for other programs to use i.e.; 2D Electronic Chart and 3D

visualization.

2.2.2 AIS Data and Telegrams

The Automatic Identification System (AIS) is a tool mainly intended for maritime safety, this implies vessel collision avoidance, for port authorities to obtain information about a ship and its cargo, and traffic management (Tetreault (2005)).

AIS equipment continuously and autonomously transmits information about the vessel, including its identity, position, course, and speed. This is transmitted via Very High Frequency (VHF) maritime mobile band. The technical specifications for the underlying communications protocols and recommended equipment are described in International Telecommunications Union (2014). The AIS messages related to the vessel's dynamic information are obtained by combining incoming data from the onboard sensors. The rate at which this data is transmitted is dictated by the vessel's speed and heading change, i.e.; around 2 to 10 seconds when moving and every 3 minutes when anchored.

The system entails some caveats as it is susceptible to errors in the transmission and reception of the information (Harati-Mokhtari et al. (2007)). Nevertheless, it has been deemed sufficiently reliable for the purposes of this project. In this context, the AIS data will be mainly used to determine the position of other vessels relative to RVG, visualize them in 2D and 3D, and make the necessary computations for generating the data for the DSS. Furthermore, since this data contains the information required for identifying the target vessel, it can also be used to display its static information, i.e.; its dimensions.

The AIS data being relayed ashore and made available for this project is provided by the AIS equipment described in Table 2.3.

2.2.3 The MQTT Protocol

MQTT is a protocol normally running atop TCP and it was originally designed for sending data accurately under long network delay and low bandwidth network conditions (Soni and Makwana (2017)).

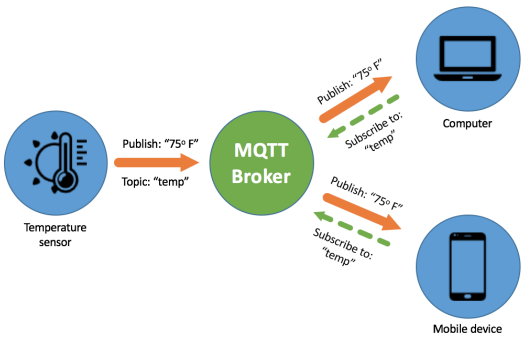


Figure 2.3: Client broker example in MQTT (Soni and Makwana (2017))

This protocol is based on a two parts architecture consisting of a client and a broker. Clients can publish messages into, and subscribe to brokers; while brokers accept messages from clients and publish them to the others subscribed, see Figure 2.3.

MQTT comes with a few drawbacks:

- Due to the lack of message expiry, there exists the possibility of brokers becoming overloaded with messages; degrading the overall performance.
- Security mostly depends on the broker being used and it is liable to negatively impact the performance.
- Ordering and re-sending messages upon losses can be challenging, the former more so than the latter.
- MQTT does not support message prioritizing.

RVG counts on MQTT communication for relaying information related to the onboard equipment. Though it is out of the scope of this project, this source of data could be leveraged down the line for use in the RVG DT.

2.3 Marine Digital Twin

A Digital Twin (DT) is a dynamic virtual representation of a physical object. They combine diverse models for the system with available sensor data to achieve a better understanding and improve upon its real-life counterpart. Simulators and data acquisition tools are therefore an integral part of developing a DT.

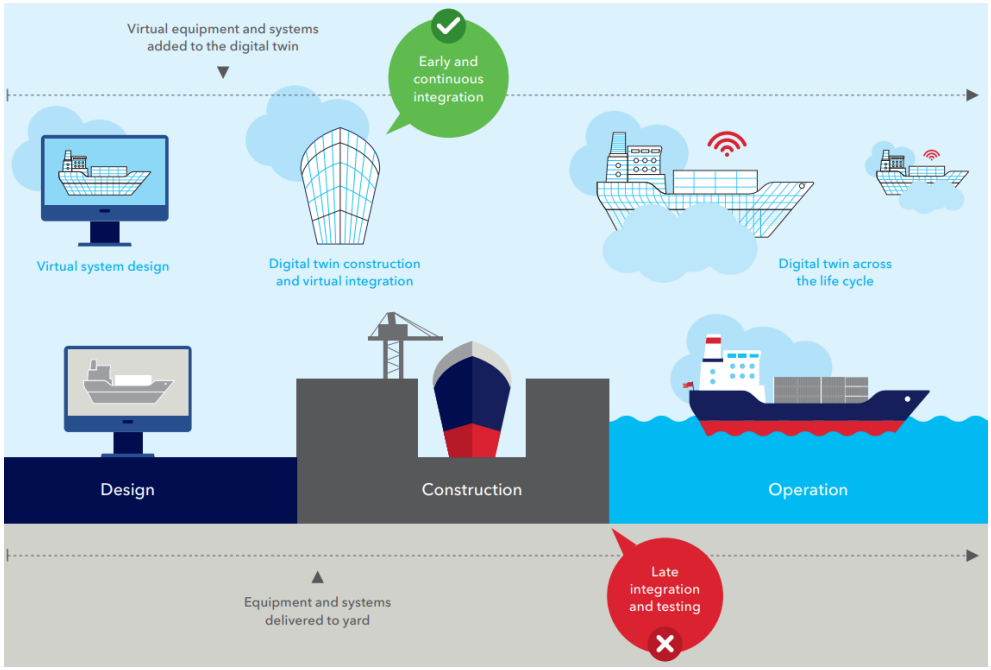


Figure 2.4: Virtualization throughout the life cycle of a ship (DNV GLAS (2020))

Digital twins can be used for a variety of practical purposes, such as development, integration, testing, monitoring, and maintenance. This entails the combination of different models for different systems for a vessel, for example, propulsion, navigation, electronics, communication, etc.

A DT can also serve a critical role throughout the entire life cycle of a ship, from its conception in the design stages; construction, and monitoring throughout the lifespan of the real counterpart; and as a platform for testing applications before implementing them in the physical vessel. See Figure 2.4.

It is worth noting that since this technology is in its early stages, DTs exist as a combination of several software, tools, and requirements rather than a single centralized platform. Nevertheless, DNV has released guidelines on recommended practices for the qualification and assurance of DTs (DNV (2021)).

2.3.1 Digital Twins as a Digital Service

Erikstad (2019) presents an approach to designing digital twins as a digital service for ships. Digital services have been identified as a key enabler for operational efficiency, safety, and reducing environmental impact. This work suggests adopting a sensor-to-service approach that seeks to leverage the information related to the ship and its surroundings; information that is collected by on-board and off-board sensors. It is pointed out that great volumes of data have been gathered over the years by the actors involved in providing ship services, however, this data has not yet been given proper usage, often resulting in digital suites being developed without exploiting that data. The sensor-to-service approach seeks to alleviate this situation, as it allows us to weigh resources versus implementation to obtain an optimal result. It is also suggested that an engineering design perspective is necessary for the development of digital twins as a service, in other words, it should follow the same process as, for example, designing a ship. The system-based design from Erikstad and Levander (2012) is used as a template and can be summarized in the following steps:

1. Customer Requirements: mission statement.
2. Functional Requirements.
3. Form: parametric exploration.
4. Engineering Synthesis.
5. Evaluation of Design.

These steps can also be visualized in Figure 2.5. Insight on decision-making for different levels: strategic, tactical, and operational are also included and put into perspective with different time-frames: hindsight, insight, and foresight. This has the aim of identifying and delimiting the scope of digital services. The author suggests all of these as a guide for developing DTs as a digital service, though hypothetical, it should serve as a framework for their development and implementation.

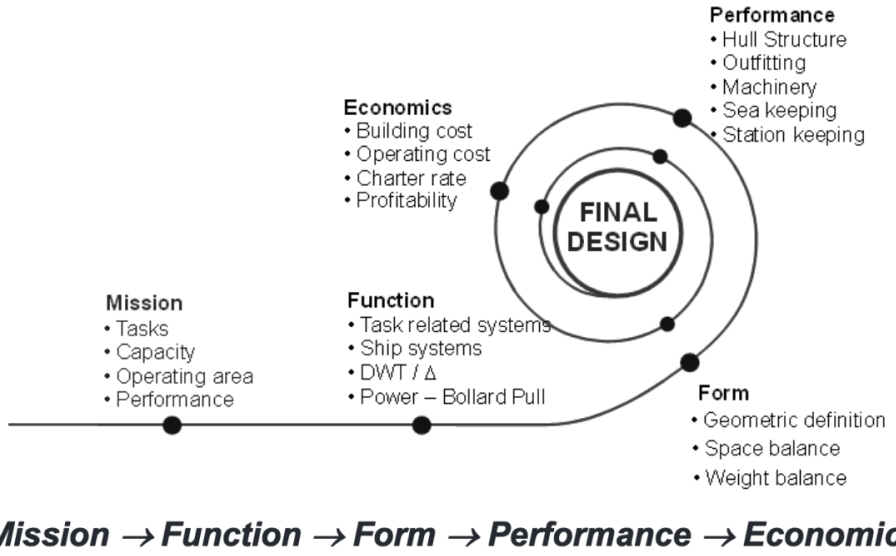


Figure 2.5: System Based Design Process, (Erikstad and Levander (2012))

2.3.2 Digital Twins for RVG

There has been a continuous push at NTNU to develop a DT of RVG spanning several aspects of the development of such a system. In Bjørum (2019), techniques for monitoring the condition of the electrical propulsion system of RVG are presented. The thermal and electrical characteristics of RVG's Permanent Magnet Azimuth thrusters are modeled with the aim of being coupled with available information from the vessel, i.e.; historical data from azimuth thrusters. Part of this work also included the verification of a pre-existent 3D and physics model of RVG. The key factors for condition monitoring identified in the work are as follows:

- Sensors: to convert physical phenomena into electrical impulses.
- Data acquisition: amplification and pre-processing of sensor data.
- Fault detection; comparing the sensor data with predictive models for early failure detection.
- Diagnosis: interpreting abnormalities in the incoming signals, pinpointing their location, and providing advice for maintenance.

Power losses and thermal profiles for different operational scenarios of Permanent Magnet Synchronous Motors (PMSM) are identified. Fault detection models based on Gaussian processes are also simulated. The data obtained through these methods is ultimately used to estimate the lifetime and damage of the simulated system.

Many complex systems that allow for navigating complex dynamic environments are at play to ensure the operation of more advanced maritime vessels. Designing such systems is an inherently complex task, therefore, scalable simulation technologies should take the lead in this regard. The challenge then is to

leverage these systems and couple them in a framework that ensures that all of these systems can be managed throughout all of the stages of a vessel's life cycle; a maritime digital twin.

Maritime digital twin models should be constructed prior to or in parallel with the actual building of a vessel. In so doing they can help with the conception of the vessel from its very early stages, as well as throughout the vessel's operation for further refinement, since they can leverage data mining as an aid for improvement upon subsequent iterations.

Some of the advantages highlighted in the work are the following:

- Reduced development time and enhanced production efficiency.
- Improved operational flexibility and reduced costs.
- Improved the system performance and health conditions.
- Improved quality and efficiency of maritime products.
- Facilitate operation approval and certification processes.

Digital twins not only resemble their physical counterparts but also the complex interactions between their subsystems, we must therefore be able to exploit all of the available information in order to construct and maintain a viable DT. Opens Simulation Platform (OSP) and Functional Mockup Interface (FMI) are tools that enable data sharing and model integration.

In Zhang et al. (2022), RVG was used as a test platform for co-simulation through FMI and OSP. This work posits a system where data is collected from RVG's onboard sensors, pre-processed and relayed to an onshore operational center, processed through simulation, and then returned as useful information for the vessel's operation. Processing of the incoming data is enabled by making use of a variety of models that simulate RVG and its subsystems, including hulls, thrusters, power plants, deck machinery, and navigational controllers. External factors such as currents and wind are also simulated and factored in.

Furthermore, modules can be created for onboard support of the vessel, actions can be simulated before occurring in the real world, ship motion predictor, performance optimization, maneuvering, and collision avoidance are such examples. Gathering this information could be useful as a means for enabling the remote operation of RVG from the onshore center as well as an onboard support tool for the crew.

Alvsaker (2020) follows previous initiatives from NTNU for the creation of a DT for RVG, this work highlights the academic use of DT as a learning tool, as it outlines the importance of digital twins in education, its relevance, and application in areas related to the academic development of the students. The work touches on many of the same points presented in previous works, such as digital twin as a digital service, and the life cycle of a digital twin. It explores a software and system realization solution, as it proposes an infrastructure for the RVG DT, where aspects such as data storage, standardization, security, and availability of the collected resources are expanded on.

It also outlines the need for preprocessing and filtering data, while develop-

ing the implementation of an anomaly detection framework based on Machine Learning as a case study. Additionally, it features detailed documentation on the backend and web application.

2.4 Autonomy Levels for Marine Vessels

By using a digital twin we can observe the performance of the system, not only on a component level; but, by analyzing the different systems interacting, as a whole. The holistic approach makes digital twins an ideal platform for developing vessels with autonomous capabilities.

Autonomy in vehicles is better represented as a goal rather than a quality; autonomous systems are those capable of performing complex tasks under a considerable level of uncertainties within themselves as well as the environment with little human intervention. They should be highly dependable and capable of learning in order to adapt and improve their functionality. The International Maritime Organization (IMO) is working towards setting a regulatory regime for Maritime Autonomous Surface Ships (MASS). Currently, the IMO has defined four degrees of autonomy for MASS (IMO (2018a)):

*"**Degree one**: Ship with automated processes and decision support: Seafarers are on board to operate and control shipboard systems and functions. Some operations may be automated and at times be unsupervised but with seafarers onboard ready to take control."*

*"**Degree two**: Remotely controlled ship with seafarers on board: The ship is controlled and operated from another location. Seafarers are available onboard to take control and to operate the shipboard systems and functions."*

*"**Degree three**: Remotely controlled ship without seafarers on board: The ship is controlled and operated from another location. There are no seafarers onboard."*

*"**Degree four**: Fully autonomous ship: The operating system of the ship is able to make decisions and determine actions by itself."*

As the human operator is increasingly taken out of the loop, these degrees of autonomy leave little ambiguity for what comprises autonomy and separate it into different levels according to the frame of marine operation. However it is worth noting that several other frames exist for defining levels of autonomy across different fields; in any case, the typical operational setup of an autonomous ship can be seen in Figure 2.6 and broken down into the following components (Smogeli (2022)):

1. **Perception sensors and object detection**: These modules are in charge of directly obtaining data from the surroundings i.e., readings from the available sensors, and making the data available for the system. A certain degree of data processing is present at this level in the form of individual object detection by some of the sensors.
2. **Situational awareness**: This module is in charge of observing and understanding what is happening in the area of operation. This is achieved by

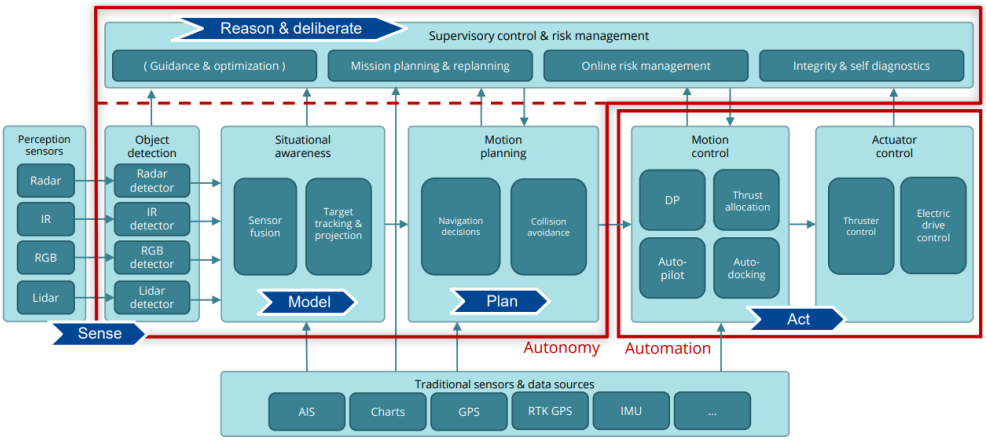


Figure 2.6: Block diagram for components involved in MASS (Smogeli (2022))

processing and synchronizing the data incoming from a variety of **perception sensors**; radar, lidar, IL and VL cameras, etc. by means of machine vision and clustering techniques. The incoming data stream can be fused in order to obtain a "*full picture*" of the world and make predictions.

3. **Motion planning and control**: This component takes the output of the Situational awareness component in order to make navigational decisions and perform collision avoidance based on the planned route. The **control and actuation** of the vessel follows the "*decisions*" as an input to steer the vessel accordingly.
4. **Supervisory control and risk management**: This component oversees the rest of the components much like a human operator does. Monitoring the vessel and its surrounding via the information provided by the other modules in order to asses risks and make strategic decisions; such as re-planning a route, canceling a mission, slowing the vessel's speed due to external conditions, etc.

Taking the aforementioned components into consideration, for an autonomous system to be adopted in the first place, enough trust must exist among the involved parties. Thus, the system in question must be backed with evidence that its functionality will comply with the current standards and regulations. One way to back such claims is by means of simulation.

2.4.1 COLREGs

Convention on the International Regulations for Preventing Collisions at Sea (COLREG) are a set of rules developed to prevent collisions between two or more vessels at sea and all waters connected to high seas and navigable by seagoing vessels. From Thyri and Breivik (2022) the rules identified as necessary for the operation of an autonomous vessel are the following (IMO (2018b)):

- **Rule 8** *Any action to avoid collision shall be taken in accordance with the Rules of this Part and shall, if the circumstances of the case admit, be positive, made in ample time and with due regard to the observance of good seamanship.*
- **Rule 13** *Any vessel overtaking any other shall keep out of the way of the vessel being overtaken.*
- **Rule 14** *When two power-driven vessels are meeting on reciprocal or nearly reciprocal courses so as to involve risk of collision each shall alter her course to starboard so that each shall pass on the port side of the other.*
- **Rule 15** *When two power-driven vessels are crossing so as to involve risk of collision, the vessel which has the other on her own starboard side shall keep out of the way and shall, if the circumstances of the case admit, avoid crossing ahead of the other vessel.*
- **Rule 16** *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.*
- **Rule 17** *Where one of two vessels is to keep out of the way the other shall keep her course and speed. The latter vessel may, however, take action to avoid collision by her maneuver alone, as soon as it becomes apparent to her that the vessel required to keep out of the way is not taking appropriate action in compliance with these Rules.*

2.5 Vehicle Simulators

Carrying out tests in the real world with vehicles may not always prove practical, be it due to the test itself potentially posing a threat to the user or a third party; or simply due to the logistics involved in setting up the environment and required components for the test itself. The specific requirements for these tests may greatly vary according to the implementation. There is always a necessity for a safe or controlled environment in which tests can be performed; thus, virtual environments enter the scene.

Simulating and verifying autonomous ships require complex simulation environments capable of modeling the vessel dynamics and ship systems, position, heading, and motion sensors, exteroceptive sensors, and external traffic and environmental conditions, and thereafter running these models in real-time in a realistic operational environment. Such simulators are essential in the assurance of autonomous vehicles

Game engines are software frameworks that provide relevant libraries primarily designed for the development of video games. As such, they offer an environment with readily available tools for rendering scenes and physics for developers to work with, sometimes even without the knowledge of scripting. Game Engines such as Unity and Unreal Engine 4 (Šmıd (2017)) are therefore useful frameworks for developing virtual environments geared towards research implementations since they are able to generate digital stand-ins or models of real vehicles with very similar properties to their real-world counterparts. However, it is worth noting that the implementation of physical phenomena built into these game en-

gines is normally very simplistic and rarely good enough for scientific purposes; hence, more high-fidelity simulations should be customized into the engine.

CARLA (Dosovitskiy et al. (2017)) is a prime example for the automotive industry and presents an open-source simulator based in Unreal Engine 4 (UE4) for autonomous driving research created to support the development, training, and validation of autonomous urban driving systems. The platform supports flexible specifications of a variety of sensors and environmental conditions. It primarily aims to simulate a self-driving vehicle tasked with navigating urban environments in the presence of other vehicles and pedestrians. This particular environment generates data for training and validating a variety of ML algorithms under different traffic and weather conditions Figure 2.7, and subsequently compare their performance.

Figure 2.7: Simulation of different weather/visibility conditions in CARLA (Dosovitskiy et al. (2017))

Autoferry Gemini (Vasstein et al. (2020)) is an open-source Unity-based platform originating from NTNU, aiming to fill the same role for autonomous marine vessels. It is used to simulate sensors in real-time by leveraging the engine's render pipeline to model lidar, radar, visible-light, and infrared camera sensors simultaneously. In turn, the results from these simulated sensors are of interest for carrying out marine research related to Dynamic Positioning (DP).

These environments can provide a solution for scenarios that would prove too complex for a mathematical simulation. Furthermore, if the desired conditions are correctly set up, virtual environments themselves can be used to generate datasets for Machine Learning (ML) algorithms. In addition, such simulators should be connected and adapted to the real operations of the vessel, in order to enable online monitoring of parameters and learning from the real operations.

2.6 Automatic Radar Plotting Aids

Environmental awareness of a vessel's situation is crucial for navigational purposes, as it allows for making decisions regarding the course the vessel should follow in order to avoid collisions. AIS and radar are one of many technologies

that enhance environmental awareness for the vessel (Lin and Huang (2006)) as they provide information about surrounding traffic as well as their movements. Automatic Radar Plotting Aids (ARPA) are a set of computations that allows for a graphical representation of the parameters required for aiding in collision avoidance (COLAV) during the vessel's navigation. For the purposes of this project, we shall focus on the parameters as presented in Lenart (2017) as well as their related calculations:

- Closest Point of approach (CPA): the position relative to own vessel's current position, located on its own course at which a target vessel will be closest during transit. This assumes that our own vessel and the target vessel maintain constant speed and course.
- Time to Closest Point of approach (T2CPA): time until CPA is reached.
- Safety Radius (R): defines a safe perimeter around the own vessel, with the minimum allowable safe distance between the own vessel and a target vessel during transit. It is the same for every extraneous vessel.
- Distance to Safety Radius (D2R): The distance from own vessel's current position to a point on its trajectory at which it will cross the safety of an extraneous vessel while in transit.
- Time to Safety Radius (T2R): the time it will take own vessel to travel D2R at its current speed. In other words, the time before it will intersect a safety perimeter during transit.

The following calculations work under the assumption that both, our vessel as well as the extraneous vessel are moving with a constant velocity and course. We are also assuming that the relative position (X, Y) and true velocity over ground (V_{tx}, V_{ty}) of the extraneous vessel are known. as well as our own velocity over ground (V_x, V_y) with the position of our vessel located at the origin (O_x, O_y). Thus we obtain the following parameters:
The velocity components of the target vessel relative to our own are defined as:

$$V_{rx} := V_{tx} - V_x \quad , \quad V_{ry} := V_{ty} - V_y \tag{2.1}$$

With the magnitude of the resulting relative velocity vector V_r being

$$V_r = \sqrt{V_{rx}^2 + V_{ry}^2} \tag{2.2}$$

And the corresponding equations of relative motion for a target vessel at a given time t:

$$X(t) = X + V_{rx}\, t \quad , \quad Y(t) = Y + V_{ry}\, t \tag{2.3}$$

The closest point of approach will be defined as the Cartesian point where our own vessel will be the closest to a target vessel, provided both vessels maintain a constant speed vector. If D(t) is the distance to an object at time t, then

$$D(t) = \sqrt{X^2(t) + Y^2(t)} = \sqrt{L^2 + V_r^2\, t^2 + 2(XV_{rx} + YV_{ry})t} \tag{2.4}$$

Where

$$L = \sqrt{X^2 + Y^2} \tag{2.5}$$

This equation will reach a minimum (Lenart (1983)) at

$$\frac{|XV_{ry} - YV_{rx}|}{V_r} \tag{2.6}$$

$$T2CPA = -(X\,V_{rx} + Y\,V_{ry})\,/\,{V_r}^2 \tag{2.7}$$

It follows that the coordinates of the target at the closest point of approach relative to the origin of our own vessel will be:

$$X(T2CPA) \quad , \quad Y(T2CPA) \tag{2.8}$$

Similarly, the distance from the origin of our own vessel to the closest point of approach is.

$$\sqrt{(V_x\,T2CPA)^2 + (V_y\,T2CPA)^2} \tag{2.9}$$

By defining a safety radius R around the target vessel and provided that the distance at the closest point of approach is shorter than the radius R we are able to find the time $T2R$ it will take our vessel to reach it by solving the equation

$$R = \sqrt{L^2 + V_r^2\,T2R^2 + 2(XV_{rx} + YV_{ry})T2R} \tag{2.10}$$

By rearranging we get

$$0 = V_r^2\,T2R^2 + 2(XV_{rx} + YV_{ry})T2R + (L^2 - R^2) \tag{2.11}$$

We then can obtain the time $T2R$ by solving

$$argmin\left(\frac{-A \pm \sqrt{A^2 - 4V_r^2 B}}{2V_r^2}\right) \tag{2.12}$$

Where

$$A = 2(X\,V_{rx} + Y\,V_{ry}) \quad , \quad B := L^2 - R^2 \tag{2.13}$$

and the distance from the vessel's current position to the point where it will intersect with the safety radius will thus be:

$$D2R = \sqrt{(V_x T2R)^2 + (V_y T2R)^2} \tag{2.14}$$

Determining these parameters with respect to the vessels surrounding our own will prove a useful aid for navigational support if the information is displayed in an intuitive and easily understandable fashion.

2.7 Control Barrier Functions

Control Barrier Functions (CBF) are generally a means of keeping or steering a system within the boundaries of a safe set (Ames et al. (2019)). In the context of maritime navigation and for the purposes of this project, a safe set shall be defined as the area that extends outside of a safety radius R (section 2.6) surrounding an extraneous vessel. The implementation presented in this work will be based on the research previously carried out in Marley, Skjetne, Breivik et al. (2020) and the equations further expanded on Marley, Skjetne, Basso et al. (2021) and Marley, Skjetne, Gil et al. (2023).

Considering an affine control system in the form

$$\dot{x} = f(x) + g(x)u \tag{2.15}$$

with state $x \in \mathbb{R}^n$ and input $u \in U \subset \mathbb{R}^m$. We shall define $B_1 : \mathbb{R}^n \to \mathbb{R}$ as a continuously differentiable function that defines the set

$$K := \{x \in \mathbb{R}^n : B_1(x) \leq 0\} \tag{2.16}$$

B is a CBF for 2.15 if there exists α and a set X with such that, $\forall x \in X$,

$$\inf_{u \in U}[L_f B_1(x) + L_g B_1(x)u] \leq -\alpha(B_1(x)) \tag{2.17}$$

Where $L_f B_1(x)$ is the Lie derivative of B_1 with respect to x along the vector field f and $L_g B_1(x)$ is the Lie derivative of B_1 with respect to x along g.

$$L_f B_1(x) := \frac{\delta B_1(x)}{\delta x} f(x) \quad , \quad L_g B_1(x) := \frac{\delta B_1(x)}{\delta x} g(x) \tag{2.18}$$

Safety is thus achieved by constraining the input u to the admissible input set

$$U_{B_1}(x) := \{L_f B_1(x) + L_g B_1(x)u \leq -\alpha(B_1(x))\} \tag{2.19}$$

A Higher Order Control Barrier Function (HOCBF) is necessary for the event that our control input is not present in 2.18; $L_g B_1(X) = 0$. In this regard, they resemble backstepping control as we need to iteratively construct new CBFs until our control input appears. Thus, B_i, for $i \in \{2, ..., q\}$, be defined by

$$B_i(x) := L_f B_{i-1}(x) + \alpha_{i-1}(B_{i-1}(x)) \tag{2.20}$$

where α_{i-1} are sufficiently differentiable functions. Similarly to 2.18 B_1 is a HOCBF of order q if

$$\inf_{u \in U}[L_f B_q(x) + L_g B_q(x)u] \leq -\alpha(B_q(x)) \tag{2.21}$$

Thus defining the admissible input set.

$$U_{B_q}(x) := \{L_f B_q(x) + L_g B_q(x)u \leq -\alpha(B_q(x))\} \tag{2.22}$$

By constraining the input of a system to the admissible input set we are able to steer and maintain said system within the safe set. This will be used in section 3.3.2 to provide an input for our vessel that will avoid collisions with the surrounding vessels.

Chapter 3

Problem Formulation

This chapter will present the key elements involved in the development of implementation for the project. This includes:

- The challenges involved in utilizing the incoming real-time data and proposed functionalities of a backend capable of addressing them.
- The characteristics of a data visualization interface based on an electronic chart and 3D visualization of the vessel and its environment.
- A proposed implementation for a Collision Avoidance Decision Support Interface based on ARPA parameters and CBFs.

The proposed framework for addressing these challenges will be explored in the following sections.

3.1 Real-Time Data Communication and Processing

In order to establish an online DT for RVG we need to have access to the real-time telemetry and AIS messages collected and relayed by the vessel. In this section, the architecture previously set up by NTNU IT personnel to communicate these messages to an onshore NTNU server, will be described as well as the method currently used to access that datastream. Furthermore, the challenges related to processing and utilizing real-time data will be highlighted.

3.1.1 The Communications Network

The NMEA messages gathered by the sensors aboard RVG are normally relayed to the OLEX computer for charting and navigation purposes; however, in order to relay the NMEA messages to shore it was necessary to tap into this system and, first, relay the generated messages to a port in the local Gunnerus network (gunnerus:25508) via UDP. As can be seen in Figure 3.1, tapping into the NMEA message stream does not prevent them from reaching their intended destination; navigational tasks can continue as per usual.

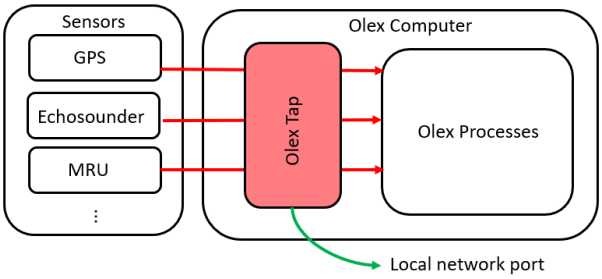


Figure 3.1: Diagram of NMEA message tapping at OLEX computer

User Datagram Protocol (UDP) is a communications protocol mainly used for relaying information quickly, with the caveat that some data might be lost in the process. Data sent via UDP is received as an array of bytes by the listener; this array can be decoded into a string: UTF-8, ASCII, etc. This type of internet communication is suitable for the purpose of this project since a great volume of incoming data has to be relayed in a short span of time. The tapped data stream is received by the Gunnerus Server and then relayed by UDP to the onshore NTNU server (fagitrelay.it.ntnu.no:25508), see Figure 3.2.

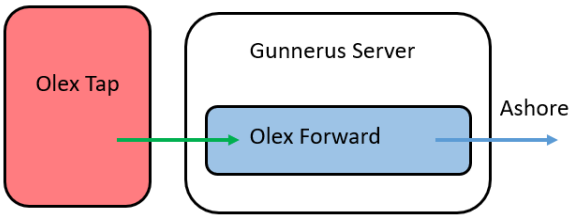


Figure 3.2: Diagram of tapped NMEA message being relayed to the onboard server

The NTNU onshore TCP server listens to the UDP stream being relayed by Gunnerus and then retransmits the data to any TCP client connecting to the NTNU server. It is worth noting that only clients connected to the NTNU network (VPN included) can access this server, see Figure 3.3.

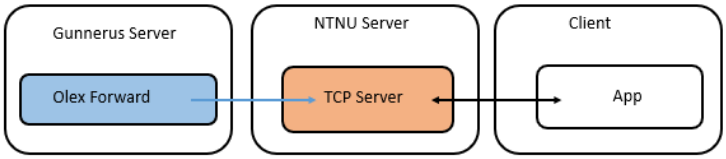


Figure 3.3: Diagram of NMEA message being relayed by on-shore server

The Transmission Control Protocol (TCP) allows for more reliable transmission of data packets at the expense of transmission speed; however, since the address of the client is not known beforehand it is necessary to set up a TCP server. The computers NTNU Ocean Systems Lab (OS-Lab) are an example of such a client.

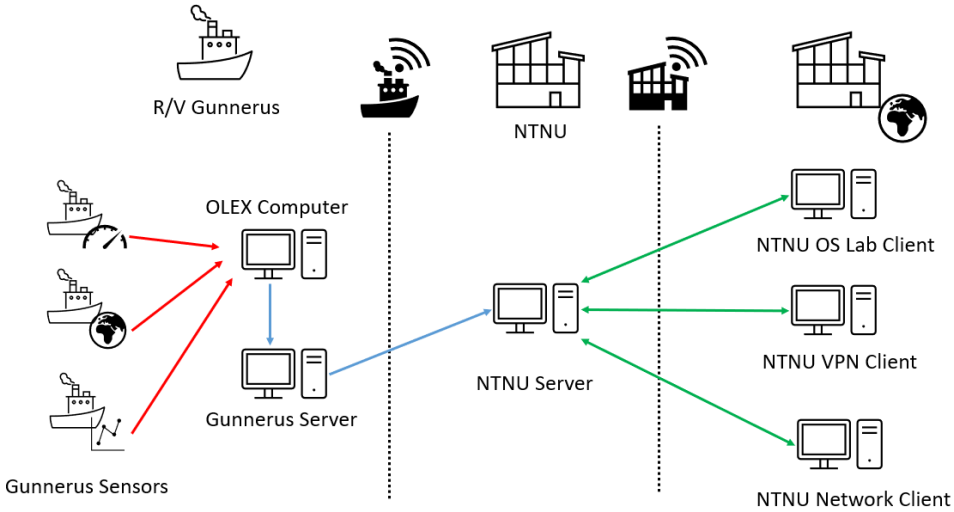


Figure 3.4: Overview of communications topology

An overview of the entire communications network topology can be seen in Figure 3.4; NMEA communications are in red, UDP communications in blue and TCP is in green.

3.1.2 Real-Time Data Processing

Having access to real-time data from RVG and the surrounding vessels is only part of the solution. Once this data is acquired it must also be processed and transformed into something usable by the rest of the applications that desire to use it. Here is where a Data Model backend implementation comes into play. For the implementation of this project, a backend application programmed in Python will be used to receive AIS and NMEA data incoming from RVG, process it, store it if required, and ultimately make it available in a format that the other applications, frontend or else, will be able to use.

An outline of the functional requirements of this backend is as follows:

- Access the real-time datastream and gather incoming messages.
- Parse the incoming messages into a usable format/data structure.
    - Be capable of logging raw data and saving it for historic purposes.
    - Replaying stored raw data for visualization/demonstration purposes.

    - Be capable of logging parsed messages and saving them for historic purposes.
- If necessary, perform operations in the data in order for other applications to use.
- Relay the parsed/transformed data to other client applications.

Some of the challenges arising from this implementation include the sheer volume of incoming data that has to be processed, not to mention the possible errors or glitches that might be present in it. Therefore, some filtering mechanisms will have to be put in place in order to address these issues; be it filtering out the messages unrelated to the desired implementation, or filtering out faulty messages thereof in order to prevent noisy signals from affecting the quality of the data being displayed.

Further details on this backend will be provided in Chapter 4. It is worth mentioning that the processed data will be relayed to external applications using WebSocket.

3.2 Vessel Monitoring and Data Visualization

Besides establishing a connection between the real-time data coming from RVG, another of the objectives of this project is to provide a method for visualizing the incoming data. This interface will consist of two main components. A 2D Electronic Chart (EC) will be used to visualize the situation of RVG in a 2D space along with the related decision support information. And a 3D visualization of RVG and the surrounding vessels.

3.2.1 2D Electronic Chart

An EC User Interface (UI) has to be designed and implemented. Its main objective will be to display data related to RVG as well as its environment in a way that can convey the most information to the user. This must include:

- A map with a distinct marker corresponding to the current coordinates of RVG. This marker should also convey the heading, course, and speed of the vessel.
- A different set of markers corresponding to the situation of surrounding vessels in the vicinity of RVG. Heading, course, and coordinates should also be conveyed.
- A predicted path assuming constant course and speed should be depicted for extraneous vessels.
- The previous positions of extraneous vessels in motion should also be displayed as a trail originating from their current position.
- Additional information on extraneous vessels must be displayed when hovering over their corresponding markers with a cursor.

- ARPA data of relevant vessels must be overlayed in a clear fashion. This overlay should be displayed for every extraneous vessel predicted to approach RVG within a certain radius. Including the point at which the extraneous vessel will be the closest to RVG, and the point at which the vessel will cross a safety perimeter set around RVG.
- A safe Path for maneuvering RVG around obstacles based on CBF should also be displayed along with the time remaining to begin an evasive maneuver.

In addition to this, the UI should be made available in a package that can be easily accessed by the onboard and onshore crew for decision support. Additional settings should be provided to the user in the UI to make it easier to use depending on their situation, these will be discussed in Chapter 5.

3.2.2 3D Visualization

The Autoferry Gemini simulator will be used for the implementation of RVG's 3D visualization, as such, this simulator will be modified in order to leverage its capabilities for the purpose of this project. The Autoferry Gemini simulator is mainly meant to be used as a tool for the simulation of VL, IL, and the related sensors, in the context of dynamic positioning for marine vehicles in Trondheim Harbor. As such, performing some modifications will be necessary in order for it to accommodate the objectives of this project:

- A 3D model of RVG must be procured and inserted into the simulator in order to visualize it.
- A method for receiving real-time NMEA and AIS data must be put in place.
- The 3D model for RVG should be connected to the incoming NMEA data in a way that will reflect the current attitude and location of its real-life counterpart.
- Surrounding vessels must also be visualized in this simulation and connected to the AIS data related to their real-life counterparts. This includes their location and heading.
- A mechanism should be put in place in order to stream the scene over a server and display it on the web application hosting the EC.

3.3 DSS for Safe Navigation

In addition to providing a method for visualizing and monitoring the situation of RVG, the UI described in the previous section should be capable of providing a Decision Support System (DSS) for the Officer on Watch (OOW). The DSS proposed as part of this work is mainly focused on providing assistance for Collision Avoidance (COLAV). This information must be conveyed to the crew in a way that highlights potential threats and provides a maneuvering suggestion for avoiding them.

3.3.1 Display of ARPA Parameters

Displaying ARPA parameters can provide useful information to the crew since they provide information related to surrounding obstacles and their behavior, i.e.; how their trajectory could interfere with RVG's course and how close these could approach RVG in such an event. The ARPA parameters that will be taken into consideration for the objectives of this project are the same as those described in Section 2.6, however, some parameters will be introduced in order to avoid confusion when referring to the CPA:

- Distance to Closest (D2CPA): relative distance from RVG's current position to the point on its trajectory at which it will be the closest to a target vessel. From eq. 2.9 we get:

$$D2CPA = \sqrt{(V_x\, T2CPA)^2 + (V_y\, T2CPA)^2} \tag{3.1}$$

- Distance at Closest Point of approach (D@CPA): the distance between RVG and the target vessel when CPA is reached. From eq. 2.6 we get:

$$D@CPA = \frac{|XV_{ry} - YV_{rx}|}{V_r} \tag{3.2}$$

For the calculations presented in the following subsections, we will use the position, course, and speed over ground data provided by AIS as the information on the target vessels. This information is provided by the related AIS and NMEA telegrams as the coordinates of either vessel, their course in degrees starting clockwise from the north, and their speed over the ground in knots. To use this information with the equations described in Section 2.6, we need to project the coordinates of RVG and the extraneous vessel onto a plane tangent to the earth's surface, note that this tangent plane will have its origin centered on RVG's current position, this can be achieved by transforming the geodetic coordinates to cartesian coordinates as described in Hofmann-Wellenhof et al. (1997). For the purposes of this project, we shall consider the height above ground h to be zero and disregard Z.

$$X = (N+h)cos\phi cos\lambda, \quad Y = (N+h)cos\phi sin\lambda, \quad Z = \left(\tfrac{b^2}{a^2}N + h\right)sin\phi \tag{3.3}$$

$$N = \frac{a^2}{\sqrt{a^2cos^2\phi + b^2sin^2\phi}} \tag{3.4}$$

Where ψ and λ are the longitude and latitude in radians; N, the radius of curvature in prime vertical; and a, b the semiaxes of the reference WGS-84 Ellipsoid. Since we want to position RVG as the origin of our reference O_x, O_y, we must subtract its cartesian coordinates X_{RVG}, Y_{RVG} from the cartesian coordinates of our target vessel X_i, Y_i.

$$X = X_i - X_{RVG} \quad , \quad Y = Y_i - Y_{RVG} \tag{3.5}$$

In order to use the equations for ARPA parameters from Section 2.6 we need to obtain the relative velocity V_{ir} of the target vessel relative to ours. Thus

$$V_{irx} = V_{itx} - V_x \quad , \quad V_{iry} = V_{ity} - V_x \tag{3.6}$$

Where V_{itx}, and V_{ity} are true components for the speed of the target vessel; and V_x, and V_y are the components for the speed of RVG. Using the course ψ_i of the target vessel and its speed over ground V_i we obtain.

$$V_{itx} = V_i sin\psi_i \quad , \quad V_{ity} = V_i cos\psi_i \tag{3.7}$$

Similarly using RVG's course ψ and speed over ground V.

$$V_x = Vsin\psi \quad , \quad V_y = Vcos\psi \tag{3.8}$$

Having obtained all the parameters necessary for performing the calculations presented in Section 2.6. The implementation of these parameters will be presented in Chapter 4 and their display in Chapter 5.

3.3.2 CBF-based Guidance for Safe Maneuvering

The main task of CBF-based guidance for safe maneuvering is to provide a reactive trajectory that maintains RVG on course while avoiding any vessels which might pose a collision threat. We can use the CBF introduced in Section 2.7 in our application by using the equations provided in Skjetne (2023). It is worth noting that the following model is not taking into consideration the dynamics of the vessel, nor is it taking into consideration any kind of disturbances. COLREGs are not being considered in the calculation of the safe trajectory, this should be taken into consideration when a maneuver is proposed by the decision support system.

For this system, we will use the cartesian coordinates obtained in the previous section as positions p and p_i.

$$p = \begin{bmatrix} X_{rvg} \\ Y_{rvg} \end{bmatrix} \quad p_i = \begin{bmatrix} X_i \\ Y_i \end{bmatrix} \tag{3.9}$$

Orientations z and z_i will thus be obtained from courses ψ and ψ_i.

$$z = \begin{bmatrix} sin\psi \\ cos\psi \end{bmatrix} \quad z_i = \begin{bmatrix} sin\psi_i \\ cos\psi_i \end{bmatrix} \tag{3.10}$$

First, we define the following system where the change in position p is defined as

$$\dot{p} = Vz \quad , \quad V > 0 \tag{3.11}$$

With V being our own vessel's speed over ground, and $z \in \mathbb{S}^1$ its orientation. Thus, the change in the vessel's orientation can be defined as

$$\dot{z} = rSz \quad , \quad S := \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \tag{3.12}$$

Where $r \in \mathbb{R}$ is the yaw rate used as our control input and S is a 90^o rotation matrix. Furthermore, we consider an extraneous target vessel with dynamics

$$\dot{p_i} = V_i z_i \tag{3.13}$$

Where U_i and z_i are assumed constant. We thus define the state vector $x := (p_i, p, z) \in \mathbb{R}^2 \times \mathbb{R}^2 \times \mathbb{S}^1$ so that

$$\dot{x} = f(x) + g(x)r := \begin{bmatrix} V_i z_i \\ Vz \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ Sz \end{bmatrix} r \tag{3.14}$$

Assuming there is a circular obstacle with a radius R centered in p_i relative to our vessel position p we define the obstacle domain:

$$K_o := \{e \in \mathbb{R}^2 : |e| < R\} \tag{3.15}$$

where $e_i = p - p_i$ and $|e_i| := \sqrt{e_i^T e_i}$ is the Euclidean norm. For each target ship p_i, we define a continuously differentiable function $B_1 : \mathbb{R}^n \to \mathbb{R}$

$$B_1(x) := R - |e| \tag{3.16}$$

Safety is ensured if we can maintain

$$B_1(x(t)) \leq 0 \quad \forall t \geq 0 \tag{3.17}$$

Thus from equation 2.18 we differentiate B_1. Note that $\dot{e_i} = (Vz - V_i z_i)$

$$\dot{B}_1 = L_f B_1 + L_g B_1 r = -\frac{2e_i^T \dot{e_i}}{2\sqrt{e_i^T e_i}} = -\frac{e_i^T}{\sqrt{e_i^T e_i}}(Vz - V_i z_i) = -\frac{e_i^T (Vz - V_i z_i)}{\sqrt{e_i^T e_i}}$$

$$L_f B_1(x) = -\frac{e_i^T (Vz - V_i z_i)}{\sqrt{e_i^T e_i}} \quad , \quad L_g B_1(x) = 0$$

We can notice that since $L_g B_1(x) = 0$ there is no input r that will satisfy the constraint

$$U_B(x) := \{L_f B_1(x) + L_g B_1(x) u \leq -\alpha_1(B_1(x)) \tag{3.18}$$

Therefore a HOCBF B_2 is introduced as in equation 2.20.

$$B_2(x) := \dot{B1} + \alpha_1(B1) \tag{3.19}$$

Where $\alpha_i(B_i) = \frac{1}{t_i} B_i$ with a time constant t_i. Therefore,

$$B_2 = \dot{B}_1 + \frac{1}{t_1}B_1 = L_f B_1 + \frac{1}{t_1}B_1 \tag{3.20}$$

$$\dot{B_2} = L_f B_2 + L_g B_2 r = \ddot{B}_1 + \frac{1}{t_1}\dot{B}_1 \tag{3.21}$$

By differentiating $\dot{B_1}$ again, thank we obtain

$$\ddot{B_1} = -\frac{\left(\dot{e_i}^T\sqrt{e_i^T e_i} - e_i^T\frac{2e_i^T\dot{e_i}}{2\sqrt{e_i^T e_i}}\right)}{e_i^T e_i}(Vz - V_i z_i) - \frac{e_i^T}{\sqrt{e_i^T e_i}}V\dot{z} \tag{3.22}$$

$$= \frac{\left[e_i^T(Vz - V_i z_i)\right]^2}{|e_i|^3} - \frac{|Vz - V_i z_i|^2}{|e_i|} - \frac{e_i^T}{|e_i|}VrSz \tag{3.23}$$

Thus

$$L_f B_2 = \frac{\left[e_i^T(Vz - V_i z_i)\right]^2}{|e_i|^3} - \frac{|Vz - V_i z_i|^2}{|e_i|} + \frac{1}{t_1}L_f B_1 \tag{3.24}$$

$$L_g B_2 = -V\frac{e_i^T}{|e_i|}Sz \tag{3.25}$$

Resulting in the safe control set

$$U_B := \left\{r \in \mathbb{R} : \dot{B_2} \leq -\frac{1}{t_2}B_2\right\} = \left\{r \in \mathbb{R} : L_f B_2 + L_g B_2 r + \frac{1}{t_2}B_2 \leq 0\right\} \tag{3.26}$$

We can now define a nominal controller r_{nom} which will help us pick an input from the safe set U_B. Letting z_{ref} be a reference course, and $\tilde{z} := \begin{bmatrix} z_{ref} & Sz_{ref} \end{bmatrix}^T z$ we get

$$r_{nom} := \frac{-k_p\tilde{z}_2}{\sqrt{1 - \lambda^2\tilde{z}_1^2}} \tag{3.27}$$

Thus, a safety critical control can be obtained by the optimization

$$r_{safe} = arg\,min_{r \in U_B}(r - r_{nom})^2 \tag{3.28}$$

which applied as control law. The explicit solution becomes

$$r_{safe} = \begin{Bmatrix} r_{nom}(x), & r_{nom}(x) \in U_B(x) \\ r_{nom} - \frac{ab^T}{bb^T}, & r_{nom}(x) \notin U_B(x) \end{Bmatrix} \tag{3.29}$$

where

$$a := L_f B_2 + L_g B_2 r_{nom} + \frac{1}{t_2}B_2 \tag{3.30}$$

$$b := L_g B_2 \tag{3.31}$$

This will be run as a simulation assuming a constant course and speed for the extraneous vessel. Furthermore, we shall use the Time to Evasive Maneuver (TTEM) from Marley, Skjetne, Gil et al. (2023), to relay information to the Officer on Watch (OOW) about the time remaining until an evasive maneuver has to be started. The TTEM will correspond to the time in the simulation when the nominal control r_{nom} will no longer maintain RVG in the safe domain U_B, eq. 3.29.

The system presented in this section will be used in the following sections as a means to provide a trajectory for RVG to follow and avoid obstacles while trying to maintain its course. The Implementation of these parameters will be presented in Chapter 4, and their display in Chapter 5.

Chapter 4

Real-Time Data Interfacing and Processing

Once the sensor data gathered by RVG is made available in the NTNU server, it is necessary to process it and make it available to a number of applications; such as simulation, visualization, or logging and saving the incoming data for future use.

This section will present the proposed design for a backend application capable of processing the data incoming from RVG in Real-Time. This processed data should be formatted in a way that will be usable by the **Vessel Monitoring and Data Visualization** (Section 3.2), and it will make use of the **DSS for Safe Navigation** (Section 3.3) implementations described in the previous chapter.

Furthermore, the computational methods for obtaining the CBF and ARPA parameters for Decision Support and Safe Navigation will be explored in this chapter.

4.1 Proposed Design

A Data Model (DM) was proposed for processing and making available the incoming data; The DM itself is a class composed of different components with specialized functions, see Figure 4.1.

- **Data Model**: This is the main class containing the combined functionality of the classes that compose it. Parameters for their execution will be exposed here for the user to modify, for example; running the DM from real-time data or a log, saving logs, printing debugging outputs, etc.
- **Data Parser**: This component will be in charge of handling the incoming data; be it a real-time UDP stream containing AIS and NMEA data, or a previously saved log. This class will include a TCP **Socket** for interfacing with the NTNU server and receiving the incoming data packets; a **Parser** in charge of decoding the incoming NMEA messages, correcting any detectable errors in the incoming string, and decrypting if necessary; finally, the parsed messages will be stored in a **Buffer** in order to make them avail-

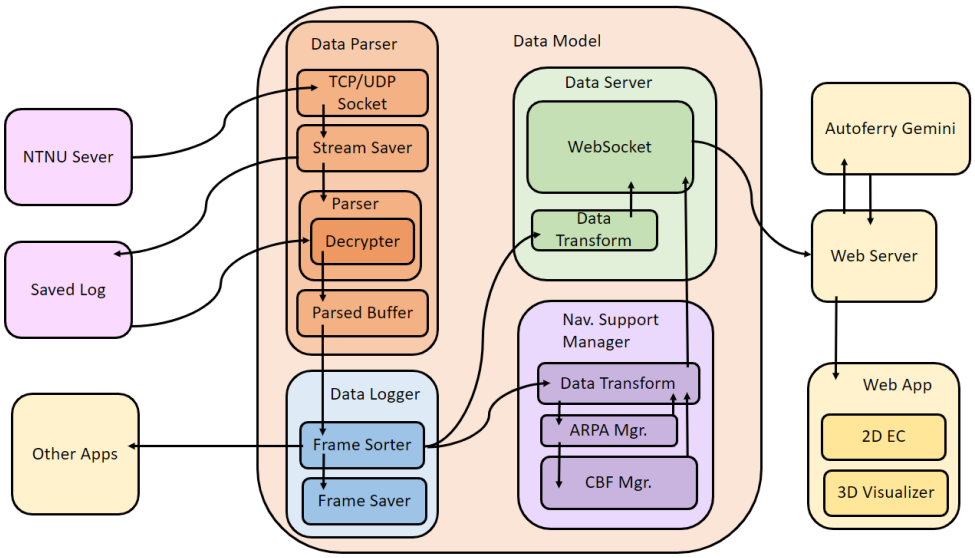


Figure 4.1: Diagram for proposed Data Model

able to the rest of the DM. Additionally, this component should be able to save a copy of the incoming raw data in case it should be required for later experimentation.

- **Data Logger**: This component will process the Parsed Buffer from the Data Parser in order to make the sorted data available for the DM. Two classes will make up this class: a **Frame sorter** in charge of processing the buffered messages and storing them in different data frames according to their individual content or *type*; and a **Frame Saver**, which will be in charge of saving the sorted data frames as comma separated value files at the end of execution for later use (csv file), see Figure 4.2. The sorted data frames originating from this component should, at this stage, be usable by other apps or APIs with access to the DM. Alternately, a **Fast Logger** for real-time applications can be used to relay data without sorting it into frames in order to reduce latency.
- **Data Server**: This component will be in charge of making the sorted data frames available to any application with no direct access to the DM. Again, it will be made up of two different components; a **Data Transform** in charge of taking the data available from the sorted data frames and applying the necessary ad hoc transformations required by the external application, and a **WebSocket** for sending the transformed data over to a port accessible by the external application.
- **Navigational Support Manager**: This component will be in charge of using the sorted Data to perform the necessary calculations related to the obtainment of ARPA parameters and the CBF safe trajectory. Two main components are at the core of its operation; the **ARPA Manager**, and the **CBF Manager**. Additionally, the **Data Transform** component is reused here to convert co-

ordinates to NED and back.

4.2 The Data Model

The actual implementation of the proposed DM has been carried out in Python with the assistance of an environment managed by Conda. A description of the involved classes and methods will be presented in this section.

The Data Model works as a container for all of the components involved in the RVG data interfacing. Initialization parameters are declared and made available here for modification prior to execution. However, the most important role of the DM is to manage the execution of the main components: due to the nature of the processes involved these need to be executed concurrently so as to not block the execution of the other components; therefore each one is executed on its own thread. Additionally, the DM is in charge of starting and terminating the threads upon user request.

4.2.1 The Data Parser

The main purpose of the data parser is to parse the incoming data from either; the NTNU server or a previously saved log. To this effect, a base **Parser** class was created with the necessary methods to parse incoming messages and making storing them in a buffer. The main parsing loop is as follows:

1. The raw message is received.
2. The raw message is attempted to be parsed as an NMEA message or an AIS message; if successful the message is stored in the parsed buffer and the loop goes back to step 1 (this shall be marked with an '*').
3. The raw message is attempted to be parsed as an encrypted message by the **Decrypter**.*
4. At this step there might be an error with the incoming message, an attempt is made to fix any bad end-of-line terminators resulting in two messages being spliced. If this error was successfully fixed, a list with the fixed messages is created and parsed recursively; each element of the list is sent to step 1(this shall be noted as '**').
5. At this step there might be an error with the incoming message, an attempt is made to fix collated sentences resulting in two or more messages being spliced**.
6. If this step was reached then the message is dropped, an error is logged and the loop goes back to 1.

Additionally, if the data is coming from the TCP socket, then the raw messages can be saved to a .txt file by the **Stream Saver**. These saved text streams can also be used as a data source for replaying tests.

4.2.2 The Decrypter

Some messages have been augmented with ancillary data containing the UNIX time, sequence number, source id, and source name. For safety reasons, these augmented packets are encrypted before being relayed to the shore server, so they must be decrypted once they arrive at the Parser; the decryption process is carried out by the Decrypter component. Since some of the encrypted messages are too long they have to be broken up into segments and sent consecutively over UDP; this information is available prior to decryption and is used to assemble the encrypted message as they are received and prior to their decryption. A public encryption key is required for carrying out this process:

1. Receive encrypted message.
2. The unencrypted message information is used to store the message in an array for its assembly prior to decryption.
3. The array containing the segmented messages is checked to see if any of the messages have been fully assembled upon step 2.
4. Fully assembled messages are decrypted with the corresponding key and relayed back to the Parser with the augmented data.
5. Back to step 1.

4.2.3 The Data Logger

This component takes the buffer of parsed messages and sorts them into structured data frames. The parsed message queue is made up of Python objects with ad-hoc named fields containing the individual values for each message. The Data Logger contains a Sorted Data class which allows for properties to be created dynamically on it and store the incoming messages in them; if available, the data frame headers originating from previous executions are used to preemptively create data frame properties. The sorting loop or **Frame Sorter** goes as follows:

1. The first element of the parsed buffer is stored in the Data Logger and removed from the buffer.
2. The id, attribute names, and attribute values of the message are extracted.
3. A time series header is created for the incoming data frame if it does not yet exist and used to create a new data frame attribute in the Sorted Data **class**.
4. The data for the incoming message is stored in the Sorted Data **object** attribute corresponding to its id; individual values of the message will be added to the corresponding field in the target data frame. Every GPGGA message will be concatenated to the GPGGA frame, and so on.
5. Back to step 1.

At the end of the execution, the data frames contained within the Sorted Data Object can be saved as csv files by the **Frame Saver**, Figure 4.2. Prior to that, the Sorted Data object is always available for the DM to access. It is worth noting that

	msg_type	timestamp	unknown_1	tcvr_num	tdcr_num	roll_deg	pitch_deg
0	SNS	80220.805		1	1	0.1	-0.95
1	SNS	80221.805		1	1	0.11	-0.94
2	SNS	80222.807		1	1	0.14	-0.9
3	SNS	80223.807		1	1	0.22	-0.97
4	SNS	80224.807		1	1	0.27	-0.92
5	SNS	80225.807		1	1	0.22	-0.84
6	SNS	80226.807		1	1	0.15	-0.92
7	SNS	80227.807		1	1	0.17	-0.99
8	SNS	80228.807		1	1	0.19	-0.94
9	SNS	80229.807		1	1	0.09	-0.89
10	SNS	80230.808		1	1	-0.09	-0.92

Figure 4.2: Example of a property available in the Sorted Data object, taken from the resulting saved csv file.

aliases for the fields of any given message can be handed to the Data Logger and used as the field values.

Real-time applications require data to be processed as fast as possible in order to reduce latency. To this effect, a **Fast Logger** has been implemented inheriting the Data Logger. The main difference lies in the fact that the Fast Logger drops all but the messages marked as required and that the Fast Logger is unable to sort and save data into frames. Instead, the messages marked as required are processed and added to a buffer for the **Data Server** to relay. The process is as follows:

1. The first element of the parsed buffer is stored in the Data Logger and removed from the buffer.
2. The id, attribute names, and attribute values of the message are extracted.
3. The data for the incoming message is appended to the Sorted Data **list**.
4. Back to step 1.

The user is able to select which version of the Data Logger to use upon running the DM. However, this means that data cannot be sorted into different frames while using the **Fast Logger**.

4.3 Data Server for External Applications

The **Data Server** is an additional component available in the event of an external application not being able to directly interact with the Sorted Data element available in the DM. It works by compiling the information made available in the Sorted Data object, transforming it, and sending it via WebSocket to a port in the local machine:

1. Verify the availability of messages in the Sorted Data element.
2. If necessary, perform the desired **Data Transform** on the data and obtain a transformed data frame if necessary.

3. The transformed data is used to compose a JSON message that can be relayed through WebSocket.
4. Take the JSON message and send it via **WebSocket**.
5. If a new relevant element is detected in the source frames, go to step 2.

Note that the WebSocket is also able to forward information from the **Navigational Support Manager**. The **Data Transform** performed by the simulation server is defined by the user and has to be created ad-hoc for the simulation case. The current data transform implemented in the data server performs any of the following transformations and manipulations.

Data Transform for 3D Visualization:

1. Gather GPS and attitude data from GPGGA_ext and PSIMSNS_ext, and AIS frames.
2. Convert GPS data to Decimal Degrees (DD).
3. Take DD data and convert it to ENU based on the WGS-84 projection taking the origin of the simulation as the origin coordinates.
4. Return frames with resulting position (x, y, z) in meters, attitude (roll, pitch, yaw) in degrees, and Unix time. In the case of AIS data, only the heading is provided as a reference for attitude.

Creating and Filtering AIS Historic Data

1. Gather MMSI, course, longitude, and latitude coordinates from AIS frames.
2. Create an entry in a dictionary with the vessel's MMSI. This dictionary entry contains lists for the course, longitude, and latitude data.
3. Once the lists have been filled with a minimum of required items, these are filtered using a SciPy Butterworth filter.
4. The resulting filtered lists are saved as part of the dictionary entry.
5. The filtered data is added to the original message.

Timing Data Relay for Replay

If the Data Model is replaying a previously saved log the following procedure is executed by the Data Server to ensure that the messages are transmitted with the appropriate timing.

1. The time at the start of execution t_{start} is saved as a reference.
2. The list of parsed messages is sorted by organizing the timestamps in ascending order.
3. The timestamp of the first message $t_{first\,message}$ in the list is saved as a second reference.
4. A simulation time used for timing the relay of the messages is computed for the relay of the messages in the list by:

$$t_{simulation} = t_{first\,message} + (t_{current} - t_{start}) \tag{4.1}$$

where $t_{current}$ is the current time at the execution of that instruction.
5. If the simulation time $t_{simulation}$ is greater than the timestamp of the current message; the message is relayed and we get the next message in the list, otherwise we wait.
6. Once all of the messages in the list have been sent we can start over from the top or stop (user-defined).

Additionally, the **Data Transform** component provides other components with tools for implementing the aforementioned transformations on demand, including the conversion from geodetic to ENU and vice-versa.

4.4 Navigational Support Features

The **Navigational Support Manager** is the component responsible for collecting AIS data from the surrounding vessels, comparing it with RVG's motion and location, and providing the necessary data for the display of the DSS for safe maneuvering. Due to its computational cost, these computations are performed on a set interval defined by the user. Its overall operation is as follows.

1. Collect AIS data from surrounding vessels.
2. Convert the coordinates of the surrounding vessels to ENU relative to RVG's position using the **Data Transform** component.
3. Use the **ARPA Manager** to obtain the ARPA parameters of these vessels and determine which ones will approach RVG within a user-defined Safety Radius R.
4. The obtained ARPA for each vessel are compiled into a JSON message and sent via the **Data Server's** WebSocket.
5. The ARPA data collected from step 3 is used by the **CBF Manager** to determine which vessels (if any) should be taken into account in order to generate the safe trajectory suggestion.
6. The **CBF Manager** runs a simulated scenario using the relevant AIS and RVG data to determine a safe trajectory.
7. The cartesian coordinates resulting from the safe trajectory are converted to longitude and latitude using the **Data Transform** component.
8. The trajectory coordinates are compiled into a JSON message and sent via the **Data Server's** WebSocket.

It is worth noting that the **Data Transform** component is essentially shared by the **Data Server** and the **Navigational Support Manager**.

4.4.1 ARPA Calculations

The **ARPA Manager** is effectively an implementation of the equations for obtaining ARPA parameters presented in Section 2.6, while the AIS data from the surrounding vessels will be used by virtue of the transformations presented in Section 3.3.1 implemented in the **Data Transform** component. Throughout the execution

of the DM, this component is constantly gathering data related to incoming AIS Frames and entering it in a dictionary. This dictionary has its keys bound to the MMSI of the AIS vessels and each entry contains the latest information for the speed over ground, course, longitude, and latitude coordinates of each vessel. When the time comes to perform the ARPA calculations, the following steps are followed:

1. Make a deep copy of the AIS dictionary in order to prevent it from changing through the course of the computations.
2. Initialize an empty list. Elements of this list will be comprised of dictionaries containing the ARPA parameters for the processed vessels.
3. Get current RVG data for speed over ground, course, longitude, and latitude.
4. Convert RVG coordinates to ENU and use its speed over ground and course to determine V_x and V_y.
5. For each AIS vessel i obtain its ENU position relative to RVG X_i, Y_i, as well as its true speed components V_{itx}, and V_{ity} based on its course and speed over ground.
    a. Obtain the corresponding CPA to RVG for this vessel along with D@CPA, D2CPA, and T2CPA.
    b. If the vessel is within the established safety radius R plus a tolerance R_{tol}[1] go to the next step, if not, continue to the next vessel.
    c. Verify if the CPA of the target vessel is within the safety radius R (now without the tolerance). If the CPA is not within the safety radius; compile the data collected on the vessel thus far in a dictionary and append it to the list in Step 2, then continue to the next vessel. Otherwise, continue to the next step.
    d. Obtain the D2R and T2R parameters for this vessel, then compile the data collected on the vessel thus far in a dictionary and append it to the list in Step 2
    e. Continue to the next vessel.
6. Verify if the list from step 2 has been filled with ARPA parameters for the vessels. If it is; continue to the next step. Otherwise, return the empty list and RVG parameters.
7. Convert the ARPA parameters in each element of the list from ENU back to geodetic coordinates. The resulting conversion should be stored in a new list while leaving the original one unaltered.
8. Compile the resulting list in a JSON message and send it via WebSocket.
9. Return the original list and RVG parameters.

The converted list sent by WebSocket will be used for displaying the ARPA parameters in the 2D EC, the usage of which shall be explored in Chapter 5. On the other hand, the list containing the ARPA parameters still in ENU coordinates

[1]The added tolerance R_{tol} allows visualization of the ARPA parameters of vessels that will not breach the safety radius R, but whose parameters are still deemed relevant for the display.

will be used for performing the calculations in the next section.

4.4.2 CBF Computations

In the previous section, we have identified the vessels that might pose a collision threat for RVG. In this section, we must perform the calculations for creating a safe trajectory using the **CBF Manager**. These calculations come from the equations presented in Section 3.3.2. The **CBF Manager** simulates the trajectory of the identified vessels and provides a control input r_{safe} for the rotation rate of a simulated RVG, which will steer it away from the extraneous vessels and ensure that it will always maintain a safe distance between the vessels and itself. **It is very important to note that this does not take into consideration the dynamic properties of RVG nor the extraneous vessels and that the real RVG might not be able to precisely follow the projected trajectory**. Thus, the resulting trajectory should be seen as a suggestion of a course for RVG to follow in order to prevent collisions.

Note that the following procedure takes as its input the ENU data for RVG and the relevant vessels from the previous section, and it is only triggered if any of the vessels pose a threat i.e; if they get closer than the safety radius R:

1. The time at the start of the CBF calculations is saved.
2. The parameters p, V, z, z_{ref} for RVG are obtained from the input data. Desired course z_{ref} is assumed to be the course of RVG at the start of the computations.
3. The parameters p_i, V_i, z_i for the extraneous vessels are obtained from the input data
4. The time for the simulation t is initialized, as well as an array for recording the positions of the simulated RVG at every step of the simulation.
5. Since the speed and course of the target vessels is assumed to be constant, we can obtain the position for the extraneous vessels at every step of the calculation at once. This is done beforehand to save computation time.
6. For every step Δt of the simulation we perform the following:
    a. Obtain the nominal control r_{nom} that will keep the simulated RVG on the desired course
    b. Obtain the Euclidean norm $|p-p_i|$ for every target vessel with relation to RVG and determine which one is the closest.
    c. Use the parameters of the closest vessel to obtain e_i and perform the operations from the equations in Section 3.3.2 to obtain B_1, L_fB_1, B_2, L_fB_2, and L_gB_2
    d. Check if the nominal control input r_{nom} will keep RVG in the safe set by comparing

$$\dot{B}_2 \leq \frac{-1}{t_2} B_2 \tag{4.2}$$

where

$$\dot{B}_2 = L_f B_2 + L_g B_2 r_{nom} \tag{4.3}$$

e. If the inequality is true, we use r_{nom} as our safe input r_{safe} , if it doesn't we obtain r_{safe} from equations 3.29, 3.30, and 3.31. We also mark this time step in our simulation and save it as the Time to Evasive Maneuver (TTEM).
f. We update the state of our system with the calculated r_{safe} using equations 3.11 and 3.12.
g. We add the updated position of RVG to the position history from step 4.

7. Once the simulation has performed all its time steps we convert the resulting position history to geodetic coordinates, compile it into a JSON message along with the time of the start of the evasive maneuver, and send it via WebSocket.

It is worth noting that the suggested trajectory is reactive. This means that an evasive maneuver A can be calculated at time t_A based on RVG's course at that time $z_{ref}(t_A)$; then, at time t_B while RVG is executing maneuver A a new evasive maneuver B will be computed for the course at that time $z_{ref}(t_B)$. This could result in suggested trajectories A and B being different depending on the scenario, i.e.; a change in the course of either, RVG or the target vessel. Consequently, this will also affect the TTEM for A and B.

The parameters for the CBF t_1, t_2, λ, and Δt, as well as the total simulation time, can be set from the Data Model when initializing the component. The data resulting from the calculation presented in this section will be used for the display of the suggested CBF-based trajectory in the 2D EC in Chapter 5.

Chapter 5

Vessel Monitoring and Data Visualization

This chapter will present the implementation for Vessel Monitoring and Data Visualization. As previously mentioned, this will consist of a 2D EC and a 3D Visualization based on the Autoferry Gemini simulator. These will be made accessible to the user by a Web App hosting the 2D EC as well as an Online 3D Stream coming from the Autoferry Gemini simulator. Details on the development of these components will be presented in the following sections.

5.1 The Web Server

It is important to note that the **Web Server** mentioned in the previous chapter arose as a part of the implementation of the Unity Render Streaming (Unity Technologies (2020)) in Section 5.2.2 which will be presented later in this Chapter. The Web Server is a server application included in the Render Streaming Plug-in for the Unity development environment, and it is originally intended for arbitrating communications between a Unity application running on a local machine, and a remote web application receiving the stream from the Unity app. However, this Web Server has been modified to also accept incoming data from our Data Model (DM) and communicate it to its clients.

The Web Server can be accessed via WebSocket by the DM for publishing data that will be accessed by both; the 2D Electronic Chart Web App and the Unity Autoferry Gemini simulator. Similarly, these applications will subscribe to the Web Server in order to access the published data. Additionally, the Web Server hosts the application that allows for visualization of the 3D stream originating from Autoferry Gemini. Due to incompatibilities, the React.Js web app containing the 2D EC could not be hosted on the same server, nevertheless, it does provide access to the 3D Render Stream through a link. These interactions can be seen in Figure 5.1.

Another detail worth noting is that, while this web server can enable web

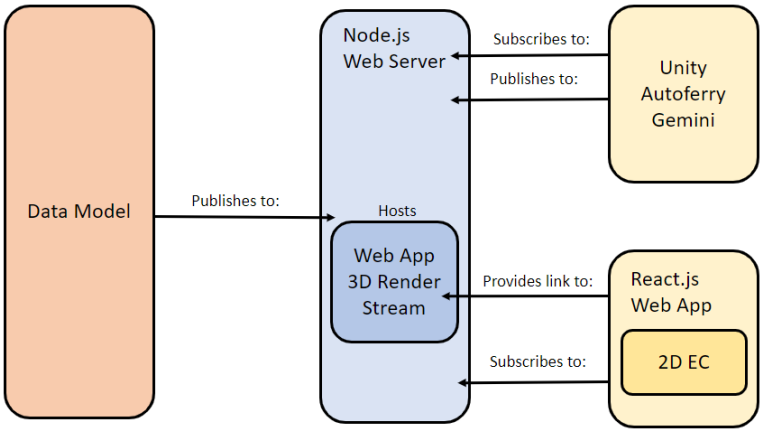


Figure 5.1: Diagram for communications between Web Server and other applications.

capabilities for this project, i.e.; being able to access the information remotely, this is not yet the case as it hasn't yet been set in a machine for it to be publicly accessed. This possibility, while theoretical at the moment of writing this thesis, is still on the table and should be explored in future works.

5.2 3D Visualization

The Autoferry Gemini Simulator is based on the Unity game engine and was originally conceived as a way to gather data from a variety of exteroceptive sensors: IL, VL, Radar, Lidar, etc. To this effect, the simulator comes with assets modeled in the likeness of the Trondheim Harbor, as well as some small vessels; see Figure 5.2.

The expected setup for this simulator is to have the Unity Editor running on Windows; Robot Operating System (ROS) running on a Ubuntu virtual machine, and have them interfacing through internal network connections. The ROS side of the simulation receives the input from the simulated sensors and is capable of sending commands back for the vessel's equipment to perform. However, for the purposes of this project, the ROS side can be disregarded for the time being; in its place, the 3D assets for the vessel will be receiving data directly from the DM via WebSocket and the transformations will be carried out by a C# script in the Unity. Similarly, information regarding the position and heading of vessels in the area will be relayed through the same mechanism. It is worth noting that transformation, in this context, specifically refers to situating any of the vessels in a coordinate relative to the origin of the simulation with their respective attitude.

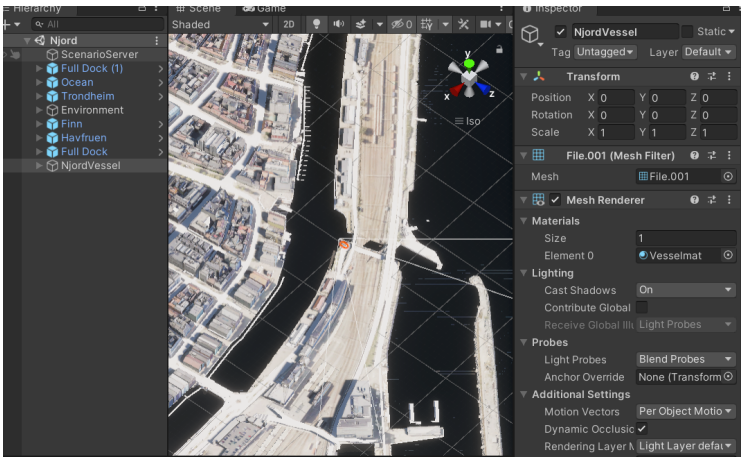


Figure 5.2: Screenshot of the Trondheim Harbor being rendered in the scene, on the Gemini simulator (Vasstein et al. (2020)).

5.2.1 Handling Data for 3D Visualization

The Unity engine is set up so that scripts can be imported as assets in the scene and, have them attached to other objects in the same scene. A Real-Time Scenario script was thus set up as an asset in the Trondheim Harbor scene for interacting with a given boat instance; see Figure 5.3.

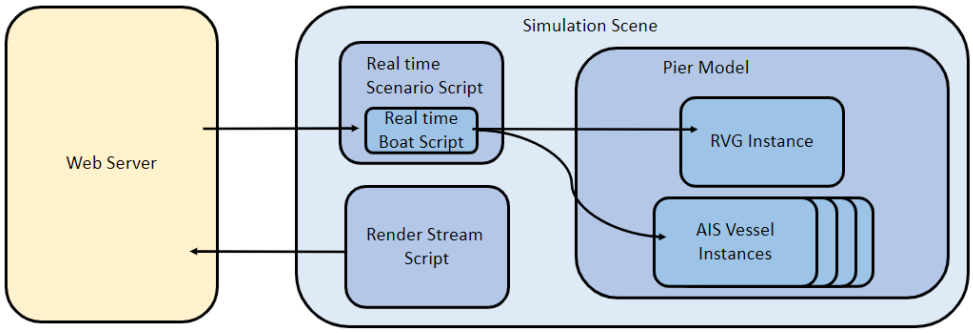


Figure 5.3: Diagram of interaction between Data Model and vessel model rendered in the simulator.

The script is currently comprised of two parts: A **Real-Time Scene** script which acts as a wrapper, for triggering the execution of the contained scripts; and a **Real-Time Boat** script, which is in charge of connecting to the Web Server via WebSocket and transforming the target vessel's pose according to the incoming data. The way the script functions is as follows:

1. When the scene starts playing the Real-Time Scenario script starts the Real-Time Boat Script and hands it over the reference for the in-scene vessel it

will be targeting.
2. Upon initialization the Real-Time Boat script establishes a connection with the WebSocket and starts receiving data.
3. Received messages are identified as either RVG data or AIS data.
4. If the incoming message is related to RVG its data is used to update the properties of the RVG assets in the scene.
5. If the incoming message is related to AIS its data is used to update the properties of the target AIS assets in the scene. These AIS assets are stored in a list in order to create multiple instances of AIS vessels and update their respective properties.
6. The scene has its physics updated at a 50hz rate, upon every refresh the Real-Time Boat script updates the visual properties of the assets in the scene, i.e.; its position and attitude.

Currently, an untextured 3D asset of RVG is used in the simulator, while a default asset is used for the display of the AIS vessels. The current model for RVG was procured from a previous RVG DT project (Rølvåg and Stranden (2022)).

5.2.2 Forwarding Data for 3D Online Visualization

As stated in Section 5.1 of this chapter, it is possible to establish a stream of the rendered scene thanks to Unity's Render Stream plugin. This is achieved by adding the provided Render Stream script to the scene and, in this instance, connecting it to the desired cameras. The script is streaming the information captured by two cameras attached to the RVG asset, these are positioned and angled in a way that provides two different points of view for the RVG asset. The WebSocket then relays the information captured by these virtual cameras to the Web Server, where the hosted Web App 3D Render Stream utilizes it, see Figure 5.1. This Web App 3D Render Stream implementation allows visualizing RVG from the point of view of the cameras set in the scene via a media player-like widget.

5.3 2D Electronic Chart

In Chapter 4, an interface for accessing and processing real-time data stream coming from RVG was presented. Similarly, the Web Server was introduced in Section 5.1 as a method for relaying the data coming from the **Data Model** to other applications via WebSocket. In this chapter, we will explore a solution for visualizing data for vessel monitoring and decision support for safe navigation in a 2D Electronic chart.

The React.js framework[1] was chosen for the development of this 2D EC due to its relative ease of use and compatibility with most web browsers. This is an open-source JavaScript library originally developed by Meta for front-end development, and it is focused on creating component-based user interfaces. Additionally, the

[1] `https://react.dev/`

Pigeon Maps[2] library is used in combination with React to provide the 2D EC Web App with map visualization and interactive map tools.

5.3.1 Handling Data for 2D Visualization

The Data Model relays the information to the Web Server, the 2D EC Web App then subscribes to the Web Server via WebSocket and relays the information to the components in charge of displaying this data. The flow is depicted in Figure 5.4.

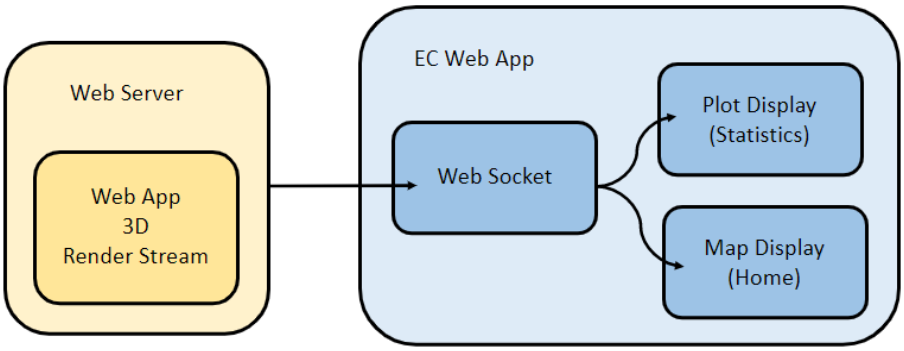


Figure 5.4: Data Flow from the Web Server to the 2D EC Web App.

The 2D EC Web App is handling the following data:

- NMEA messages from RVG communicating its coordinates and heading.
- AIS data related to the vessels surrounding RVG.
- ARPA data of vessels surrounding RVG that could become a safety concern.
- CBF data for safely maneuvering around the vessels which could pose a safety threat.

This was the information identified as necessary for carrying out the objectives of the 2D EC, however, any number of additional telemetry messages coming from RVG and relayed by the Data Model could be displayed in this 2D EC Web Application given a concise method for doing so, i.e.; plotting the data in a line plot.

5.3.2 User Interface

The following components are thus proposed for the User Interface (UI) of the 2D EC Web App:

- A Home page with a geographical map displaying the surrounding area of RVG along with all the information related to vessel monitoring and decision support.

[2] `https://pigeon-maps.js.org/docs/`

- A Statistics page with line plots for displaying the historical data of RVG telemetry being received by the Web Application.
- A Settings page with used selectable parameters for altering the behavior of the information on display on the Home page.
- An Explore page linking the Web App 3D Render Stream from the Web Server to this 2D EC Web App.

These pages will be made accessible and selectable via a side menu located on the left side of the browser window and the contents of these pages will be displayed on the right side of the menu as exemplified in Figure 5.5.

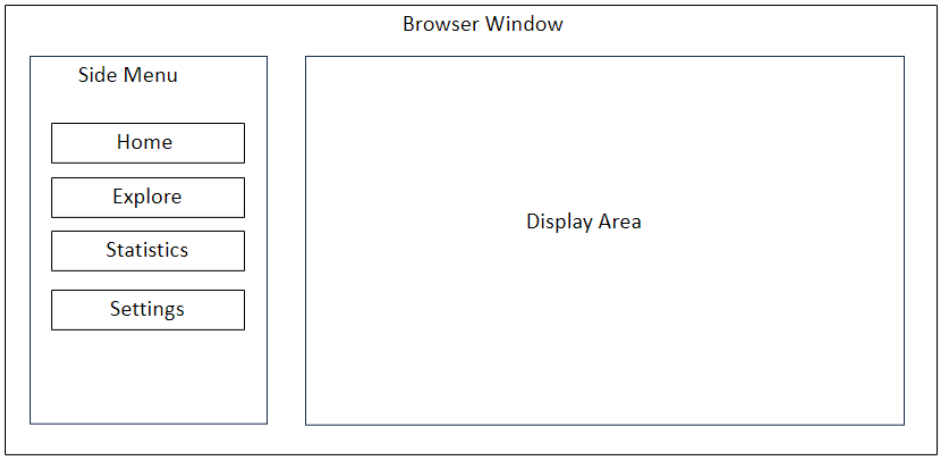


Figure 5.5: Proposed UI Layout for the 2D EC Web App

5.3.3 RVG Data Display

The NMEA message GPGGA, and PSIMSNS received from the Data Model are used to obtain the information related to RVG's coordinates and heading. These coordinates are then displayed on the map on the Home page by providing a marker on the map positioned at the coordinates of RVG. The heading information, on the other hand, is used to display an image of a red vessel overlayed onto the map at the coordinates of RVG. This overlayed image is rotated to match the heading of RVG. Furthermore, the marker, as well as the vessel image, are scaled to fit the zoom level of the map itself. This is done in order to prevent these visual items from becoming disproportionately small when zooming away to get a broader picture of the area surrounding RVG or disproportionately big when zooming in to get a more detailed picture, this behavior is also present on the representation for AIS vessels. An example of these components and their behavior can be seen in Figure 5.6.

Figure 5.6: 2D Representation of RVG stationed in Trondheim Harbor at two different zoom levels, the image on the left is zoomed away from RVG, while the image on the right is zoomed in.

5.3.4 AIS Vessels Data Display

Similar to the way data is obtained for the display of RVG in the map, the information for the display of AIS vessels is obtained from the AIS messages received from the Data Model. The data received for individual AIS vessels is stored in a list for their display. The data received for the display of the AIS vessels consists of the following elements:

- The Maritime Mobile Service Identity (MMSI) of the vessel.
- Current coordinates of the vessel.
- Speed over ground, course, and heading.
- A history containing the previous coordinates of the vessel.
- A predicted coordinate of the position of the vessel in the next 60 seconds.

The display of the AIS vessels behaves in the same fashion as that for RVG, differing in the fact that the image for the AIS vessel is green and is not shown when the ship is stationary in order to prevent cluttering the UI. There are also a few additions to the representation of these vessels in the UI, for example, the historical data on the previous coordinates for the vessel is used to draw a blue trail behind the vessel during transit; this allows the user to visualize the trajectory the ship has followed. Similarly, the predicted position for the AIS vessel is drawn in orange by using the current information on its course and speed over ground. These visual items can be seen in Figure 5.7 (left).

By hovering over or clicking the marker for the AIS vessel, a popup appears with relevant information. This includes its MMSI, longitude, latitude, course, speed over ground, and ARPA parameters if available, see Figure 5.7 (right).

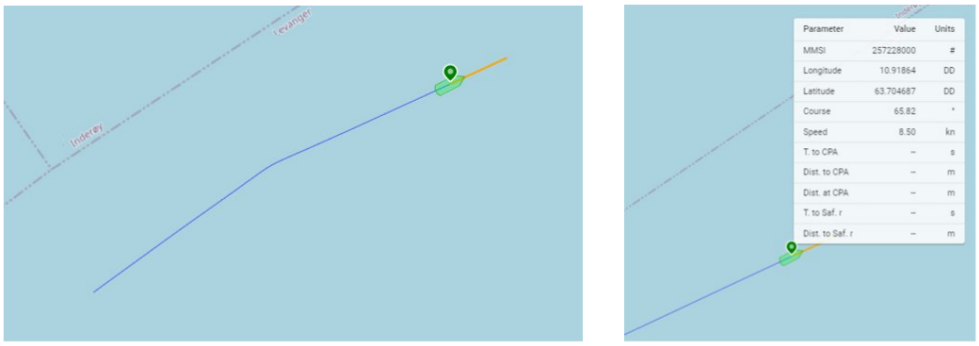


Figure 5.7: 2D representation of an AIS vessel during transit in the Trondheim Fjord. The image on the left shows the blue trail drawn behind the vessel as it moves and the orange line shows its predicted position. The image on the right shows the popup with the information on the vessel.

5.3.5 Plot Display for Incoming Data

As stated in Section 5.3.2, plots for incoming data related to RVG can be visualized in the **Statistics** page and will be accessible from the menu. This page will simply consist of line plots of time-labeled data describing some of the parameters from RVG. At the moment of writing, the data displayed on this page is the longitude, latitude, altitude, heading, pitch, and roll of RVG. An example of one of these plots can be observed in Figure 5.8.

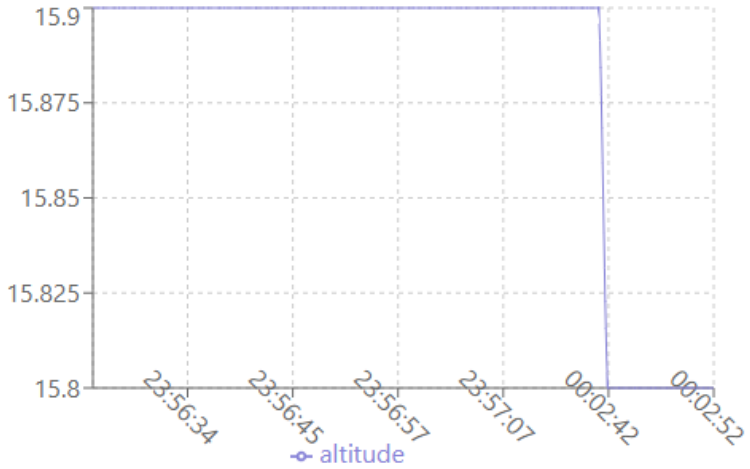


Figure 5.8: Line plot for RVG heading. The vertical axis represents the value of the altitude in degrees, while the horizontal axis is labeled with the timestamp for the received data.

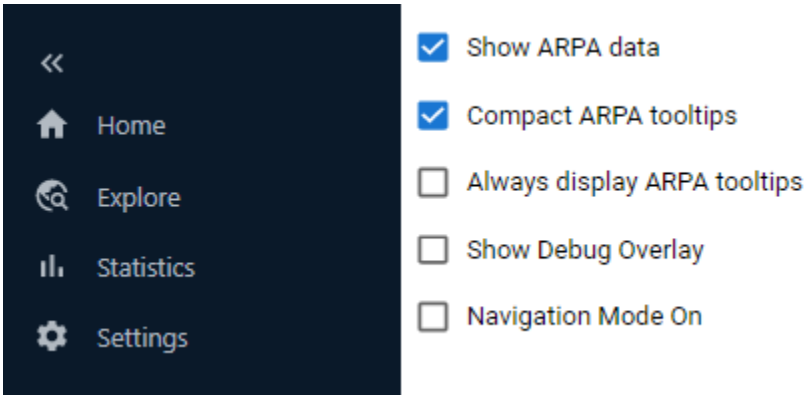


Figure 5.9: Settings page from the 2D EC Web App.

5.3.6 User Settings for Data Display

As mentioned in Section 5.3.2, the web application for the 2D EC will include a settings page (see Figure 5.9) for the user to modify some of the display parameters:

- Show ARPA data: when not checked, the ARPA parameters display in Section 5.4.1 will be hidden.
- Always display ARPA tooltips: when checked, the pop-ups containing ARPA data for vessels identified as a threat will be automatically displayed with the rest of the visual items. Otherwise, the user will have to mouse over the marker for a target vessel for the pop-up to show. This is done in order to prevent cluttering the UI.
- Show Debug Overlay: Displays a draggable text area that displays raw information related to AIS vessels. The marker for the vessel must be clicked in order to fill the text area with the related data. Used for debugging purposes.
- Navigation Mode On: When checked, the display of the map changes its point of view to a radar-style UI where RVG is centered on the display and pointing upwards, while the rest of the map rotates around it as RVG's heading changes (body-fixed perspective). When this is not checked, the default perspective of the map has the North at the top of the display and RVG rotates relative to this (world-fixed perspective).

5.4 Display of DSS for Safe Navigation

The 2D EC is in charge of displaying information regarding the DSS for Safe navigation. As mentioned in Section 3.3 from Chapter 3, this will consist of the ARPA parameters for the relevant AIS vessels as well as the CBF-based guidance for safe maneuvering. The data used for the display of these elements will come from the

navigational support components described in Section 4.4 in Chapter 4.

5.4.1 Display of ARPA Parameters on the 2D EC

Following the description of the ARPA parameters presented in Section 3.3.1 from Chapter 3. This information can be seen displayed in Figure 5.10, where the related parameters are highlighted with colored arrows. Their display is as follows:

- Closest Point of approach (CPA, black arrow): This will be represented by displaying both, RVG and the target vessel at the CPA with slightly translucent markers on the same color.
- Distance to Closest (D2CPA, white arrow): This will be displayed as a gray line connecting the translucent images for RVG and its translucent image at the CPA.
- Time to Closest Point of approach (T2CPA, orange arrow): this information will be displayed in a popup next to the target vessel.
- Distance at Closest Point of approach (D@CPA, pink arrow): This will be displayed as a gray line connecting the translucent images for RVG and the target vessel at the CPA.
- Safety Radius (R, purple arrow): This will be displayed as a gray circle with its center in the location where the target vessel will be when RVG crosses the Safety Radius. Additionally, a yellow translucent image of a vessel will be used to represent the point at which RVG will intersect with the Safety Radius.
- Distance to Safety Radius (D2R, blue arrow): this will be displayed as a gray line connecting the current position of RVG and the yellow translucent image representing RVG intersecting the safety radius.
- Time to Safety Radius (T2R, green arrow): this information will be displayed in a popup next to the target vessel.

5.4.2 Display of CBF-based Guidance for Safe Maneuvering on the 2D EC

The CBF-based Guidance for Safe Maneuvering consists of two elements, a red line depicting the proposed safe trajectory for RVG, and a pop-up displaying the TTEM. This information comes from the computations described in Section 4.4.2. The time left until the maneuver corresponds to the time in the computed simulation where RVG had to steer away from its desired course in order to avoid an obstacle, this event is timestamped and kept track of in the UI while a suggested evasive trajectory is on display. It is worth mentioning that the proposed trajectory assumes only changes in the orientation of RVG, so that, in theory, this maneuver will be achievable by steering RVG towards the proposed path without having to slow down or speed up. However, since the simulation is not taking the dynamics of RVG into consideration, this path will most likely be impossible to follow as RVG could drift away from the path when turning. Nevertheless, this trajectory

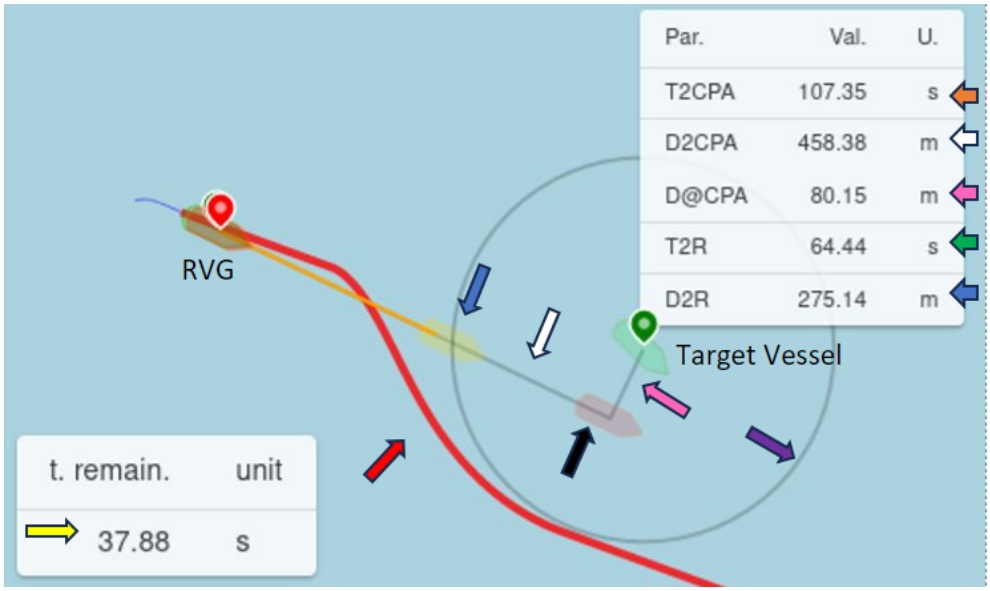


Figure 5.10: Display of navigational support information, ARPA, and CBF parameters, on the Electronic Chart. Black arrow, CPA; white arrow, D2CPA; orange arrow, T2CPA; pink arrow, D@CPA; purple arrow, R; blue arrow, D2R; green arrow, T2R; red arrow, suggested trajectory; yellow arrow, TTEM. Note that TTEM is displayed with a different name in the UI.

serves as a suggestion for the OOW to steer into in order to avoid collisions and maintain their course, even if it is not possible to stick perfectly to the suggested path.

The pop-up displaying the TTEM can be seen in Figure 5.10 highlighted with a yellow arrow, while the proposed safe trajectory is highlighted with a red arrow.

Chapter 6

Demonstration of RVG Online Digital Twin

The results of the project will be presented in this chapter followed by a brief discussion of each of the main topics. The experimental setup for these results consists of the following:

- A computer for running the scripts.
- Network access to the TCP datastream from RVG. This can be achieved by using NTNU's network at any of the campuses, with an NTNU VPN, or by connecting directly to RVG's network while onboard (Chapter 4).
- The script for the Real-Time Data Interface either connected to Real-Time data or replaying a previously saved log (Chapter 4).
- The script for the Web Server and 3D online Data Visualization (Chapter 5).
- Unity editor loaded with the corresponding Real-Time Scene (Chapter 5).
- The script for the 2D EC Web App and a web Browser for its visualization (Chapter 5).

The Online DT for RVG is ready once these components are running. Demonstrations for the implementation of the elements proposed in Chapters 4, and 5 will be presented in the following sections.

6.1 Demonstration of Real-Time Data Communication and Processing

The interface for Real-Time Data Communication and Processing in Chapter 4 was implemented in Python with its environment being managed by Conda. We shall focus on the capabilities of the Data Model for Relaying processed data to external Applications i.e.; the 2D EC Web App. and its capabilities for storing relevant data and using it as a data source for replaying past streams.

6.1.1 Data Storage and Replay

Data from RVG was stored during field tests carried out on October 27, 2022. During these tests, the Data Model was connected directly to RVG's onboard network and used to store RVG data at increasing user-defined intervals to test its capabilities. The output for each of these tests was the following:

- A .txt file containing the raw data stream as received by the Data Model directly from RVG's onboard network. The names for these saved files are automatically generated by the Data Model in the following format:

  datastream_{MMddyy}_{hh-mm-ss}_{duration}s.txt

  Where the fields within the curly braces are automatically filled with the date, time, and duration of the saved logs. For example, the file:

  datastream_102722_11-41-43_1200s.txt

  will correspond to a log saved on October 27, 2022, at 11:41:43, with a total duration of 1200 seconds.
- A folder with the same name convention as the raw text file, containing the individual .csv files containing telemetry data from RVG's NMEA messages.

Data corresponding to 600, 1200, and 3600 seconds intervals was saved at different points of the field test. The .csv files resulting from saving RVG telemetry data can be used to visualize information regarding the test, examples for these plots can be found in Appendix A.3.

Furthermore, the raw data from the .txt files can be fed into the Data Model by choosing it from a dialog when prompted or by passing it as an argument when running the Data Model Script. This allows for the raw .txt data to be processed and visualized in 2D and 3D with the corresponding Apps, i.e.; 2D EC, Autoferry Gemini, and 3D Online Visualizer. This replay feature is used in Section 6.3 for demonstrating the Implementation of Decision Support for Safe Navigation. The duration of these replays matches the duration of the recording itself, indicating that the data is being relayed to external applications with the timing indicated in the timestamps of the recorded data. Additionally, this feature has been used to demo the 2D EC in a TV at NTNU Campus Moholt. It is worth noting that the replay can be either set to run once, or in a loop.

6.1.2 Relaying Data to External Apps

An important consideration for real-time applications is latency. In other words, it is important to know how much time it will take between data being captured by the telemetry equipment aboard RVG and this data being received by the frontend applications. Therefore, the latency for the data transmission has been measured by comparing the timestamp of the messages being received by the 2D EC Web App with the current time at their reception.

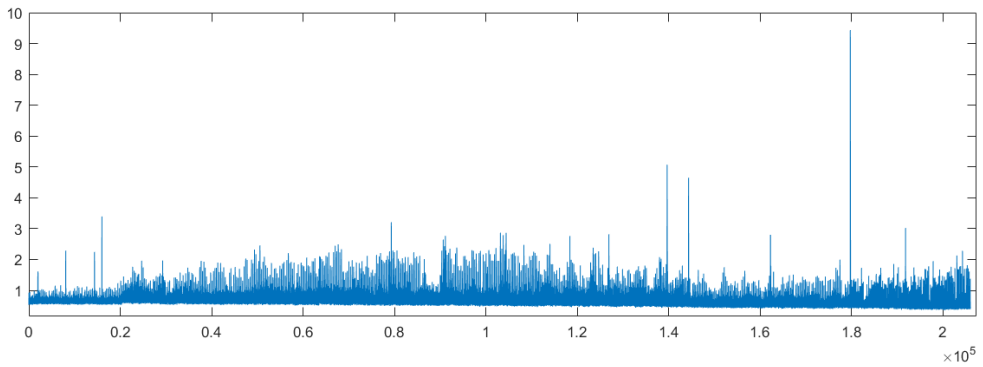


Figure 6.1: Plot of latency for data received in the Web App. The y-axis is the latency in seconds, the x-axis is the number of messages received

The resulting latency for the data Received in the Web Application was measured for 12 hours, resulting in the data shown in Figure 6.1. The latency was found to have a median of 0.6310 seconds, a minimum latency of 0.3580 seconds, a maximum latency of 9.4330 seconds, and a standard deviation of 0.2433 seconds.

6.1.3 Discussion

The data collected and processed by the Data Model is being relayed in a timely fashion to external apps. Despite some spikes in the latency, this seems to be kept relatively low. This is sufficiently responsive for a real-time implementation for the decision support of a vessel. Considering that the cruising speed of RVG is 9.4 knots, split-second updates, and computations should not be strictly necessary.

The responsiveness of the Data Model comes with the caveat that the Fast Logger (Chapter 4, Section 4.2.3) is not capable of storing data in separate Data Frames. It is possible, however, to save the raw data in a real-time test, and use the raw data later to create the separate data frames.

Additionally, the replay feature appears to be a useful way to demo the capabilities of the project. It should be noted that all the features available in the DM have to be set up or otherwise selected at the moment of launching the script.

6.2 Demonstration of Vessel Monitoring and Data Visualization

2D and 3D visualization for RVG is available for both, real-time data and replay data. This is achieved by relaying the processed data from the Data Model to the External Applications.

6.2.1 3D Visualization Demonstration

Using the Unity Editor allows for freely visualizing the scene from any perspective and zoom level. We will use this feature to get an eagle-eye perspective of RVG and the surrounding area as the 3D visualization is updated with the incoming data. In Figure 6.2 (left) we can see a zoomed-out eagle-eye perspective of RVG approaching Trondheim Harbor after a trip, the 2D EC on the same Figure (right) serves as a reference. We can notice how, due to the scale of RVG, it becomes difficult to spot when zooming out on the Unity Editor. We can also notice how the Trondheim harbor (bottom right in both images) is the only land mass being rendered in the scene, however, this is the only modeled land mass available for the 3D visualization. By comparing both images, we can notice how their position relative to the Harbor remains the same. However, AIS vessels prove difficult to frame and visualize in this 3D scene due to the sheer distance between the vessels and their relatively small scale. A close-up of the AIS vessels rendered in the 3D scene can be seen in Figure 6.9 (left).

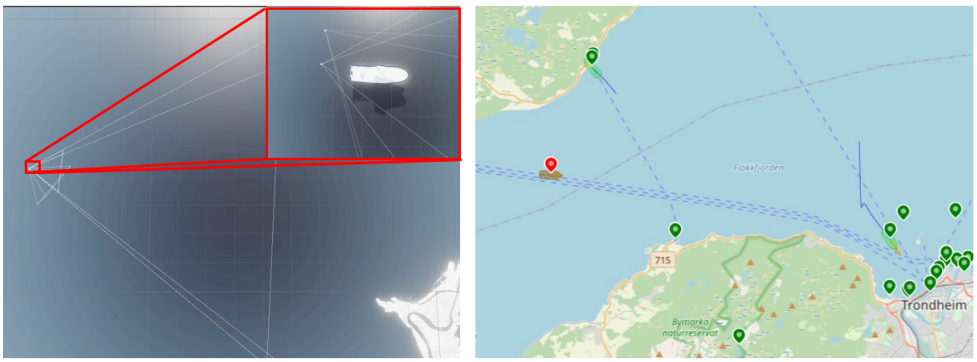


Figure 6.2: Unity Editor 3D visualization of RVG approaching Trondheim Harbor (left), and the corresponding view in the 2D EC Web App (right).

In Figure 6.3 (left) we can see an eagle-eyed perspective of RVG now docked in the harbor, the 2D EC chart on the same Figure (right) serves again as a reference. The 3D visualization in this Figure presents more detail compared to the previous scene. Here we can see RVG docked in its usual location highlighted in red at the bottom. Similarly, scale now allows for visualizing some of the AIS vessels in the vicinity of RVG highlighted in red at the top of the image. The 3D model of the Trondheim Harbor provides now a visual representation of the landmass surrounding RVG.

In addition to the eagle-eye perspective available when using the Unity Editor there is the in-game perspective normally used when executing a Unity program. This perspective is bound to virtual cameras set in the 3D scene bound to specific assets, in this case, two virtual cameras pointing at RVG from different angles have been set up. In Figure 6.4 (left) we can see the in-game perspective RVG when returning to Trondheim harbor (same scenario as Figure 6.2), the image in the

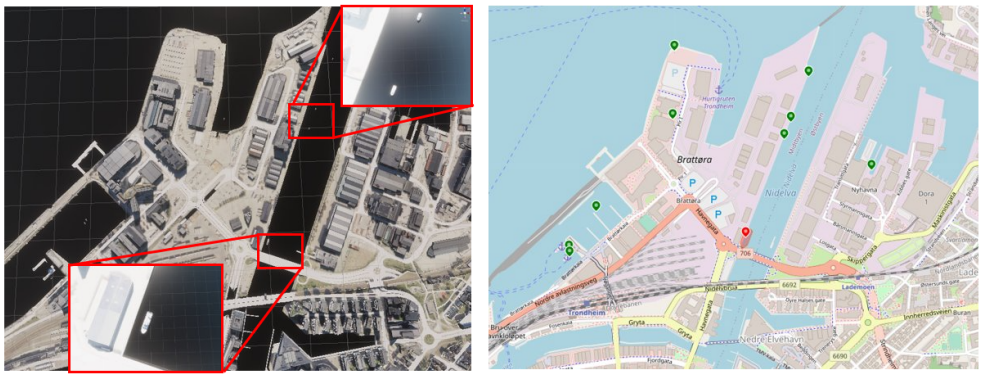


Figure 6.3: Unity Editor 3D visualization of RVG docked Trondheim Harbor (left), and the corresponding view in the 2D EC Web App (right)

right shows the in-game perspective of RVG docked in the harbor (same scenario as Figure 6.3). This in-game perspective comprises the stream of the 3D Online visualization, with a main and secondary perspective of RVG. The images in Figure 6.4 come from the Web Application for 3D Online Visualization.

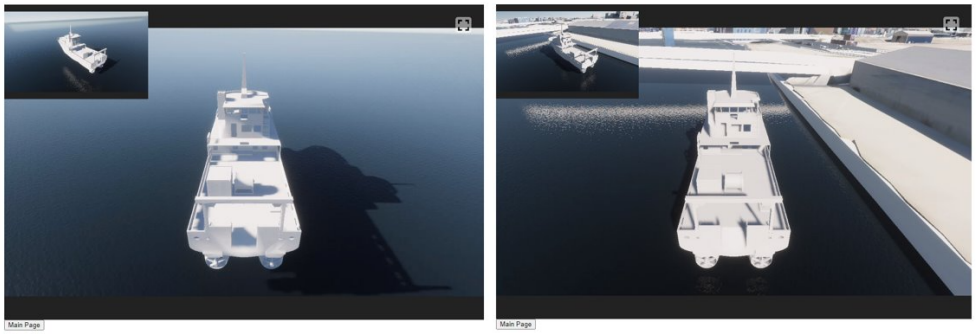


Figure 6.4: 3D Online visualization of RVG approaching Trondheim Harbor (left), and 3D Online visualization of RVG docked in Trondheim Harbor (right).

6.2.2 Discussion for 3D Visualization

Following the implementation of the 3D visualization, some issues were found. First, since environmental conditions besides light are not part of the 3D Visualization; the vessel can sometimes be located at odd positions relative to the rendered sea level, either levitating above the water or submerged. This is due to current sea levels (tides and waves) not being part of the simulated environment, and the incoming data for RVG's altitude. However, this can be remediated by ignoring the z reference for RVG.

Another issue noticed is that the simulated 3D area has fixed dimensions, therefore if the vessel travels to a point far away from the origin of the simulation it will end up being rendered in a void. Not to mention that the harbor

area is the only geographical location currently available for being rendered in the scene. That said, RVG is often seen surrounded by vast patches of the empty sea, which doesn't appear to convey much information other than the attitude and position of the vessel.

An online DT for RVG is rendered in a 3D scene along with the AIS vessels, however, the visualization will benefit from having different assets for the different AIS vessels in order to more accurately render them into the scene.

6.2.3 2D EC Demonstration

The resulting implementation for the 2D EC is the Web Application containing the features presented in Sections 5.3, and 5.4 from Chapter 5. The Home Page for the 2D EC Web Application can be seen in Figure 6.5, displaying the map with RVG (red marker with red vessel image) in the center and the surrounding AIS vessels (green markers). The menu for navigating the application and accessing different features can be seen left of the map.

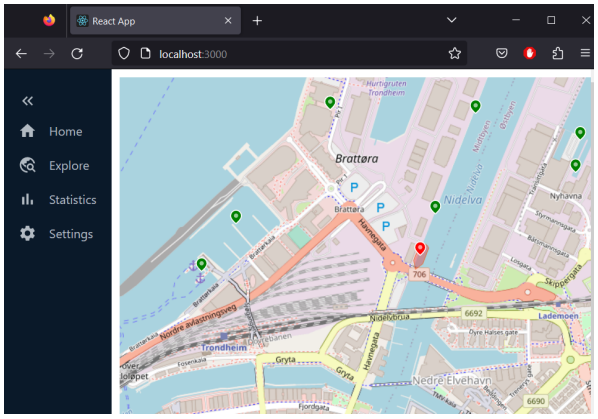


Figure 6.5: Screen capture of web browser displaying the 2D EC Web App.

The historic data for the positions of AIS vessels was embedded in the 2D EC as presented in the previous chapter. Similarly, the predicted position of the vessel is displayed as an orange line marking where the vessel will be 60 seconds in the future assuming a constant course and speed. See Figure 6.9 (right). In Figure 6.6, we can see a comparison between the Setting for Navigational Mode. The body-fixed perspective is shown in the image on the left (Navigation Mode on), while the image on the right corresponds to the world-fixed perspective (Navigation Mode off).

In Figure 6.7 we can see the difference between the setting for Compact ARPA Tooltip: off (left), and Compact ARPA Tooltip: on (right). The image on the left contains additional information on the vessel not necessarily related to the ARPA parameters, i.e.; MMSI, longitude, latitude, course, and speed over ground. The Always Display ARPA Tooltips setting controls whether these tooltips are automatically shown when a vessel's D@CPA is shorter than the safety radius R plus

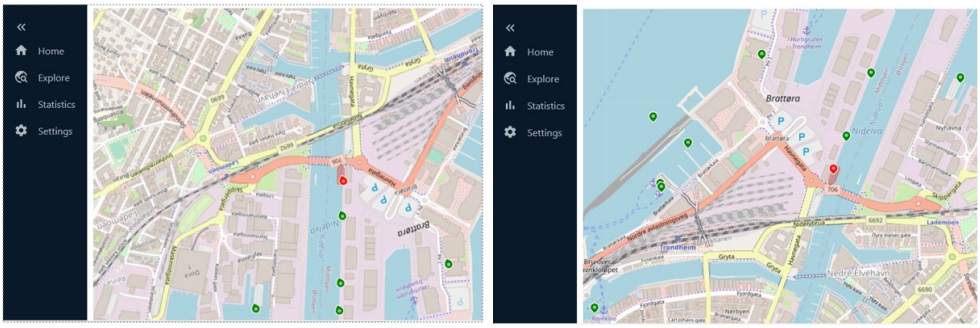


Figure 6.6: Comparison between Navigation Mode: on (left) and Navigation Mode: off (right).

tolerance R_{tol} (Section 4.4.1, Chapter 4). If this setting is unchecked, the user has to manually hover over the respective marker or click on it to show its ARPA Tooltip. The Settings page itself remains the same as in Figure 5.9 from the previous chapter.

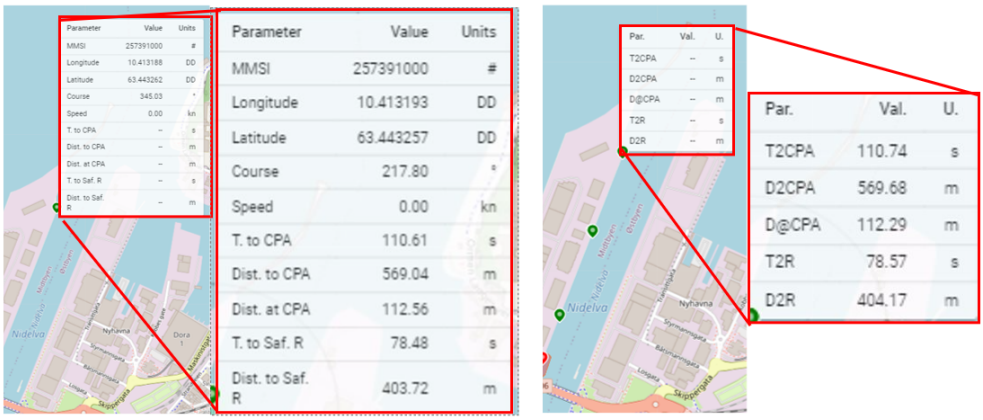


Figure 6.7: Comparison between extended ARPA tooltip (left), and compact ARPA tooltip (right). In both images, the red highlight is used to provide a closer look at either version of the pop-up.

In Figure 6.8 we can see a snippet from the Statistics page with a line plot of the longitude part of RVG's coordinates. Otherwise, A full view of the Statistics Page can be found in Appendix A.4.

6.2.4 Discussion for 2D EC

The 2D EC Web App allows for a meaningful visualization of RVG, its location, and the surrounding vessels. The 2D EC provides multiple ways for accessing and visualizing information for RVG monitoring. That said, the Data Server is yet to be made public in order to allow users with security clearance to access the web application online.

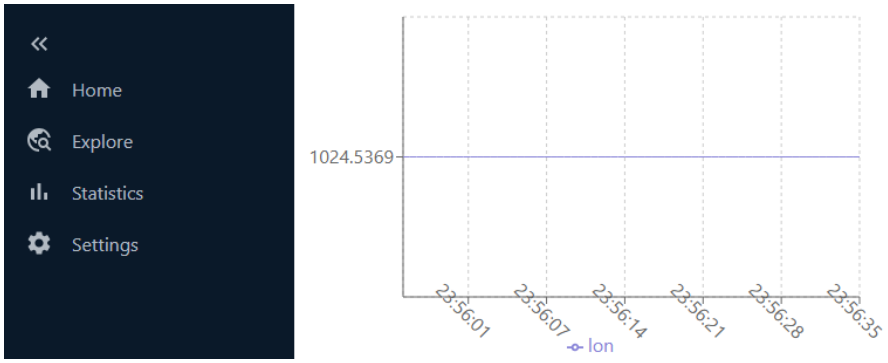


Figure 6.8: Snippet from the Statistics page in the 2D EC Web App.

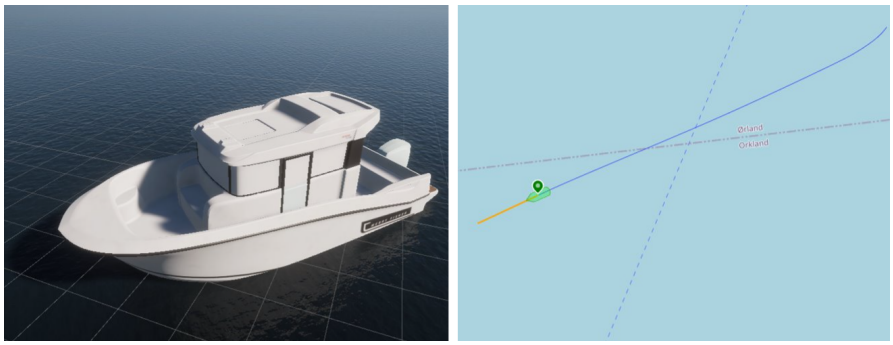


Figure 6.9: Close up of AIS vessel in the 3D Visualizer (left). Detail of AIS vessel in the 2D EC (right). Position history for the AIS vessel is shown as a blue path in the 2D EC, similarly, the predicted position is drawn with an orange line.

Wildpoints in the data are still an issue despite the data being filtered, as can be seen in Figure 6.10, these should be addressed in order to more clearly convey information to the OOW.

6.3 Demonstration of DSS for Safe Navigation

The resulting implementation for the display of ARPA parameters and CBF-based guidance for safe maneuvering on the 2D EC will be presented in this section. These demonstrations will be based on previous logs collected from RVG's field trip from Section 6.1.1, as it allows us to visualize RVG in motion. A signal for a non-existent AIS vessel will be spoofed[1] in order to make the Data Model think there is a vessel in RVG's path, make it obtain the related ARPA parameters, and propose a CBF-based trajectory for safely maneuvering around it. For the ARPA parameters, a safety Radius $R = 200$m and a safety Radius tolerance $R_{tolerance} = 100$m were used. Similarly, the following parameters were used for the CBF computations

[1]Send a fake signal to deceive a receiver. In this case, the fake signal will be embedded in the Data Model

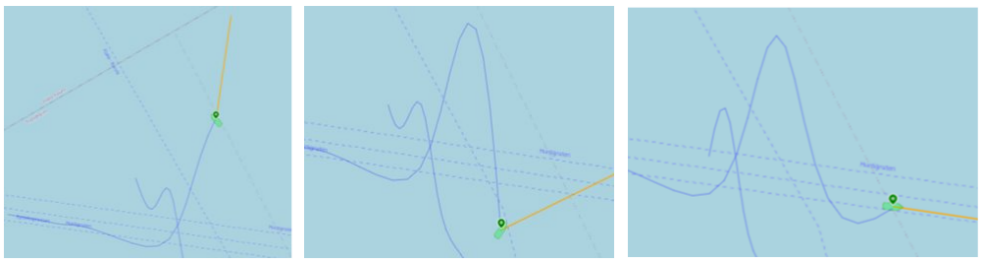


Figure 6.10: Wildpoints on the signal for an AIS vessel, coordinates, course, and heading appear to be jittering at different points of the AIS vessel trip. We can see how the filter tries to smooth out the blue path behind the vessel.

$t_1 = 0.2\,\text{s}$, $t_2 = 40\,\text{s}$, $k_p = 1$, $\lambda = 0.5$, $\Delta t = 0.2\,\text{s}$ and a total simulation time of 600 s. The value for the reference course z_{ref} is set as the heading of RVG at the moment of starting the calculations for the CBF.

The resulting display for the Decision Support for Safe Navigation can be observed in Figures 6.11 and 6.12. These images were captured a few seconds apart. The parameters for ARPA, as well as the CBF trajectory, are recalculated every 10 seconds, updating the information on the 2D EC Web Application each time.

In Figure 6.11 we can see RVG approaching the target vessel with the following ARPA parameters: T2CPA = 107.35 s, D2CPA = 458.38 m, D@CPA = 80.15 m, T2R = 64.44 s, and D2R =275.14 m. The CBF computations result in a TTEM of 37.88 s, with the proposed trajectory shown in red

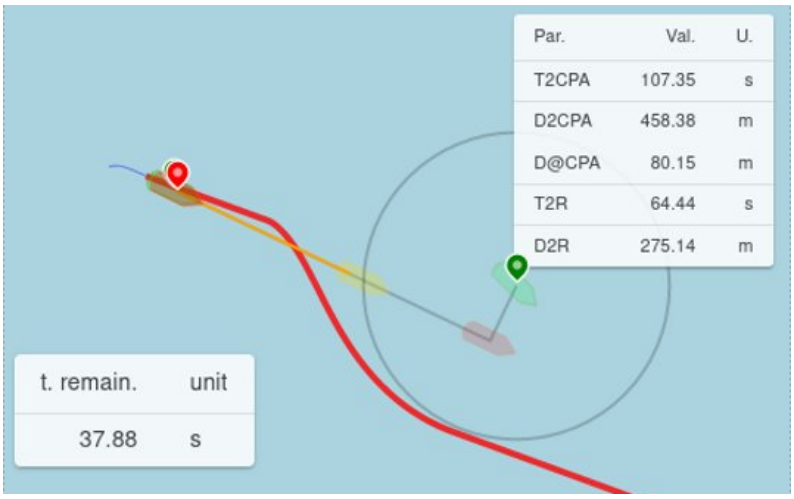


Figure 6.11: 2D EC Web App Display of the Decision Support for Safe Navigation Information for RVG approaching a target vessel.

In Figure 6.12 we can see RVG now closer to the target vessel with the new ARPA parameters:T2CPA = 83.64 s, D2CPA = 408.76 m, D@CPA = 89.91 m, T2R = 47.08 s, and D2R =230.10 m. The CBF has been recalculated resulting in a TTEM of 9.14 s, with the updated proposed trajectory shown in red.

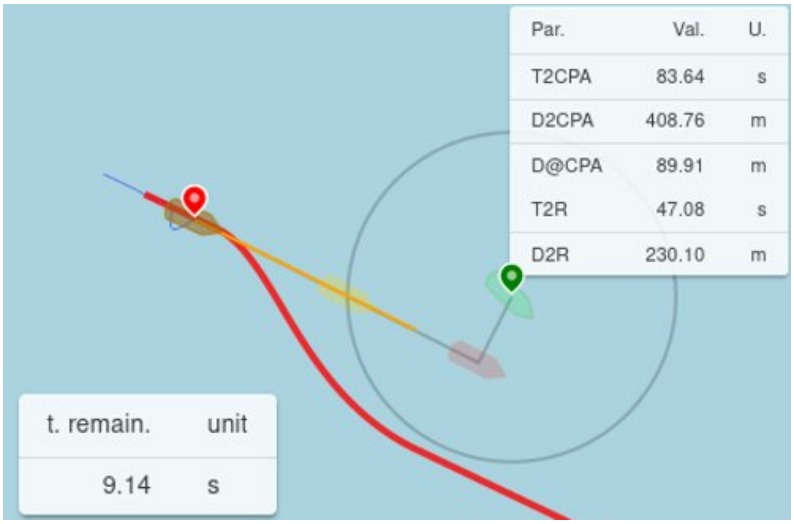


Figure 6.12: Updated 2D EC Web App Display of the Decision Support for Safe Navigation Information for RVG approaching a target vessel.

6.3.1 Discussion

Information for DSS for Safe Navigation appears to be conveyed to the OOW at an adequate pace by re-computing the parameters every 10 seconds. The visual representation for the ARPA parameters appears to convey information in a clear way, although it will be necessary to test this with the end user in order to ensure the data is being relayed in an understandable fashion.

The suggestion for the safe trajectory comes with a few caveats. First, the fact that the simulation for the CBF trajectory does not take the dynamics of RVG into consideration should be addressed in subsequent iterations of the project. It also must be noted that the proposed trajectory is not taking into consideration COLREGs, making the suggestion unfeasible. This can be addressed however by modifying the shape of the obstacle domain (the circle around the target vessel with radius R) with a polygonal domain as proposed in Marley, Skjetne, Gil et al. (2023).

It is also worth noting that, since the z_{ref} course reference is automatically set to the course of RVG at the start of the CBF computations, this could lead to some confusion when a new trajectory is computed during the execution of a previous one. This would still lead to a suggestion for RVG to safely maneuver around the obstacle, but resulting in a different course than that at the start of the evasive maneuver. However, this is to be expected, as the CBF-based safe trajectory proposed is reactive; meaning that evasive maneuvers will change or be proposed based on the trajectory of the obstacle relative to RVG's course. The proposed CBF trajectory is meant to be interpreted as a prediction for a safe trajectory rather than a control input for RVG.

Chapter 7

Conclusions

In this project, a method for interacting with Real-Time data incoming from RVG was achieved along with a method for visualizing the incoming datastream in a 2D Electronic Chart Web Application and a 3D environment based on Autoferry Gemini. Furthermore, information for Decision Support for Collision Avoidance can be visualized in the 2D EC by using ARPA parameters and CBF.

The Python-based backend, the Data Model is capable of accessing the datastream incoming from RVG, saving, processing the data, and relaying it to external applications in a timely fashion. In addition, this back end is able to replay the raw data from .txt files in order to visualize the replay in the external applications or extract its information and save it as individual data frames in .csv files

The 3D visualization based on Autoferry Gemini provides a 3D scene for monitoring the situation of RVG, the surrounding AIS vessels, and the Trondheim Harbor. Meanwhile, the 2D EC Web App provides the aforementioned monitoring capabilities, albeit in a simpler format. Additionally, the 2D EC provides the user or OOW with additional information related to the surrounding AIS vessels, such as a trail corresponding to the vessel's previous position and an indicator for its predicted position.

Furthermore, the 2D EC displays information related to the DSS for safe navigation by displaying the ARPA parameters of relevant AIS vessels, as well as a CBF-based trajectory suggestion for safely avoiding multiple target vessels. This information is meant to aid the OOW in making decisions on how to safely avoid obstacles by making changes in RVG's course without affecting its speed.

The work presented in this thesis addresses the objectives set in the project definition. That said, even though some shortcomings were identified during development, it was important to draw a line for delimiting the scope of this project. Thus, recommendations for future work will be presented in the following section.

7.1 Recommendations For Future Work

Even though the main objectives set for the project have been achieved, there is still room for improvement following the shortcomings discussed in the previous

chapter, these are:

- The Data Model cannot sort data into different frames for them to be exported into .csv files while providing real-time data to external applications.
- The Data Server is yet to be made public in order to access the 2D EC Web App and by extension the 3D Visualization.
- The dynamics of RVG are yet to be implemented in the calculations for CBF.
- The visualization for land masses is limited to the Trondheim Harbor in the 3D Visualizer.
- The AIS vessels displayed on the 3D visualization all have the same asset and do not directly relate to the actual dimensions of the vessel.
- Wild points on incoming data can add noise to the display of the information on the 2D EC.

It is also worth noting that CBFs used in this project do not take into consideration the dynamics of the vessel, nor COLREGs, as such, a more sophisticated model accounting for these parameters should be developed upon subsequent iterations. The implementation of the 2D EC Web App must also be validated by testing it on real scenarios and obtaining feedback on its user interface.

Following the shortcomings identified with the current implementation the following points are suggested as a continuation of the work presented in this thesis. These are sorted in order of priority:

1. Set up a public server that allows users to access the 2D EC Web App remotely. This server should also run the Unity Editor for 3D Visualization and Streaming.
2. Add a better mechanism to the Data Model for handling wild points and filtering data.
3. Include the dynamic properties of RVG in the CBF calculations.
4. Integrate COLREGs into the proposed trajectory for RVG by following the polygonal domain implementation in Marley, Skjetne, Gil et al. 2023.
5. Create a new thread for sorting and saving the data frames without interfering with the relay of incoming data for external applications.
6. Query the MMSI of the AIS vessels in order to obtain and better represent their dimension in both, the 2D EC and the 3D Visualizer.
7. Add more information related to land masses into the 3D visualization.

In addition to this, as an alternative to reading the real RVG datastream, a function that simulates RVG using a high-fidelity 4DOF maneuvering model of RVG could be implemented.

Bibliography

Polarkonsult (n.d.). *RV Gunnerus*. https://www.polarkonsult.no/rv-gunnerus. [Online; accessed 08-June-2023].

Rølvåg, T. and Ø. Stranden (2022). 'Digital Twin Based Structural Health Monitoring of Offshore Crane.' In: *41st Int. Conf. Ocean, Offshore Arctic Eng. (OMAE2022)* OMAE2022-85896.

Heyn, H.-M., G. Udjus and R. Skjetne (2017). 'Distributed motion sensing on ships'. In: *OCEANS*.

National Marine Electronics Association (2022). *NMEA 0183*. SiRF Technology Inc.

Kongsberg Maritime (2013). *Reference manual Kongsberg MDM 500 Marine Data Management*. Kongsberg Maritime.

National Marine Electronics Association (2007). *NMEA Reference Manual*. SiRF Technology Inc.

Tetreault, B.J. (2005). 'Use of the Automatic Identification System (AIS) for maritime domain awareness (MDA)'. In: *Proceedings of OCEANS 2005 MTS/IEEE*, 1590–1594 Vol. 2. DOI: 10.1109/OCEANS.2005.1639983.

International Telecommunications Union (Feb. 2014). *Technical characteristics for a universal shipborne automatic identification system using time division multiple access in the VHF maritime mobile band*. DOI: RecommendationM.1371-5.

Harati-Mokhtari, A., A. Wall, P. Brooks and J. Wang (2007). 'Automatic Identification System (AIS): Data Reliability and Human Error Implications'. In: *The Journal of Navigation* 60.3, pp. 373–389.

Soni, D. and A. Makwana (Apr. 2017). 'A SURVEY ON MQTT: A PROTOCOL OF INTERNET OF THINGS(IOT)'. In: *ICTPACT 2017*.

DNV GLAS (2020). *Technology Outlook 2030*. DNV.

DNV (2021). *Qualification and assurance of digital twins*. DNV.

Erikstad, S. (Mar. 2019). 'Designing Ship Digital Services'. In: *COMPIT ´19 – 18th Conference on Computer and IT Applications in the Maritime Industries*.

Erikstad, S. and K. Levander (2012). 'System based design of offshore support vessels'. In: *Proceedings 11th International Marine Design Conference—IMDC201*.

Bjørum, L. (2019). 'Development of a Digital Twin for Condition Monitoring, Focusing on Electrical Propulsion Systems for Marine Application'. In: *Master's thesis in Marine Technology, Faculty of Engineering, Department of Marine Technology*.

Zhang, H., G. Li, L. Hatledal, Y. Chu, A. Ellefsen, P. Han, P. Major, R. Skulstad, T. Wang and H. Hildre (2022). 'A Digital Twin of the Research Vessel Gunnerus for Lifecycle Services: Outlining Key Technologies'. In: *IEEE Robotics Automation Magazine*, pp. 2–15. DOI: 10.1109/MRA.2022.3217745.

Alvsaker, J. (2020). 'R/V Gunnerus Digital Twin Infrastructure'. In: *Master's thesis in Marine Technology, Faculty of Engineering, Department of Marine Technology*.

IMO (2018a). 'Maritime Safety Committee (MSC), 100th session, 3-7 December 2018'. In: *https://www.imo.org/en/MediaCentre/MeetingSummaries/Pages/MSC-100th-session.aspx*.

Smogeli, Ø. (2022). *2022 Autumn course TMR06 Autonomous Marine Systems*.

Thyri, E. and M. Breivik (May 2022). 'A domain-based and reactive COLAV method with a partially COLREGs-compliant domain for ASVs operating in confined waters'. In: *Field Robotics* 2, pp. 637–677. DOI: 10.55417/fr.2022022.

IMO (2018b). *Convention on the International Regulations for Preventing Collisions at Sea, 1972 Consolidated edition, 2018*. IMO.

Šmıd, A. (2017). 'Comparison of unity and unreal engine'. In: *Bachelors Thesis*.

Dosovitskiy, A., G. Ros, F. Codevilla, A. M. López and V. Koltun (2017). 'CARLA: An Open Urban Driving Simulator'. In: *CoRR* abs/1711.03938. arXiv: 1711.03938. URL: http://arxiv.org/abs/1711.03938.

Vasstein, K., E.F. Brekke, R. Mester and E. Eide (Nov. 2020). 'Autoferry Gemini: a real-time simulation platform for electromagnetic radiation sensors on autonomous ships'. In: *IOP Conference Series: Materials Science and Engineering* 929.1, p. 012032. DOI: 10.1088/1757-899X/929/1/012032. URL: https://dx.doi.org/10.1088/1757-899X/929/1/012032.

Lin, B. and C.-H. Huang (2006). 'COMPARISON BETWEEN ARPA RADAR AND AIS CHARACTERISTICS FOR VESSEL TRAFFIC SERVICES'. In: *Journal of Marine Science and Technology* 14.3. DOI: 10.51400/2709-6998.2072. URL: https://jmstt.ntou.edu.tw/journal/vol14/iss3.

Lenart, A. (2017). 'Approach Parameters in Marine Navigation - Graphical Interpretations'. In: *TransNav, the International Journal on Marine Navigation and Safety of Sea Transportation* 11.3, pp. 521–529. ISSN: 2083-6473. DOI: 10.12716/1001.11.03.19. URL: ./Article_Approach_Parameters_in_Marine_Navigation_Lenart,43,756.html.

Lenart, A. (Sept. 1983). 'Collision Threat Parameters for a new Radar Display and Plot Technique'. In: *Journal of Navigation* 36, pp. 404–410. DOI: 10.1017/S0373463300039758.

Ames, A.D., S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath and P. Tabuada (2019). *Control Barrier Functions: Theory and Applications*. arXiv: 1903.11199 [cs.SY].

Marley, M., R. Skjetne, M. Breivik and C. Fleischer (Nov. 2020). 'A hybrid kinematic controller for resilient obstacle avoidance of autonomous ships'. In: *IOP Conference Series: Materials Science and Engineering* 929.1, p. 012022. DOI: 10.1088/1757-899X/929/1/012022. URL: https://dx.doi.org/10.1088/1757-899X/929/1/012022.

Marley, M., R. Skjetne, E. Basso and A.R. Teel (2021). 'Maneuvering with safety guarantees using control barrier functions'. In: *IFAC-PapersOnLine* 54.16. 13th IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles CAMS 2021, pp. 370–377. ISSN: 2405-8963. DOI: https://doi.org/10.1016/j.ifacol.2021.10.118. URL: https://www.sciencedirect.com/science/article/pii/S2405896321015202.

Marley, M., R. Skjetne, M. Gil and P. Krata (2023). 'Four degree-of-freedom hydrodynamic maneuvering model of a small azipod-actuated ship with application to onboard decision support systems'. In: *IEEE Access*. DOI: 10.1109/ACCESS.2023.3284684.

Hofmann-Wellenhof, B., H. Lichtenegger and J. Collins (1997). *Global Positioning System: Theory and Practice*, p. 281. ISBN: ISBN 3-211-82839-7.

Skjetne, R. (Mar. 2023). *Technical note: CBF derivations for a unicycle model*.

Unity Technologies (2020). *Unity Render Streaming 1.2.3-preview*. https://docs.unity.cn/Packages/com.unity.renderstreaming@1.2/manual/en/overview.html. [Online; accessed 22-March-2023].

Appendix A

Additional Material

Additional material that does not fit in the main thesis but may still be relevant to share,

A.1 NMEA messages from Gunnerus

YCMTW: (YC) Transducer Temp. (MTW) Mean Temperature of Water
Format: $–MTW,x.x,C*hh<CR><LF> Format Description:

1. Temperature, degrees
2. Unit of Measurement, Celsius
3. Checksum

SDDBT: (SD) Depth Sounder. (DBT) Depth below transducer
Format: $–DBT,x.x,f,x.x,M,x.x,F*hh<CR><LF> Format Description:

1. Water depth, feet
2. f = feet
3. Water depth, meters
4. M = meters
5. Water depth, Fathoms
6. F = Fathoms
7. Checksum

SDDBS: (SD) Depth Sounder. (DBS) Below Surface
Format: $–DBS,x.x,f,x.x,M,x.x,F*hh<CR><LF> Format Description:

1. Water depth, feet
2. f = feet
3. Water depth, meters
4. M = meters
5. Water depth, Fathoms
6. F = Fathoms
7. Checksum

SDDPT: (SD) Depth Sounder. (DPT) Depth of Water
Format: $–DPT,x.x,x.x,x.x*hh<CR><LF> Format Description:

1. Water depth relative to transducer, meters
2. Offset from transducer, meters positive means distance from transducer to water line negative means distance from transducer to keel
3. Maximum range scale in use (NMEA 3.0 and above)
4. Checksum

PFEC: Propietary sentence
Format: $PFEC,„xxxxx,x,x,x,x,x,x,x*hh<CR><LF> Format Description: unknown

PSIMSSB: Propietary sentence, SSB SSBL position
Format: $PSIMSSB,hhmmss.ss,cc__,A,cc_,aa__,aa__,aa__,x.x,x.x,x.x,x.x,*hh<CR><LF> Format Description:

1. = empty or time of reception
2. cc__ = Tp code (Examples: B01, B33, B47)
3. A = Status, A for OK and V for not OK. When the measurement is not OK the Error code field contains the error code
4. cc__ = Error code. Empty or a three character error code
5. aa__ = Coordinate system. C for Cartesian, P for Polar and U for UTM
6. aa__ = Orientation. H for vessel head up, N for North and E for East
7. aa__ = SW filter. M for Measured, F for Filtered and P for Predicted
8. x.x = X coordinate. The Coordinate system and Orientation fields should be used to decode the X and Y coordinate field
9. x.x = Y coordinate. The Coordinate system and Orientation fields should be used to decode the X and Y coordinate fields
10. x.x =Depth in metres
11. x.x =Expected accuracy of the position
12. aa__ = Additional information. N for None, C for Compass and I for Inclinometer
13. x.x = First additional value. Empty, Tp compass or Tp x inclination
14. x.x = First additional value. Empty or Tp y inclination
15. * = checksum delimiter
16. hh = empty or checksum

GPGBS: (GP) Global Positioning System receiver. (GBS) GPS Satellite Fault Detection
Format: $–GBS,hhmmss.ss,x.x,x.x,x.x,x.x,x.x,x.x,x.x*hh<CR><LF> Format Description:

1. UTC time of the GGA or GNS fix associated with this sentence. hh is hours, mm is minutes, ss.ss is seconds
2. Expected 1-sigma error in latitude (meters)
3. Expected 1-sigma error in longitude (meters)
4. Expected 1-sigma error in altitude (meters)
5. ID of most likely failed satellite (1 to 138)

6. Probability of missed detection for most likely failed satellite
7. Estimate of bias in meters on most likely failed satellite
8. Standard deviation of bias estimate
9. Checksum

GPRMC: (GP) Global Positioning System receiver. (RMC) Recommended Minimum Navigation Information
This is one of the sentences commonly emitted by GPS units.
Format: $–RMC,hhmmss.ss,A,ddmm.mm,a,dddmm.mm,a,x.x,x.x,xxxx,x.x,a*hh<CR><LF>
Format Description:

1. UTC of position fix, hh is hours, mm is minutes, ss.ss is seconds.
2. Status, A = Valid, V = Warning
3. Latitude, dd is degrees. mm.mm is minutes.
4. N or S
5. Longitude, ddd is degrees. mm.mm is minutes.
6. E or W
7. Speed over ground, knots
8. Track made good, degrees true
9. Date, ddmmyy
10. Magnetic Variation, degrees
11. E or W
12. FAA mode indicator (NMEA 2.3 and later)
13. Nav Status (NMEA 4.1 and later) A=autonomous, D=differential, E=Estimated, M=Manual input mode N=not valid, S=Simulator, V = Valid
14. Checksum

IIMWV: (II) Integrated Instrumentation. (MWV) Wind Speed and Angle
Format: $–MWV,x.x,a,x.x,a*hh<CR><LF> Format Description:

1. Wind Angle, 0 to 359 degrees
2. Reference, R = Relative, T = True
3. Wind Speed
4. Wind Speed Units, K/M/
5. Status, A = Data Valid, V = Invalid
6. Checksum

GPVTG: (GP) Global Positioning System receiver. (VTG) Track Made Good and Ground Speed
This is one of the sentences commonly emitted by GPS units.
Format: $–VTG,x.x,T,x.x,M,x.x,N,x.x,K,m*hh<CR><LF> Format Description:

1. Course over ground, degrees True
2. T = True
3. Course over ground, degrees Magnetic
4. M = Magnetic
5. Speed over ground, knots
6. N = Knots

7. Speed over ground, km/hr
8. K = Kilometers Per Hour
9. FAA mode indicator (NMEA 2.3 and later)
10. Checksum

GPZDA: (GP) Global Positioning System receiver. (ZDA) Time and Date
This is one of the sentences commonly emitted by GPS units.
Format: $–ZDA,hhmmss.ss,xx,xx,xxxx,xx,xx*hh<CR><LF> Format Description:

1. UTC time (hours, minutes, seconds, may have fractional subseconds)
2. Day, 01 to 31
3. Month, 01 to 12
4. Year (4 digits)
5. Local zone description, 00 to +- 13 hours
6. Local zone minutes description, 00 to 59, apply same sign as local hours
7. Checksum

PSIMSNS: (SNS) sensor values
compliant telegram designed by Kongsberg Maritime. It holds sensor values from a acoustic positioning transceivers. This telegram is typically generated by a HiPAP or HPR system.
Format: $PSIMSNS,hhmmss.ss,c–c,xx__,xx__,x.x,x.x,x.x,x.x, , , , , ,*hh<CR><LF>
Format description

1. PSIMSNS = talker identifier and telegram identifier
2. hhmmss.ss = coordinated universal time (UTC) of position. The decimal fraction is optional
3. xx__ = Transceiver number
4. xx__ = Transducer number
5. x.x = The roll angle in degrees
6. x.x = The pitch angle in degrees
7. x.x= The heave in metres
8. x.x= The heading in degrees. A value between 0 and 360
9. = empty field. May be used in the future
10. = empty field. May be used in the future
11. = empty field. May be used in the future
12. = empty field. May be used in the future
13. *= checksum delimiter
14. hh= checksum

VDVBW: (VD) Velocity Sensor, Doppler, other/general (VBW) Dual Ground/Water Speed
Format: $–VBW,x.x,x.x,A,x.x,x.x,A,x.x,A,x.x,A*hh<CR><LF> Format Description:

1. Longitudinal water speed, "-" means astern, knots
2. Transverse water speed, "-" means port, knots
3. Status, A = Data Valid
4. Longitudinal ground speed, "-" means astern, knots

5. Transverse ground speed, "-" means port, knots
6. Status, A = Data Valid
7. Stern traverse water speed, knots *NMEA 3 and above)
8. Status, stern traverse water speed A = Valid (NMEA 3 and above)
9. Stern traverse ground speed, knots *NMEA 3 and above)
10. Status, stern ground speed A = Valid (NMEA 3 and above)
11. Checksum

GPGGA: (GP) Global Positioning System receiver. (GGA) Global Positioning System Fix Data
Format: $–GGA,hhmmss.ss,ddmm.mm,a,ddmm.mm,a,x,xx,x.x,x.x,M,x.x,M,x.x,xxxx*hh<CR><LF>
Format Description:

1. UTC of this position report, hh is hours, mm is minutes, ss.ss is seconds.
2. Latitude, dd is degrees, mm.mm is minutes
3. N or S (North or South)
4. Longitude, dd is degrees, mm.mm is minutes
5. E or W (East or West)
6. GPS Quality Indicator (non null)
7. Number of satellites in use, 00 - 12
8. Horizontal Dilution of precision (meters)
9. Antenna Altitude above/below mean-sea-level (geoid) (in meters)
10. Units of antenna altitude, meters
11. Geoidal separation, the difference between the WGS-84 earth ellipsoid and mean-sea-level (geoid), "-" means mean-sea-level below ellipsoid
12. Units of geoidal separation, meters
13. Age of differential GPS data, time in seconds since last SC104 type 1 or 9 update, null field when DGPS is not used
14. Differential reference station ID, 0000-1023
15. Checksum

VDVLW : (VD) Velocity Sensor, Doppler, other/general. (VLW) Distance Traveled through Water
Format: $–VLW,x.x,N,x.x,N,x.x,N,x.x,N*hh<CR><LF> Format Description:

1. Total cumulative water distance, nm
2. N = Nautical Miles
3. Water distance since Reset, nm
4. N = Nautical Miles
5. Total cumulative ground distance, nm (NMEA 3 and above)
6. N = Nautical Miles (NMEA 3 and above)
7. Ground distance since reset, nm (NMEA 3 and above)
8. N = Nautical Miles (NMEA 3 and above)
9. Checksum

RAOSD: (RA) RADAR and/or ARPA. (OSD) Own Ship Data
Format: $–OSD,x.x,A,x.x,a,x.x,a,x.x,x.x,a*hh<CR><LF>*hh<CR><LF> Format

Description:

1. Heading, degrees True
2. Status, A = Data Valid, V = Invalid
3. Vessel Course, degrees True
4. Course Reference B/M/W/R/P
5. Vessel Speed
6. Speed Reference B/M/W/R/P
7. Vessel Set, degrees True
8. Vessel drift (speed)
9. Speed Units K/N
10. Checksum

RATTM: (RA) RADAR and/or ARPA. (TTM) Tracked Target Message
Format: $–TTM,xx,x.x,x.x,a,x.x,x.x,a,x.x,x.x,a,c–c,a,a,hhmmss.ss,a*hh<CR><LF>
Format Description:

1. Target Number (0-99)
2. Target Distance
3. Bearing from own ship
4. T = True, R = Relative
5. Target Speed
6. Target Course
7. T = True, R = Relative
8. Distance of closest-point-of-approach
9. Time until closest-point-of-approach "-" means increasing
10. Speed/distance units, K/N
11. Target name
12. Target Status
13. Reference Target
14. UTC of data (NMEA 3 and above) hh is hours, mm is minutes, ss.ss is seconds.
15. Type, A = Auto, M = Manual, R = Reported (NMEA 3 and above)
16. Checksum

A.2 Gunnerus coordinate reference points

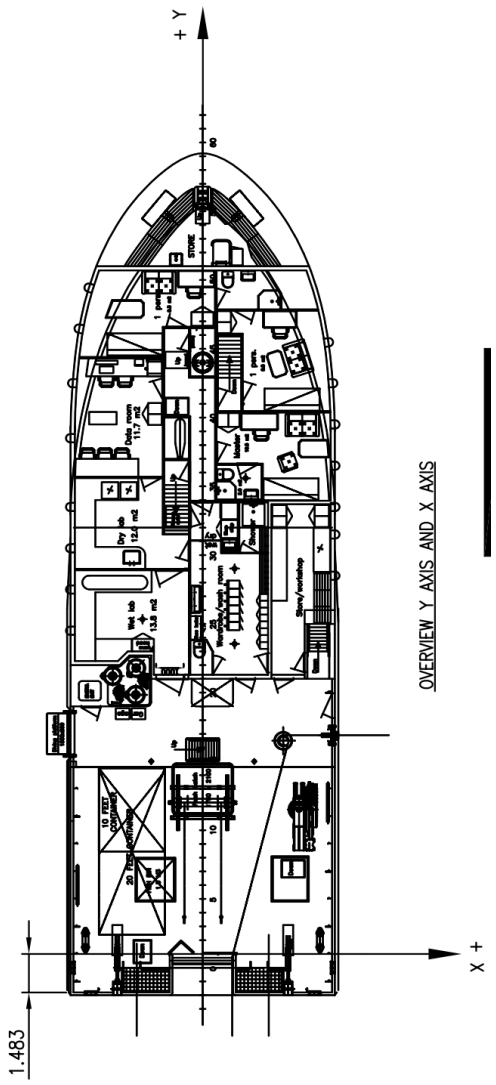


Figure A.1: Y X view of RVG (before extension)

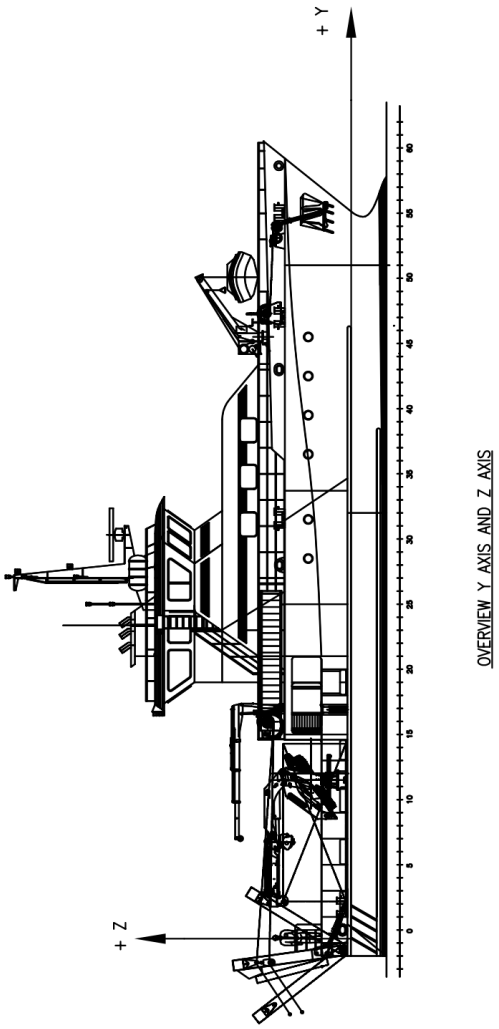


Figure A.2: Y Z view of RVG (before the extension)

A.3 Plots from stored RVG data during field trip

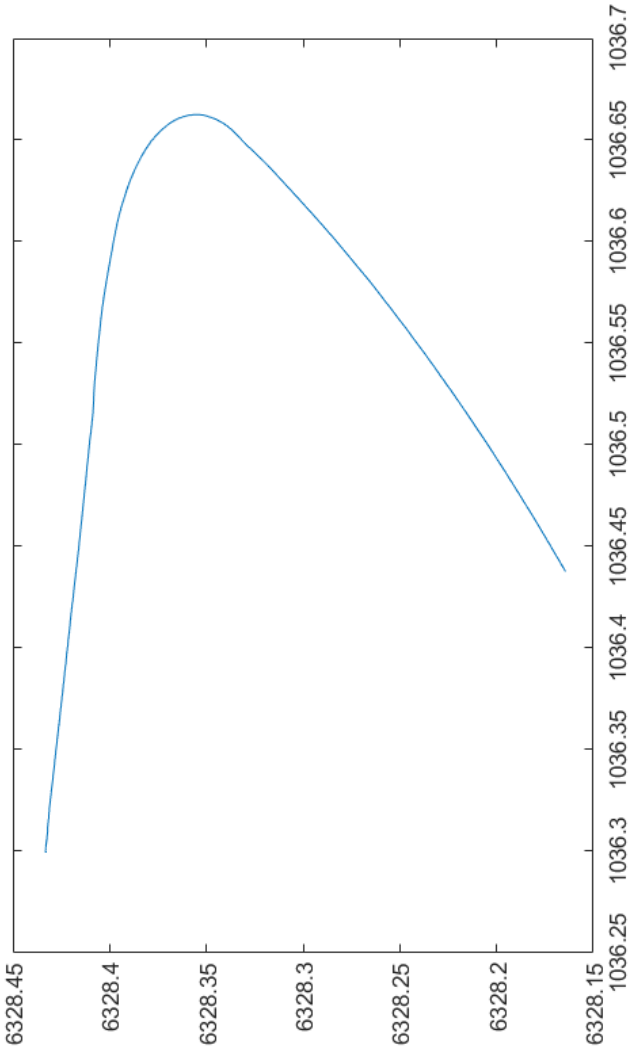


Figure A.3: Position of RVG from log datastream_102722_11-44-35_1200s/$GPGGA.csv. x-axis is longitude, y-axis is latitude.

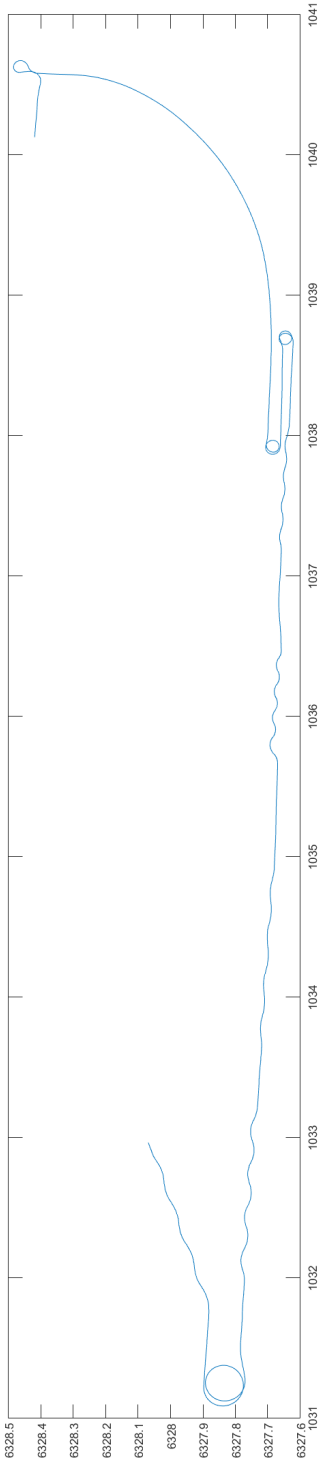


Figure A.4: Position of RVG from log datastream_102722_10-02-12_3600s/$GPGGA.csv. x-axis is longitude, y-axis is latitude.

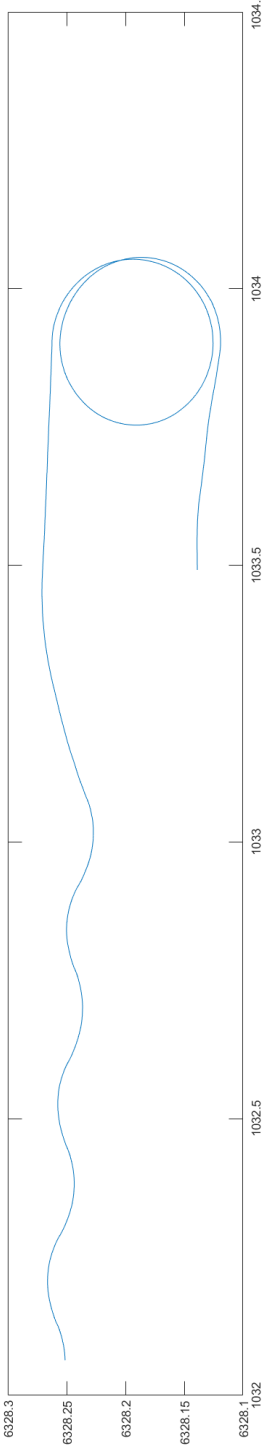


Figure A.5: Position of RVG from log datastream_102722_09-50-46_600s/$GPGGA.csv. x-axis is longitude, y-axis is latitude.

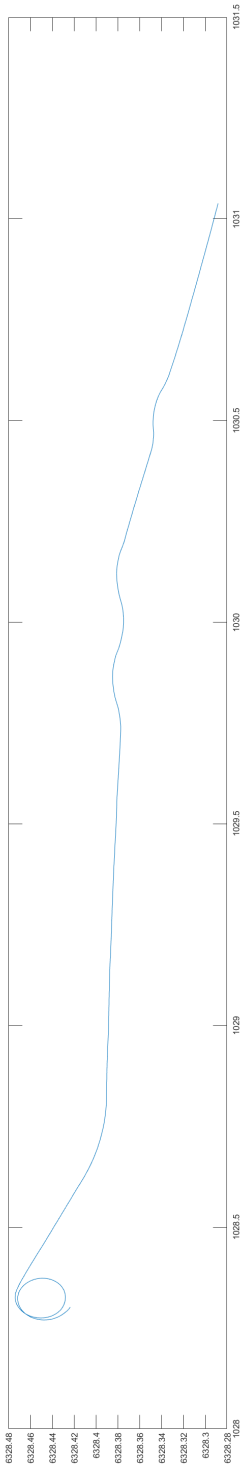


Figure A.6: Position of RVG from log datastream_102722_09-38-11_600s/$GPGGA.csv. x-axis is longitude, y-axis is latitude.

A.4 Full View of Statistics Page in the Web Application for 2D EC

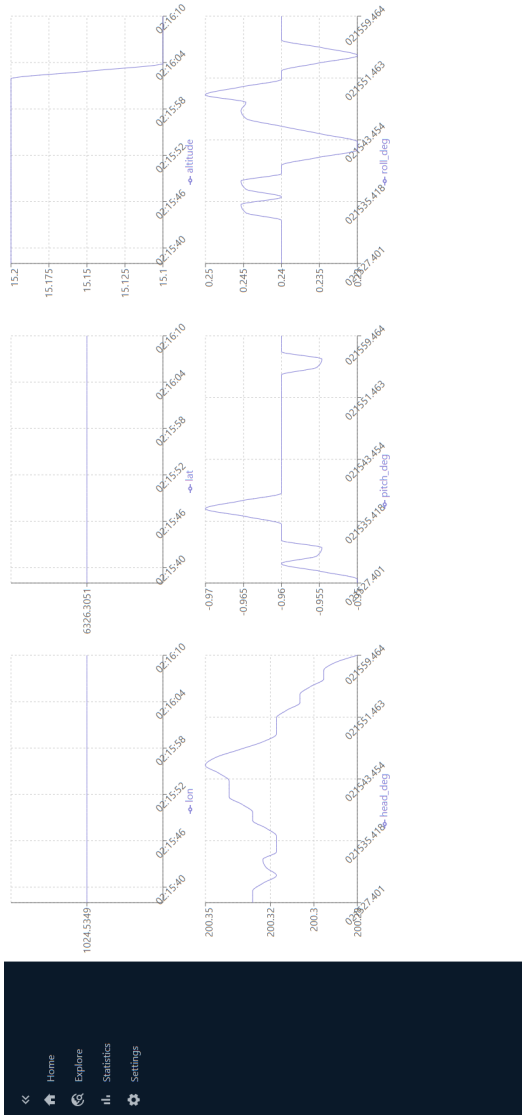
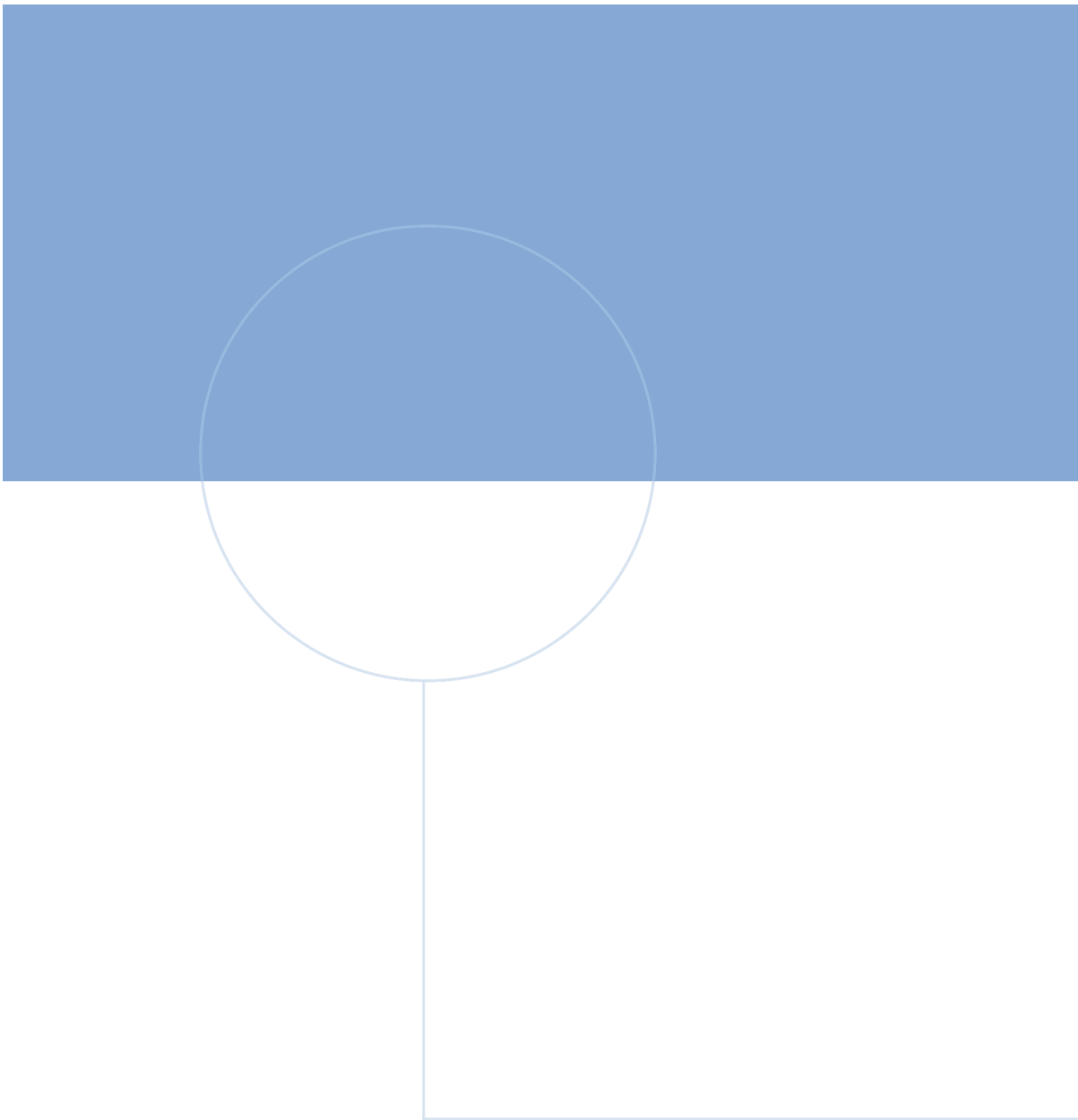


Figure A.7: Full View of Statistics Page in the Web Application for 2D EC



NTNU

Norwegian University of
Science and Technology