Master's thesis

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

James Premraj

Development of a Digital Twin for the Marine Cybernetics Lab: 3D visualization and simulation, online monitoring, and remote control

Master's thesis in Marine Technology
Supervisor: Roger Skjetne
June 2023

NTNU
Norwegian University of
Science and Technology

James Premraj

Development of a Digital Twin for the Marine Cybernetics Lab: 3D visualization and simulation, online monitoring, and remote control

Master's thesis in Marine Technology
Supervisor: Roger Skjetne
June 2023

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:	Premraj, James
Field of study:	Marine cybernetics
Thesis title (Norwegian):	Utvikling av en digital tvilling for Marine Cybernetics Lab: 3D visualisering og simulering, fjernovervåking og fjernstyring
Thesis title (English):	Development of a Digital Twin for the Marine Cybernetics Lab: 3D visualization and simulation, online monitoring, and remote control

Background

Autonomous ship applications are trending, and in the future there will be needs for students with background and competence in maritime autonomous control systems and operations, and use of remote control centres for autonomous ships. Simulating entire closed-loop autonomous systems requires complex simulation and test environments capable of modeling and verifying vessel dynamics, perception sensors, and external factors. Such simulators are essential in the assurance of autonomous vehicles, CARLA (https://carla.org) being a prime example from the autonomous automotive industry. Gemini is a Unity-based platform originating from NTNU, aiming to fill the same role for autonomous ships. At the Department of Marine Technology we have the privilege of emulating autonomous ship operations using autonomous model ships in MC-Lab connected to a remote control centre (RCC) hosted in the OS-Lab.

The objective of this project is to virtually recreate the MC-Lab and selected ROS-enabled cybership vessels, such as C/S Enterprise I (CSE1), C/S Arctic Drillship (CSAD), and C/S Saucer (CSS), in Gemini's 3D simulated environment and other relevant digital components. This will allow for software-in-the-loop and hardware-in-the-loop testing, as well as emulating RCC operations, of the cybership vessels and their control systems as autonomous ship systems. As an extension, the relevant vessels can be outfitted with perception sensors (regular camera and lidar). The final goal is for this digital twin, remote control centre, and autonomous ship system to be available for students as a digital-physical learning platform during their education.

Scope of Work

1. Perform a background and literature review to provide information and relevant references on:
   - Marine digital twin definitions, scope, services, platforms.
   - Maritime autonomous surface ships (MASS) and RCC; definitions, concepts, setups, etc.
   - Situational awareness functions and relevant sensors.
   - Certification of and building trust in autonomous marine vessels.
   - Unity and Gemini simulation platforms, ROS, etc.
   - MC-Lab, Ocean Systems Lab (OS-Lab), Shore Control Lab, Fjordlab/TBS, etc.

   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 MC-Lab model-scale operations and present them with real-time odometry in a 3D environment of MC-Lab, incl. situation awareness sensors (that may only be virtual). Implement 3D object drawings for relevant CS-vessels with correct dimensions, including a user-friendly functionality for selecting which vessel that corresponds to the datastream. Include also a simulation function that can numerically emulate MC-Lab operation of a CS-vessel in Gemini, also including situation awareness sensor(s). Demonstrate the solution.
3. Develop the OS-Lab with an RCC function for MC-Lab, by either simulating the operation or being connected to a real experiment. Demonstrate scenarios where an RCC-operator monitors and commands an autonomous CS-vessel in MC-Lab, say in a DP operation, both simulated and real.

4. Develop a (simple) obstacle avoidance method for a CS-vessel, with an emulated Lidar sensor being available. Design a test scenario where a reference ship is emulated in Gemini. Set up virtual obstacles and demonstrate the collision avoidance response for the CS-vessel.
5. Document well the developed software system, functionality, and startup procedure, including repository for availability to code, for new students to continue your good work.

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 available for NTNU, for education and research purposes, and be included electronically with the report.

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

Supervisor: Roger Skjetne
Co-advisor(s): N/A

Signatures:

Digitally signed by rskjetne
Date: 2023.06.05 16:05:37 +02'00'

Preface

This master thesis is written as part of the Marin Technology study program at the Norwegian University of Science and Technology (NTNU). It presents a theoretical background and literature review of relevant studies on digital twins and remote control. Additionally, it includes the development of such a digital twin with remote control functions, and presents results from both simulations and physical experiments.

Working on this thesis has been an interesting and fulfilling experience. It has given me a sense of accomplishment by creating a valuable and useful product that is built on 5-years worth of accumulated knowledge from the Marine Technology Master's degree program. There have been some challenges throughout the process, but it felt rewarding when the solutions started to show promising results.

Ideally, the reader should have some basic knowledge on marine cybernetics and programming.

James Premraj
June 2023, Trondheim

Abstract

Physical testing of an autonomous system can be expensive and time-consuming. However, this issue can be mitigated by creating digital twins (DT) of the vessels, which can be altered and tested in simulators. The Marine Cybernetics lab (MC-lab) at Tyholt contains different vessels that are used to test out motion control systems and perform hydrodynamic tests. This thesis focuses on the development of a DT of the MC-Lab with remote control center (RCC) functions.

The DT comprises 3D models (in Unity) of the MC-Lab and the cybership vessels: CS Enterprise 1 (CSEI), CS Saucer (CSS), and CS Arctic Drillship (CSAD). The motion of these 3D models are connected to simulators in ROS, which run on a seperate machine. This enables visualization of 2D ROS simulators in a 3D environment. Virtual sensors, such as light detection and ranging (LiDAR) and camera, are attached on the virtual vessels, providing sensor data from the virtual 3D environment in Unity.

The RCC function consists of two main components; remote control and monitoring. Remote control allows an operator to control the physical vessels at the MC-Lab from a remote location. This can be done either by manual control using joysticks or by sending desired waypoints for a DP controller. The remote monitoring function allows visualization of the physical vessel in the 3D environment. This combination opens up the possibility of conducting hybrid tests with virtual sensor data and virtual obstacles that are not present in the physical lab, while simultaneously controlling the physical vessel.

The RCC is hosted at the OS-Lab which is connected to the local network at the MC-Lab using a Meraki SD-WAN. Data, such as joystick inputs or odometry data from the physical vessel, is transmitted between the labs using UDP messages. The data is formatted to ensure its size is small enough to be sent and received at a high frequency.

These systems were tested through simulations and physical experiments in both the MC-Lab and OS-Lab. The systems proved to be successful in representing the physical lab in a virtual environment in real-time. However, the efficiency and accuracy of the system have some limitations due to hardware restrictions, particularly in the remote monitoring module. Nevertheless, considering the hardware available at the MC-Lab and OS-Lab at the time of publishing this thesis (spring 2023), the system yields acceptable error margins.

Sammendrag

Fysisk testing av et autonomt system kan være dyrt og tidskrevende. Dette kan motvirkes ved å lage digitale tvillinger av fartøyene som kan testes i simulatorer. MC-laben (Marine Cybernetics lab) på Tyholt har forskjellige fartøy som brukes til å teste ut kontrollsystemer og gjøre hydrodynamiske tester. Denne oppgaven tar for seg utviklingen av en digital tvilling av MC-Lab med fjernkontroll funksjoner.

Den digitale tvillingen består av 3D-modeller (i Unity) av MC-Laben og fartøyene; CS Enterprise 1 (CSEI), CS Saucer (CSS) og CS Arctic Drillship (CSAD). Bevegelsen til 3D-modellene er koblet til simulatorer i ROS som kjører på en annen maskin. Dette gjør at 2D-simulatorer i ROS kan visualiseres i et 3D-miljø. Virtuelle sensorer som light detection and ranging (LiDAR) og kamera er festet på de virtuelle fartøyene og sender ut sensordata fra det virtuelle 3D-miljøet i Unity.

Fjernkontrollsystemet består av to hoveddeler; fjernstyring og overvåking. Fjernstyring lar en operatør kontrollere de fysiske fartøyene på MC-laben fra en annen lokasjon. Dette kan enten være ved å styre dem manuelt med en joystick eller sende koordinater til en DP-kontroller. Fjernovervåkingsfunksjonen gjør at det fysiske fartøyet kan visualiseres i 3D-miljøet. Ved å kombinere dette åpnes muligheten for å gjøre hybridtester med virtuelle sensordata med virtuelle hindringer som ikke er tilstede i det fysiske laboratoriet, og samtidig kontrollere det fysiske fartøyet.

Verten for fjernkontrollsystemet er OS-laben som er koblet til det lokale nettverket på MC-laben ved hjelp av et Meraki SD-WAN. Data, som joystick-inputen eller odometridataene fra det fysiske fartøyet, sendes mellom laboratoriene ved hjelp av UDP-meldinger. Dataen blir formatert slik at datastørrelsen er liten nok til å kunne sende og motta dem med høy frekvens.

Disse systemene ble testet gjennom både simuleringer og fysiske eksperimenter i MC-Laben og OS-Laben. Systemene viste seg for å være vellykket med å representere det fysiske laboratoriet i et virtuelt miljø i sanntid. Effektiviteten og nøyaktigheten til systemet har noen begrensninger på grunn av maskinvarebegrensninger, hovedsakelig i fjernovervåkingsmodulen. Systemet gir imidlertid akseptable feilmarginer med maskinvaren til stede ved MC-laboratoriet og OS-laben på tidspunktet denne oppgaven blir utgitt(våren 2023).

Acknowledgements

Throughout this project, there have been many contributors that have assisted with different aspects of the work. Firstly, I would like to thank my supervisor, Roger Skjetne, for the guidance and suggestions regarding the thesis.

I would also like to thank Senior Engineers at IMT, Robert Opland and Stian Molden, for setting up the network between the OS-Lab and the MC-Lab. Without their help, it would have not been possible to create a remote control center hosted from the OS-Lab.

Learning about Gemini and understanding Unity was also a lot of work. Therefore, I would like to thank Robin Stokke for providing existing 3D models of the MC-Lab and some scripts like the "Nav client" and "Force controller".

Lastly, I would like to thank Eirik Midtun for collaborating on executing the physical tests in the MC-Lab.

Table of Contents

List of Figures

List of Tables

Nomenclature

List of Abbreviations

2D Two dimensional

3D Three dimensional

AIS Automatic identification systems

AUV Autonomous underwater vehicle

AWACS Active Wave Absoption Control System

CNN Classical Convolutional Neural Network

COLREG Convention on the International Regulations for Preventing Collisions at Sea

CPU Central Processing Unit

CSAD C/S Arctic Drillship

CSEI C/S Enterprise I

CSS C/S Saucer

DOF Degrees of freedom

DP Dynamic Positioning

DS4 DualShock 4

DT Digital Twin

EMR Electromagnetic Radiation

GNSS Global navigation satellite system

GPU Graphics Processing Unit

HDRP High definition render pipeline

IMAT Integrated Maritime Autonomous Transport Systems

IMO International Maritime Organization

INS Inertial Navigation System

IR Infrared

L left turn

LiDAR light detection and ranging

LOAS Land-based Operation of Autonomous Ships

LTL Linear Temporal Logic

MASS Maritime Autonomous Surface Ships

MC-Lab Marine Cybernetics Laboratory

MQTT Message Queuing Telemetry Transport

NTNU Norwegian University of Science and Technology

OS-Lab Ocean Systems Laboratory

P2P Peer-to-Peer

PLM Pig Loop Moudle

QTM Qualisys Track Manager

R Right turn

RADAR Radio Detection and Ranging

RAM Random Access Memory

RDF Radio Direction Finder

ROS Robot Operating System

ROV Remotely Operated Vehicle

RPC Remote Procedure Call

RPN Regional Proposal Network

RTK Real Time Kinematic

S Straight

SAREPTA Safety, autonomy, remote control and operations of industrial transport systems

SDP Subsea Docking Plate

SFI Center for reasearchdriven Innovation

SOLAS Safety Of Life At Sea

SONAR Sound Navigation and Ranging

TBS Trondhjem Biological Station

TCP Transmission Control Protocol

UDP User Datagram Protocol

VL Visible light

VR Virtual reality

List of symbols

ϵ Estimated emissitivity variable for surfaces

λ Radar wavelength

ϕ Elevation angle in the YZ-plane

σ Stefan-Boltzmann constans

θ azimuth angle in the XY-plane

K Camera calibration matrix

R Rotation matrix

T Translation matrix

$\triangle$ Displacement

A Cross section Area

B Radar Bandwidth

G Antenna gain

P_r Received power

P_t Transmitted power

P_{spokes} Received power for a single spoke

R Target range

T Temperature

W Emissivity

Chapter 1

Introduction

1.1 Motivation

Simulators have emerged as cheaper and more time-efficient tools to substitute physical tests in the design process of autonomous systems. Simulating entire closed-loop autonomous systems requires complex simulation models of vessel dynamics, perception sensors and external factors. In this context, Gemini, a Unity-based platform originating from NTNU, has emerged as a promising candidate due to its ability to provide a less complex, yet highly accurate simulation environment.

The primary goal of this master's thesis is to virtually recreate the Marine cybernetics laboratory (MC-lab) and selected Robot Operating System(ROS) enabled cybership vessels, such as C/S Enterprise I (CSEI), C/S Arctic Drillship (CSAD), and C/S Saucer (CSS), in Gemini. A digital twin (DT) like this will allow for software-in-the-loop testing, as well as hardware-in-the-loop testing of the cybership vessels. To expand the applications, the physical vessels can be fitted with perception sensors, such as light detection and ranging (LiDAR) and camera. These sensors can then be accurately recreated virtually within the Gemini platform, allowing for the design and evaluation of situational awareness algorithms in a more efficient manner. Moreover, by comparing real-world data collected from the physical vessels with the simulated data generated in Gemini, the effectiveness and reliability of the simulation can be evaluated and validated.

1.2 Scope of work

This thesis consists of four objectives. The first objective is to gather the necessary background information that is needed to recreate the MC-Lab in Gemini. The second objective is to develop a DT of the Lab and the cybership in Gemini that can be simulated

to behave like the real vessels when receiving control input. The third objective is to fit those vessels with virtual sensors like LiDAR and camera. The fourth objective is to create a Remote Control Center (RCC), hosted from the Ocean Systems Lab (OS-Lab) to remotely control and monitor the vessels. To achieve this, the following scope of work is defined:

1. Perform a background and literature review to provide information and relevant references on:

   - Marine DT definitions, scope, services, platforms.
   - Maritime autonomous surface ships (MASS) and RCC; definitions, concepts, setups, etc.
   - Situational awareness functions and relevant sensors.
   - Certification of and building trust in autonomous marine vessels.
   - Unity and Gemini simulation platforms, ROS, etc.
   - MC-Lab, Ocean Systems Lab (OS-Lab), Shore Control Lab, Fjordlab/TBS, etc.

   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 MC-Lab model-scale operations and present them with real-time odometry in a 3D environment of MC-Lab, incl. situation awareness sensors (that may only be virtual). Implement 3D object drawings for relevant CS-vessels with correct dimensions, including a user-friendly functionality for selecting which vessel that corresponds to the datastream. Include also a simulation function that can numerically emulate MC-Lab operation of a CS-vessel in Gemini, also including situation awareness sensor(s). Demonstrate the solution.

3. Develop the OS-Lab with an RCC function for MC-Lab, by either simulating the operation or being connected to a real experiment. Demonstrate scenarios where an RCC-operator monitors and commands an autonomous CS-vessel in MC-Lab, say in a dynamic positioning (DP) operation, both simulated and real.

4. Develop a (simple) obstacle avoidance method for a CS-vessel, with an emulated LiDAR sensor being available. Design a test scenario where a reference ship is emulated in Gemini. Set up virtual obstacles and demonstrate the collision avoidance response for the CS-vessel

5. Document well the developed software system, functionality, and startup procedure, including repository for availability to code, for new students to continue your good work.

1.3 Contributions

The first contribution of this thesis involves the development of a DT of the MC-Lab in Gemini. Existing 3D models were integrated and scripts were created in order to enable joystick control of the virtual vessels. A system to connect different ROS simulators and the visualization in Gemini was developed predominately in Python and C#. Virtual sensors, such as LiDAR and cameras, have been attached on the virtual vessels, allowing for easy access to sensor data. The OS-Lab has been connected to the local network at the MC-Lab in order to be able to monitor and control the physical vessels. Python code was written to receive the 6 degrees of freedom (DOF) position of the physical vessel and visualize it in Gemini. Additional python scripts were written to enable remote control of the vessels using a DualShock4 (DS4) controller. An interface was implemented to select waypoints for a DP controller and control the physical vessels remotely. This thesis contributes also with tools, documentation, and procedures for the system. Lastly, simulations and physical tests were performed to assess the limitations and advantages of these systems.

1.4 Outline

This report is structured as following:

Chapter 1: Introduces the thesis and explains the motivation, scope, and contributions of the thesis.

Chapter 2: Provides the relevant theoretical background for the presented work and a literature review of existing relevant studies.

Chapter 3: Presents the problem formulation for the thesis.

Chapter 4: Presents the DT. It includes the setup, different submodules, and the test cases that are performed.

Chapter 5: Presents the remote control center. This includes the architecture of the system, interface, and test cases that are performed.

Chapter 6: Presents the results from the simulations and physical tests.

Chapter 7: Presents the conclusions and further work.

Chapter 2

Background and literature review

This chapter presents relevant references and background information on creating a digital twin (DT) of the MC-lab in Gemini with remote control center (RCC) functions. This includes Unity, Gemini, situational awareness, ROS, the laboratories relevant for autonomous marine vessels, and RCCs.

2.1 Marine digital twins

The concept of a DT has become a buzzword that can be interpreted differently by different people. This is because a DT can vary depending on the domain and applications of the system. The term "digital twin" was first introduced by Michael Grieves in 2002. This term was defined as "set of virtual information constructs that fully describes a potential or actual physical manufactured product from the micro atomic level to the macro geometrical level" in Grieves and Vickers [2017]. It is also mentioned in Grieves and Vickers [2017] that the real system and DT should be linked together for the whole lifecycle of the real system and mirror it.

This definition is too optimistic. Mirroring every part of a real system is not realistic. A marine system is built of several complex systems. An example of a DT would be DNV's project called "Nerves of steel", a DT of a vessel that can estimate the damage on a ship hull depending on Automatic Identification Systems (AIS) data and determine what kind of weather and sea condition the vessel has operated in [DNV, 2021b]. Another use case of a DT could be the CSEI simulator in git [2022] which mirrors the behaviour of a vessel that is given different thruster inputs. These are clear examples of two very different systems, but could be labelled as DTs according to Grieves and Vickers [2017]. Therefore, there is a need for a better definition that is more generally applicable. According to Cabos and Rostock [2018] a DT should fulfill three criteria:

- Asset representation: The DT should represent a unique physical system.
- Behavioural model: The DT should mirror the behaviour of the physical system and be able to predict behaviour and aid in decision making.
- Condition or configuration data: There should be data reflecting the status of the physical system and changes to the system should be updated into the DT throughout the lifecycle. This can be done by inspections or measurements.

Combining all of this, a good definition can be found in the recommended practice document, DNV [2021a] , stating "A digital twin is a virtual representation of a system or asset that calculates system states and makes system information available, through integrated models and data, with the purpose of providing decision support over its life cycle." A DT can not be defined by a specific variable such as size or complexity, but rather be a virtual representation of the most important states that is needed to be able to aid as decision support.

The benefits of using a DT can be:

- To increase the manufacturing flexibility and competitiveness.
- Improve product design performance.
- Forecasting the health and performance of products over lifetime.
- Improved efficiency and quality in manufacturing [Cabos and Rostock, 2018].

This is only possible if the DT is continuously updated for every change on the physical vessel over its lifetime. If performed correctly, the benefits can outweigh the cost the of development and maintenance of the DT.

2.2 Maritime autonomous surface ships (MASS)

The term "Maritime autonomous surface ships" can be systems with different operational use and levels of complexity. This section presents general understanding of autonomous marine vessels and the different categories within.

2.2.1 Autonomy levels

Autonomy can be implemented into different levels and is not restricted to only one. It is important to note that an unmanned system does not imply an autonomous system. An unmanned vessel can be purely remote control, which is not fully autonomous. An autonomous vessel can be manned as a safety barrier in order to take control if the system

fails. Vessels can also be periodically unattended, e.g., being unmanned when performing automatic well defined problems and manned during more complex operations.

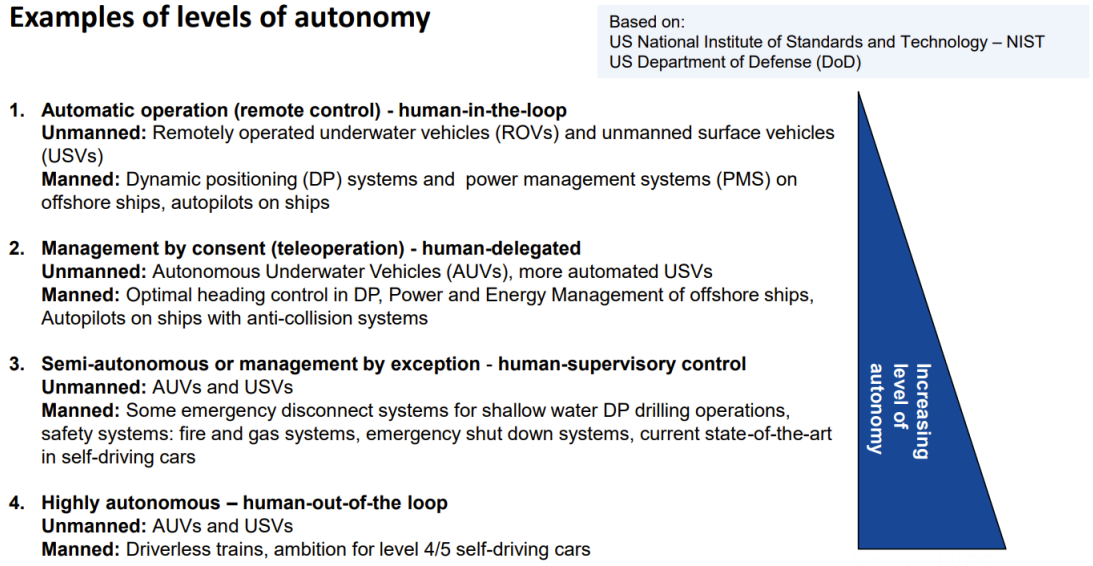


Figure 2.1: Levels of autonomy. Courtesy: Smogeli [2022a].

Figure 2.1 is just one example of how one can divide autonomy into different levels. The simplest form of autonomy is remote control, where the system is controlled by a human operator. One step above this level is management by consent, where the system will give recommendations to the operator. Another level is management by exception where the system takes control during specific situations, e.g., where the operator does not have time to react. The last level is the highly autonomous level, where human is almost completely taken out of the system.

The Norwegian Maritime Authority presented a different proposal of these levels in Authority [2020]:

1. Decision support
2. Autonomous
3. Periodically unmanned
4. Unmanned
5. Fully autonomous

This interpretation of levels shows that there is no clear standard of levels of autonomy and not all of them are correct. In the definition of the Norwegian Maritime Authority, unmanned and autonomous systems are compared, where unmanned systems are at a higher level than autonomous systems. This is therefore not a fair comparison.

2.2.2 Operational setups of typical autonomous ship operations

There are various operations an autonomous ship can perform. Many of the "low level" automatic systems have been in use in the maritime industry for many years such as DP systems. DP systems have traditionally been used to keep a fixed position and heading or move from one location to another at low-speed [Sørensen, 2011]. There are various DP systems, but the basic functionallity consists of the same submodules:

- Signal Processing - signals from sensors need to be analyzed and processed. With multiple sensors on board, the measurements need to be weighted to ensure redundancy.
- Vessel observer - Wave filtering and state estimation.
- Controller logic - DP systems can operate in different modes. Mode transition are controlled by this submodule.
- Control law - controllers such as PID or feedforward controllers that send out the commanded thrust forces based on the state of the vessel and reference.
- Guidance system and reference systems - specifies the desired position and heading, and generates smooth reference paths for the vessel to follow.
- Thrust allocation - computes the force and direction commands of each thruster based on the commanded forces and moments in surge, sway, and yaw.
- Process plant - the vessel itself or a mathematical model describing the vessel dynamics.

These systems focus mostly on the motion control part found in Figure 2.2.

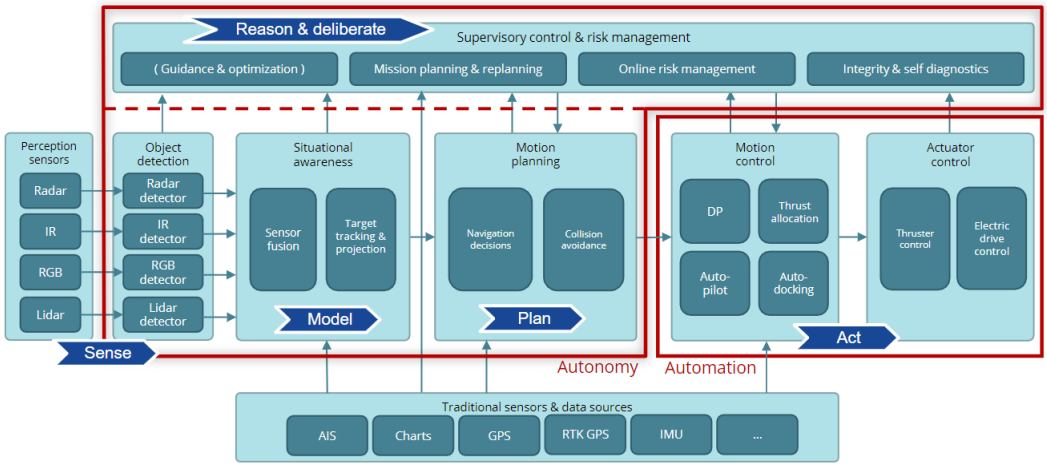


Figure 2.2: Typical control architecture. Courtesy: Smogeli [2022a].

Other systems like course autopilots, line-of-sight guidance, path following and path tracking systems presented in Fossen [2021], have more complex algorithms in the motion planning module. These algorithms have the objective of creating a path that the vessel is able to follow. One method is to use path generation using straight lines and inscribed circles in order to avoid sharp turns located at the waypoints. Another method is straight-line paths based on circles of acceptance, where the next waypoint is chosen when the vessel is close enough to the target waypoint using a circle of acceptance with a defined radius. Path generation using Dubins path presented in Dubins [1957] is also a possible way of path generation based on combinations of a right turn (R), left turn (L) or straight (S).

Fully autonomous vessels like the Milliampere 2 by NTNU have more complex modules like the situational awareness and supervisory control & risk management modules as well. This vessel uses Real Time Kinematic (RTK) GNSS, radar, LiDAR, optical cameras and infrared cameras in order to perceive the environment, detect objects and then making deliberate choices in the supervisory module [Zeabuz, 2022]. This shows that the subsystems become more and more complex as the autonomy level increases.

2.2.3 Certification of autonomous marine systems

The growth of interest in autonomous marine systems in recent times have created a new obstacle for the marine industry regarding safety and trust. The former is addressed by rules and regulations set by International Maritime Organization (IMO), Safety Of Life At Sea (SOLAS), Convention on the International Regulations for Preventing Collisions at Sea (COLREG), etc. These are based on the assumption that the vessel is operated and manned by real humans. By extracting the human out of these systems, there is a challenge on how to certify and regulate these autonomous systems.

As of today, there are no clear rules and regulations regarding autonomous marine vessels. This is still in development and regulators are currently developing the rules, e.g., the MASS Code presented on IMO [2022]. IMO has completed a regulatory scoping exercise on MASS, designed to assess IMO instruments to see how they can apply to ships with different levels of automation presented in Section 2.2.1. This scoping exercise contains:

1. Description of all degrees of autonomy.
2. Most appropriate ways of addressing MASS operations.
3. Identifying potential gaps.
4. Identifying possible links between instruments.
5. Identifying priorities for further work.

IMO may set the international rules, but flag states such as the Norwegian Maritime Authority can provide exemptions within national waters and make bi-lateral agreements with other states. This can be an opportunity for autonomous vessels to operate in Norwegian waters, but this is still in progress.

One possibility of certifying autonomous marine vessels is to use formal methods that are mathematically based techniques for the specification, development and verification of software and hardware systems.

Formal specification is the process of formulation requirements of a system using a mathematically based syntax. This process avoids ambiguities, inconsistencies and incompleteness in the definition of the requirements. One method of formal specification is *Linear Temporal Logic* (LTL). This method is based on temporal logic that specifies behavior on boolean discrete-time signals in a machine readable way. LTL formulas can be constructed using logical operators like:

- And
- Or
- Not
- Implication

and temporal operators such as:

- Eventually
- Always
- Next
- Until

Formal specification can be used to verify models. One method is *Model checking* presented by Clarke and Schlingloff [2001]. This verification technique checks if a finite-transition system meets a temporal logic property and verifies by evaluating every possible execution. The final output of this technique is either a verification or a falsification. One downside of this technique is that evaluating every possible execution is exhaustive and not scalable [Clarke and Schlingloff, 2001].

Another method is *Automated Theroem Proving* presented in Robinson [1980]. This is a computer tool that, given a set of system properties and a system requirement, builds a step-by-step proof that the requirement holds, given the properties.

There are also other types of verification methods such as simulation-based methods e.g., manual testing, simulation-guided Lyapunov Analysis, and concolic testing. There are also analytical methods such as linear analysis, stability proofs, etc. [Smogeli, 2022b].

2.2.4 Building trust in autonomous systems

Building trust in the autonomous systems for customers and stakeholders is just as important as certifying and verifying them. If no one is willing to use the systems, they do not add any value to society.

There are three main areas to look at when it comes to filling the "trust gap" [Smogeli, 2022b]:

1. Trust the human or machine.
2. Trust the platform or company.
3. Trust the idea or technology.

If these aspects are satisfied, the systems can be implemented into society with promising outcomes. There are many challenges in solving these problems. One of them is to find the right balance between *technical safety* and *perceived safety* [Smogeli, 2022b]. An autonomous vessel may be technically safe and documented as safe, but can feel unsafe for the passengers. One example is a plane in turbulence. This may not be technically unsafe, but can worry the passengers on board.

Another issue is balancing between safety and performance. An autonomous marine vessel may be safe, but if it is too safe, the performance can be reduced, e.g., a vessel that moves too slow due to traffic that is far away from the vessel.

There is no clear guideline to solve these problems. Companies can try to make test drives in a controlled environment with passengers and get feedback and improve their systems and marketing strategies, but there is no guarantee that this will work. Building trust in autonomous systems is an ongoing process.

2.3 Simulation and embedded control platforms

2.3.1 Unity

Unity is a cross-platform game engine developed by Unity Technologies. The first version of Unity was launched in June 2005 and has had several updates til this date. This engine is used to create 3D and 2D games, simulations, data visualization and other non-game applications [Haas, 2014].

C++ is used to develop the game engine; however, developers use C# when creating games in Unity. A way to bypass this is to use the graphical editor which allows the user to create shapes, work with the camera and add materials without writing code. Nonetheless, creating more complex features will require some degree of coding in C#.

Game objects in Unity are built by several different components. The mesh of an object defines the physical shape of the object. The mesh renderer adds materials and lighting to the objects. Blender is a 3D modelling tool that can be used to create such shapes and objects. Physics components, e.g., rigid bodies and colliders, can simulate the physical behaviour like gravity and add the 3D shape of the object that can collide with other objects. It is also possible to add C# code scripts to add features when the objects interact with the world, e.g., receiving extra points when a bullet object hits a target.

The unity editor is comprised of different sub-windows. A detailed description of every feature is presented in UnityTechnologies [2015]. The main windows are: Project Browser, Inspector, Game View, Scene View and Hierarchy. The Project Browser contains all the assets that have been imported into Unity that can be used. This window is similar to Explorer in Windows and allows the user to easily navigate through the necessary files.

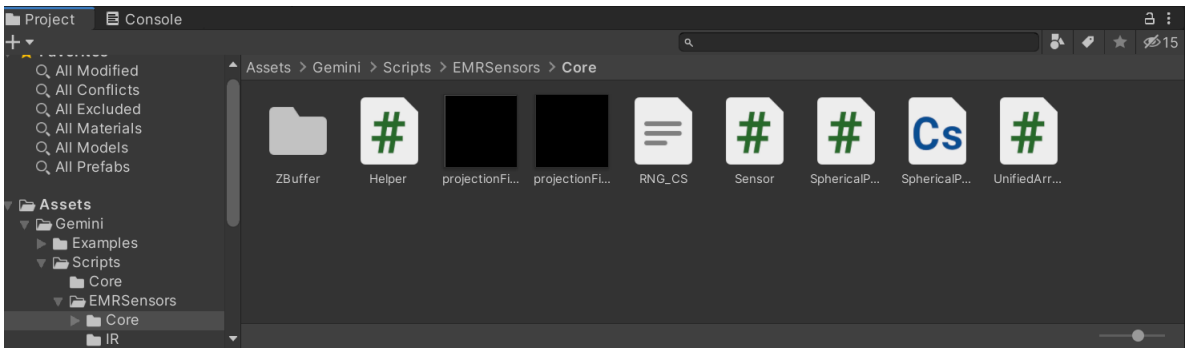


Figure 2.3: Screenshot of the project browser in Njords simulator [git, 2023].

The Inspector window allows the user to view and modify the properties and details of the game objects, e.g., physics, colliders, and Scripts.

Figure 2.4: Screenshot of the scene view in Njords simulator [git, 2023].

In the Scene View window, the user can quickly select and change the position and rotation of every object, e.g., the player, camera, and environmental objects.

Figure 2.5: Screenshot of the game view in Njords simulator [git, 2023].

The Game View is the rendered view from the camera(s) that were placed using the Scene View. Rendering is the process of generating a photorealistic or nonphotorealistic image from a 2D or 3D model. The Game View is the view that presents the final version of the game. This mode allows the user to see what the player sees when playing the game through the camera(s). Figure 2.5 shows, when looking at the pixelated 3D models, that the quality of the rendering of the models are impacted by the GPU of the machine the simulator is running on.

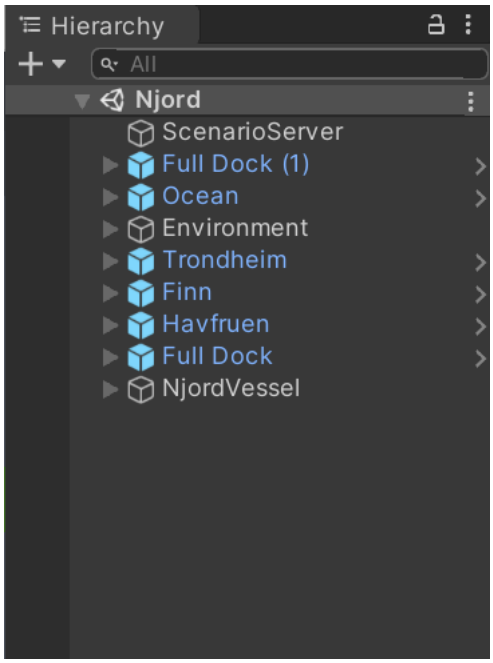


Figure 2.6: Screenshot of the hierarchy in Njords simulator [git, 2023].

The Hierarchy window contains every object present in the current Scene. Objects in this window can be chosen and dragged to use the *parenting* feature. This feature allows the children objects to have the same movement and rotation as the parent object.

2.3.2 Gemini

Gemini is a Unity-based visual simulator originally developed at NTNU as a foundation for Electromagnetic Radiation (EMR) based sensors, e.g., LiDAR and Radar, to be tested for autonomous applications. It is also possible for developers to interface and communicate with the vessels and environments simulated in Gemini.

Virtual environments that are currently in progress are the milliAmpere and milliAmpere 2 in Ravnkloa, Njord autonomous challenge, "Digital Trondheimsfjord" - Ocean Autonomy Cluster, and Zeabuz 1 in Stockholm [Smogeli, 2022b]. This simulator is used to test dangerous scenarios that cannot be performed in the real world without high risks. It is also easier to test new sensor configurations and generate labeled image data for machine learning algorithms using the simulated sensors.

The implementation of the visible-light (VL) and infrared (IR) camera sensors and simulations are presented in Vasstein et al. [2020]. Gemini uses Unity's high definition render pipeline (HDRP) to implement these sensors. LiDAR and radar sensors are implemented using a custom made pipeline. These pipelines can be observed in Figure 2.7 where the purple boxes are already implemented by Unity and the blue boxes are implemented work in Gemini.

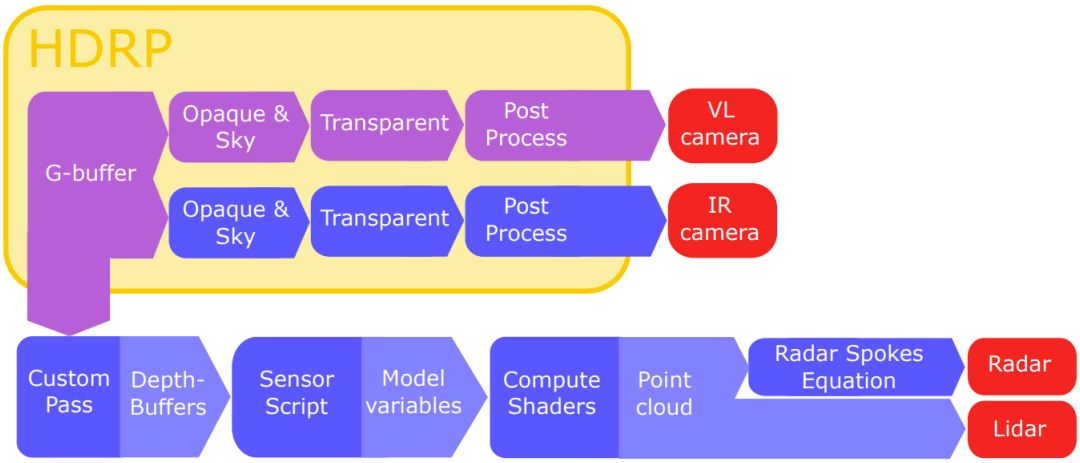


Figure 2.7: GPU data pipeline with HDRP for VL and IR sensors and a custom pipeline to simulate radar and LiDAR. Courtesy: Vasstein et al. [2020].

The VL camera is created using Unity's HDRP that uses G-buffers. The G-buffer contains 4 or more images containing information such as the reflections, surface normals, colors, and depth. Combining these images results in the VL camera. The IR camera on the other hand is created by implementing the thermal radiation using Stefan-Boltmanns law [Maldague, 2001] to determine the total emissivity,

$$W(T) = \epsilon\sigma T^4, \tag{2.1}$$

where σ is the Stefan-Boltzmann constant, ϵ is the estimated emissivity variable for a surface and T is the temperature. This is based on the assumptions that the surfaces are close to ambient temperatures and emit light described by Lambert's emission law. This law is then implemented using Unity's Shader Graph tool creating a function.

The range of the objects perceived by the LiDAR can be simulated by using the camera's depth buffer which can be fetched from the G-buffer. A 360° LiDAR can be simulated by stitching depth-buffers from multiple cameras placed and rotated around the azimuth angle to increase the field and get a panoramic view, into a 2D array. The values can be expressed in both the *image space* and the *camera view space*. Mapping the values between the coordinate system are presented in detail in Vasstein et al. [2020], but a more general mapping is described in Section 2.4.2

The Radar plot is constructed by rotating a single radar spoke that transmits and receives a power radiation by using an antenna. This can be described using the radar range equation

$$P_r = \frac{P_t G^2 \lambda^2 A}{(4\pi)^3 R^4}, \tag{2.2}$$

where λ is the radar wavelength, G is the antenna gain, P_t is the transmitted power, R is the target range, and A is the cross-section area. To simplify, $\lambda = 1$ and the antenna

gain is created with an amplitude $|G_{max}|$ and normalized radiation pattern $G(n_x, n_y)$. The values from the 2D array are set for each pixel. To transform this into one single spoke, a variable called the spoke range, $r = \frac{R(n_x,n_y)}{f} r_{max}$, needs to be introduced, with the resolution of r_{max} cells. This leads to the altered equation

$$P_{\text{spokes}}(r,k) = \frac{P_t \lambda^2 |G_{\max}|^2}{(4\pi)^3} \sum_{n_x=0}^{L \cdot W} \sum_{n_y=0}^{H} \frac{G\left(n_x + k \cdot \frac{L \cdot W}{h}, n_y\right)^2}{R(n_x, n_y)^4}, \tag{2.3}$$

where L is the number of the camera , W is the depth-buffer's width, H is the depth-buffer's height and k is the spoke index. The final radar spokes equation includes radar range uncertainty using a discrete Gaussian function and is presented as [Vasstein et al., 2020]

$$\bar{P}_{\text{spokes}}(r,k) = \frac{P_t \lambda^2 |G_{\max}|^2}{(4\pi)^3 \sqrt{\pi \frac{c}{B}}} \sum_{m=-M}^{M} e^{-\frac{(r-m)^2}{\left(\frac{1}{B}\right)^2}} \sum_{n_x=0}^{L \cdot W} \sum_{n_y=0}^{H} \frac{G\left(n_x + k \cdot \frac{L \cdot W}{h}, n_y\right)^2}{R(n_x, n_y)^4}. \tag{2.4}$$

With the radar bandwidth B, speed of light in vacuum c, and window size 2M, this allows for a more realistic blurry radar plot.

The limitations of these models can be observed from the simulations in Vasstein et al. [2020] presented in Figure 2.8, Figure 2.9, Figure 2.10, and Figure 2.11.

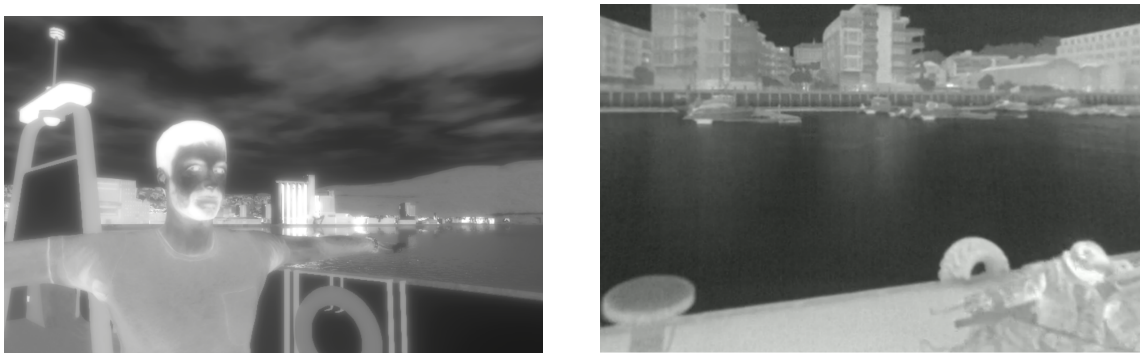


Figure 2.8: Simulated IR camera on the left and real IR image on the right. Courtesy: Vasstein et al. [2020].

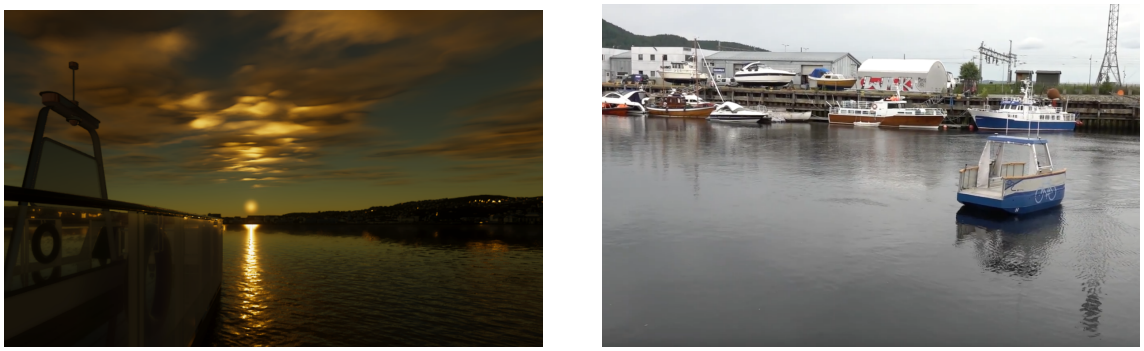


Figure 2.9: Simulated VL camera on the left and real VL image on the right. Courtesy: Vasstein et al. [2020].

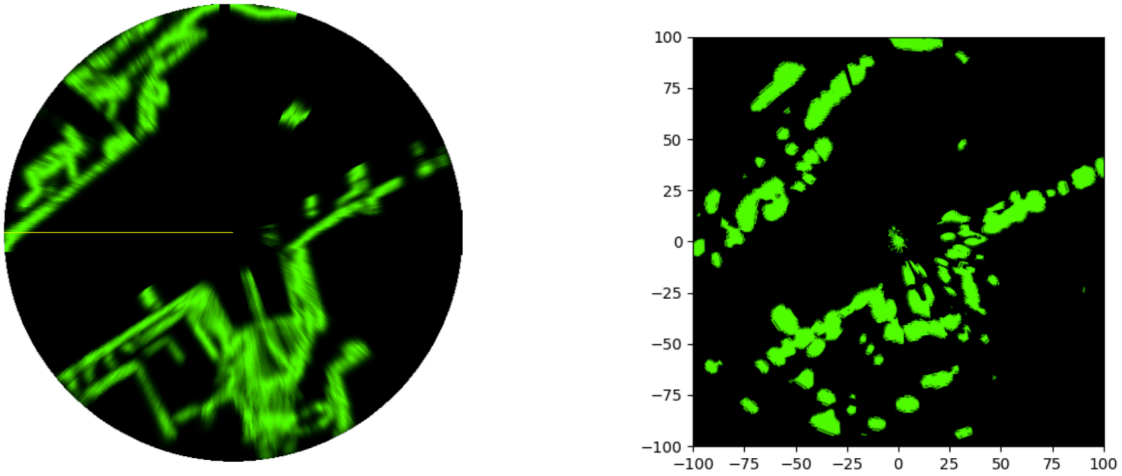


Figure 2.10: Simulated radar on the left and real radar image on the right. Courtesy: Vasstein et al. [2020].

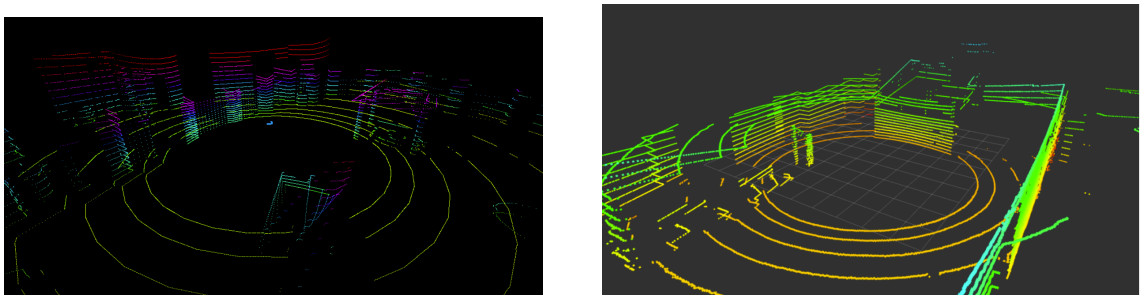


Figure 2.11: Simulated LiDAR on the left and real LiDAR pointcloud on the right. Courtesy: Vasstein et al. [2020].

As seen in Figure 2.8, the human face does not match reality, since the face should be lighter than the beard. This limitation is due to using artistic textures, because of no available empirical data. This means that the assumptions made in the emissivity model in Equation 2.1 are not sufficient and need further work. The lighting, however, is close to real IR photographs. The LiDAR, as seen in Figure 2.11, works as it should and

maintains the simulation framerate, but the performance and accuracy was decreasing when increasing the number of cameras. This could be due to the render pipeline having troubles handling many cameras. The radar plot in Figure 2.10 shows that the simulated radar plot is too accurate compared to the real radar data. The main difference is that the real radar data has missing data. This could be accommodated for, by removing data at randomized intervals.

2.3.3 Robot operating system

Robot Operating System (ROS) is a set of software libraries and tools that originated at Stanford in 2007. This framework enables people to build robotic systems without having to spend much time on writing the software infrastructure for the different independent subsystems of a robot [Quigley et al., 2015].

The ROS system consists of independent processes that perform computations, called nodes. These nodes communicate with each other in a peer-to-peer communication by passing messages. These messages are not sent directly from one node to another. A node sends the message by publishing the message to a "topic". Another node that is interested in that message then subscribes to that "topic". A node can publish and/or subscribe to multiple topics, but will generally not be aware of the other nodes.

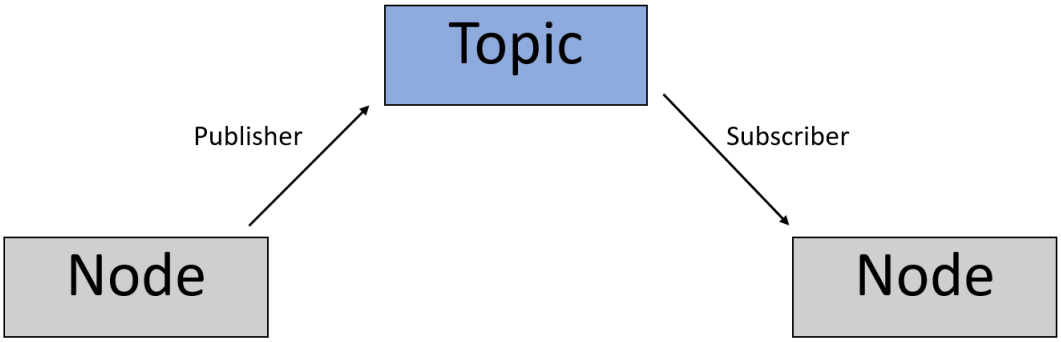


Figure 2.12: Message passing architecture in ROS.

The advantage of ROS is that the messages are standardized, using primitive types (integer, floating point, boolean, etc.) or arrays of those primitive types. This means that nodes can be written independently with different programming languages such as C++ and Python and still be able to communicate with each other [Quigley, 2009].

2.4 Situational awareness and safe maneuvering

A situational awareness module for an autonomous vessel will take the input from the sensors, process the signals, and create a representation of the surrounding environment.

Common sensors used for autonomous marine vessels for SA can be divided into 4 sensor families: visual (camera), remote sensing (RADAR and LiDAR), audio (microphone) and localization (satellite navigation, AIS, and Inertial Navigation System (INS)). There are also other sensors that can be used, e.g., depth-sensors, 3- dimensional Sound Navigation and Ranging (SONAR), and Radio Direction Finder (RDF). The range of the detection zone can affect the choice of sensors, e.g., AIS, RADAR, and stereo camera (in good weather conditions) for long range above 1 Nautical Mile, or LiDAR, cameras, and microphones for close range under 1 Nautical Mile.

The primary source of position and timing for vessels today are Global Navigation Satellite Systems (GNSS). By integrating an Inertial Measurement Unit(IMU), to determine heading, roll and pitch, in addition to the GNSS, a precise position of the vessel can be derived.

Object detection can be done with one or multiple cameras. Classical Convolutional Neural Networks (CNN) are designed for image classification and not object detection. A workaround, is to use a small sliding window to recursively search for objects on different areas in a large image. The disadvantage is that this costs a lot of computational power and the size of the sliding window has to be determined beforehand. This problem can be solved by using a Region Proposal Network (RPN) by Ren et al. [2015], which reduces the search for objects by giving a potential area of objects to appear. This means that instead of searching the whole picture, the object detection algorithm searches in some parts of the picture where there is a high chance of detecting an object.

2.4.1 LiDAR (Light detection and ranging)

One sensor that has been implemented on the CSS at the MC-Lab is a LiDAR. A LiDAR is an electro-optical sensor that measures the distance to an object, by sending out a laser pulse and measuring the duration between when the signal is sent and when the signal is bounced back to the receiver.

There are different types of LiDARs. The CSS is equipped with a low cost 360° 2D laser scanner with adjustable rotation speed. A 3D LiDAR uses spherical coordinates to describe the measurements. These measurements can be transformed into cartesian coordinates by using

$$x = r \sin\phi \cos\theta \tag{2.5}$$

$$y = r \cos\phi \sin\theta \tag{2.6}$$

$$z = r \cos\phi, \tag{2.7}$$

where ϕ is the elevation angle in the YZ-plane measured from positive z-axis, and θ is the azimuth angle in the XY-plane measured from the positive x-axis as shown in Figure 2.13.

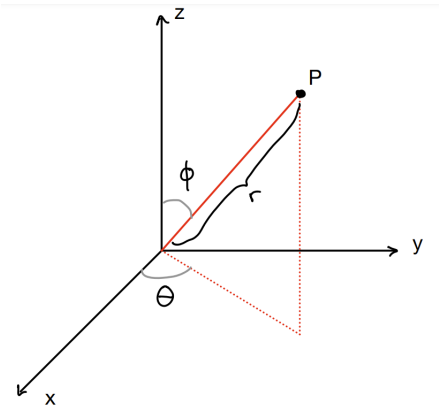


Figure 2.13: Spherical coordinates to Cartesian

Due to the fact that the LiDAR on the CSS is a 2D laser scanner, these equations can be simplified to

$$x = r \sin\phi \tag{2.8}$$

$$y = r \cos\phi \tag{2.9}$$

$$z = constant. \tag{2.10}$$

The range r in this case can be determined by

$$r = \frac{c}{2}(t - t_0), \tag{2.11}$$

where t_0 is the transmission time and t is the receiver time [Collis, 1970].

2.4.2 Camera

The CSS at the MC-Lab is also equipped with a HQ camera. A camera is an electro-optical sensor that projects the 3D environment in its field of view into a 2D image. This can be modelled by the *pinhole camera model* by Hartley and Zisserman [2004].

The pinhole model maps a point in space with coordinates $(X, Y, Z)^T$ to the image plane $(\frac{fX}{Z}, \frac{fY}{Z}, f)$ as shown in Figure 2.14.

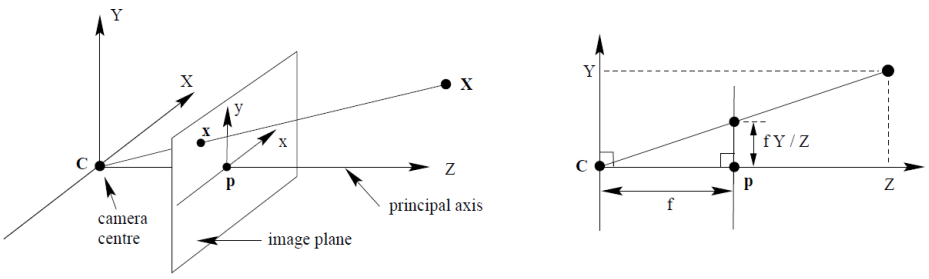


Figure 2.14: Pinhole model. Courtesy: HeidiVision [2019].

This mapping can be expressed as a linear mapping by matrix multiplication if homogeneous coordinates are assumed. A homogeneous coordinate system uses a fourth coordinate W in addition to X, Y and Z coordinates to represent the distance between the projector and the camera. For the *pinhole model* the value of this coordinate is set to one, such that the mapping becomes:

$$\begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{2.12}$$

This model assumes that the origin of the coordinates in the image plane is at the principal point **p**. However, this may not always be the case, such that the first matrix on the right side of Equation 2.12 is exchanged with a *camera calibration matrix* called **K**. This leads to the new system,

$$\begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} = \begin{bmatrix} f_x & 0 & p_x & 0 \\ 0 & f_y & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}. \tag{2.13}$$

The points in space defined by the *world coordinate frame* need to be transformed to the camera coordinates as well. This relation can be expressed by using a 3 x 3 rotation matrix **R** and 3 x 1 translation matrix **T**. This leads to the new system,

$$\mathbf{x}_{camera} = \mathbf{K}[\mathbf{RT}]\mathbf{x}. \tag{2.14}$$

Images cannot only be modelled linearly. They are also affected by nonlinear disturbances such as distortion. Two main types of distortions are radial distortion and tangential distortion. Radial distortion are caused by light rays bending more around the lens edges than the optical center, while tangential distortion is caused by the lens and sensor not being placed parallel to each other, causing a skewed image. A model of the distortion is presented by Heikkila and Silven [1997]. The radial distortion can be approximated using

the following expression,

$$\begin{bmatrix} \delta x \\ \delta y \end{bmatrix} = \begin{bmatrix} \tilde{x}\left(k_1 r^2 + k_2 r^4 + \ldots\right) \\ \tilde{y}\left(k_1 r^2 + k_2 r^4 + \ldots\right) \end{bmatrix}, \tag{2.15}$$

where $[\delta x\ \delta y]^T$ is the distortion, $[\tilde{x}\ \tilde{y}]$ is the image coordinates, $k_1, k_2 ...$ are coefficients for the radial distortion, and $r = \sqrt{\tilde{x}^2 + \tilde{y}^2}$. The tangential distortion can be modelled by

$$\begin{bmatrix} \delta x \\ \delta y \end{bmatrix} = \begin{bmatrix} 2p_1\tilde{x}\tilde{y} + p_2\left(r^2 + 2\tilde{x}^2\right) \\ p_2\left(r^2 + 2\tilde{x}^2\right) + 2p_1\tilde{x}\tilde{y} \end{bmatrix}, \tag{2.16}$$

where p_1 and p_2 are coefficients to be determined.

2.4.3 Control Barrier Functions

Control barrier functions (CBF) are a new tool to examine the safety of a control systems presented in Ames et al. [2019]. Similar to Lyapunov functions that guarantee the safety of a control system by guaranteeing stability, this function guarantees the safety by restricting the system to a given set that is considered to be safe and guaranteeing that it never leaves the set.

A generic nonlinear control system can be presented in the form

$$\dot{x} = f(x) + g(x)u, \quad x(0) = 0 \tag{2.17}$$

Here $x \in \mathbb{R}^n$ is the state vector and $u \in \mathbb{R}^m$ is the input vector. $f : \mathbb{R}^n \to \mathbb{R}^n$ and $g : \mathbb{R}^m \to \mathbb{R}^m$ are locally Lipschitz. This is used to set the restrictions

$$\begin{aligned} \mathcal{C} &= \{x \in \mathbb{R}^n : h(x) \geq 0\} \\ \partial\mathcal{C} &= \{x \in \mathbb{R}^n : h(x) = 0\} \\ \text{Int}(\mathcal{C}) &= \{x \in \mathbb{R}^n : h(x) > 0\} \end{aligned} \tag{2.18}$$

where $\mathcal{C}$ is the set that is considered to be safe, $h : \mathbb{R}^n \to \mathbb{R}^n$ is the barrier function, and Int is the interior of the set $\mathcal{C}$. By keeping the barrier function positive in the interior of the safe set, safety is guaranteed [Ames et al., 2019]. An example of a CBF implementation is Solheim [2022], where a CBF is implemented in the guidance system of CSS to avoid an obstacle.

2.5 Laboratories for autonomous marine systems

There exists multiple laboratories owned by NTNU in order to research, design and test autonomous marine vessels. This section will have a closer look into the available laboratories today and laboratories that will be available in the future.

2.5.1 MC-Lab

The marine cybernetics lab (MC-lab) is a small wave basin suited for tests of motion control systems for marine vessels, due to the small size and advanced instrumentation package. The basin has dimensions:

Length	Width	Depth
40m	6.45m	1.5m

Table 2.1: Dimensions for the MC-Lab basin.

The fixed equipments are:

- Qualisys Motion Capture System
- Towing Carriage
- Wave Generator
- Video-Camera [git, 2022]

The Qualisys Motion Capture System is able to provide 6 degrees of freedom data tracking with millimeter precision in real-time at 50Hz. The system consists of three Oqus high speed infrared cameras, that run on a Peer-to-Peer (P2P) network transmitting camera data to a computer running Qualisys Track Manager (QTM) software [git, 2022]. The cameras do not have fixed position and orientation. This means that the origin of the coordinate system for QTM is not the same each time the system is run. In the future, these cameras will be will be exchanged with fixed cameras [Premraj and Opland, 2022].

The wave maker is a single paddle wave making machine controlled by a dedicated computer. This generator has a width of 6 meters and is equipped with Active Wave Absorption Control System (AWACS 2).

There are several small vessels that are used in the MC-Lab:

- CS Enterprise I
- CS Saucer

- CS Arctic Drillship

CS Enterprise I was initially bought in 2009 by NTNU, and later refitted in 2011 [Skåtun, 2011]. The main use of CSEI in the past years have been DP systems, maneuvering systems and path following, and navigation with virtual reality. The vessel is fitted with two Voith Schneider propellers (VSP) astern and one bow thruster (BT). The main dimensions of the vessel are:

Length over all	Breadth	Displacement $\triangle$
1.105 m	0.248 m	14.11 kg

Table 2.2: Dimensions for CSEI

The CS Saucer was initially designed by Idland [2015], in collaboration with Ph.D. candidate Andreas Reason Dahl. The CSS has a symmetric, circular hull with a diameter of 0.55m. It was designed as such to be highly maneuverable behaving similarly in surge and sway, thus yielding quicker response and maneuverability than a conventional ship. The vessel has interchangeable top modules to allow different payload configurations. The vessel was attached with a LiDAR and camera sensor by Solheim [2022].

The CS Arctic Drillship was built in 2016 [Bjørnø, 2016]. The main focus of building the vessel was to perform research on Thruster-Assisted Position Mooring. The vessel is a 1:90 scale model of the inocean Cat I Arctic Drillship, and is equipped with 6 azimuth thruster (3 fore and 3 aft), in addition to a moonpool for turret and mooring lines. The main dimensions of the vessel are:

Length over all	Breadth	Displacement $\triangle$
2.5785 m	0.440 m	127.92 kg

Table 2.3: Dimensions for CSAD

2.5.2 Ocean Systems Lab

The Ocean Systems Lab (OS-Lab) located at the NTNU Department of Marine Technology is installed with two computers running:

- Graphics Processing Unit (GPU): Nvidia GeForce RTX 3070 Ti
- Central Processing Unit (CPU): AMD Ryzen 9 5950X (16 cores, 3.4 GHz)
- Random Access Memory (RAM): 32 GB

There is also a large screen wall with two virtual reality headsets and a 360°camera for 3-dimensional visualization:

- Oculus Quest 2, 128 GB
- GoPro Max [OSL, 2023]

The machines are also already installed with software and repositories such as [OSL, 2023]:

Software	**Repositories**	**Data Sources**
Programming: Python VS Code Matlab VirtualBox	Simulation: Open Simulation Platform DTU Ship Simulation Workbench Flettner rotor performance simulation Vessel.js Gemini	Ship design: System-based ship design (SBSD)
CAD & CAE: Siemens NX ShipX Rhino Simpack Abaqus PIAS	Artificial Intelligence: Image recognition Vessel speed prediction in icy waters	Operational data: Long-term wind statistics Gunnerus live data
Video & graphics: Unity Blender OBS Studio Oculus Software		

Table 2.4: Resource description from OSL [2023]

2.5.3 Shore Control lab

The Shore control Lab is owned by NTNU and located at the Trondheim Maritime Centre at Nyhavna. The primary focus of the lab is monitoring and control of autonomous maritime vehicles. There are several ongoing research projects using this lab such as:

- SFI (Center for reasearchdriven Innovation) AutoShip
- NTNU Autoferry
- LOAS (Land-based Operation of Autonomous Ships)
- IMAT (Integrated Maritime Autonomous Transport Systems)
- SAREPTA (Safety, autonomy, remote control and operations of industrial transport systems) [Sho, 2022]

2.5.4 Fjordlab/Trondhjem Biological Station (TBS)

The Trondhjem Biological Station (TBS) is a research facility with 600 m^2 of wet lab space, 800 m^2 of well equipped standard laboratories, and a constant supply of seawater from 100m depth. TBS has also access to the Gunnerus Research Vessel at NTNU and a Remotely Operated Vehicle (ROV), named "Minerva". These are mainly used for educational and research purposes [NTNU, 2022].

2.5.5 Ocean Lab

The Ocean Lab is an infrastructure which will be integrated to the future *Norwegian Ocean Technology Centre*. This infrastructure is made of 4 nodes [NTNU and SINTEF, 2022]:

1. Subsea Facility
2. National test area for autonomous vessels
3. Aquaculture
4. Marine observatory

The subsea infrastructure node acts as a platform for research, innovation, and qualification of technology focusing on underwater activities. The main research areas for this infrastucture are underwater operations, underwater robotics, and marine science. The assets that can be used in this infrastructure are [NTNU and SINTEF, 2022]:

- Existing assets
  - TBS (Trondhjem Biological Station)
  - RV Gunnerus
  - SDP (Subsea Docking Plate)
  - ROV Minerva
  - Autonomous underwater vehicles (AUVs)
  - PLM (Pig Loop Module)
- Assets in developement
  - Instrument rig for SDP
  - Instrument rig for PLM and Umbilical for PLM module
  - Upgrade for ROV Minerva II
  - Eelume
  - Control room

The national test area for autonomous ships is placed in Trondheimsfjorden and was established in 2016 by NTNU, SINTEF, Kongsberg Seatex, Kongsberg Maritime, Maritime Robotics, Port of Trondheim, The Norwegian Coastal Administration, and The Norwegian Maritime Authorities. Ocean lab will invest in sensors, technologies and relevant equipment for research to be used in the starting phase.

The aquaculture node is a full scale laboratory for aquaculture technology operating at four sites in Møre & Romsdal and Trøndelag. This node will be used for full scale research both with and without live fish, with sensors measuring [NTNU and SINTEF, 2022]:

- Weather conditions
- Salinity
- Turbidity
- Echo sounders
- Air temperature
- Wind speed/ gust and direction
- Air pressure
- Air humidity
- Oxygen saturation
- Current (speed and direction)
- Waves (height, period and direction)
- Temperature (water)
- Video and split beam (on 1 site)
- Load shackles on mooring system (on 2 sites)
- Accelerometer (or MRU) on net cage (on 2 sites)

The OceanLab Observatory make environmental research and sensor technology development in Trondheimsfjorden accessible to users around the world. This makes data easier to access and a lot cheaper, leading to faster improvements in research.

2.6 Remote control center

A RCC has two main applications; remote monitoring and control. The scope of this application can vary from system to system, depending on the goal of the RCC.

Remote control can have different applications, such as machinery control, navigation and piloting or operation of payload systems [DNV, 2018]. Navigation can be manual control of a vessel, setting desired waypoints for a DP system or setting another vessel as a reference vessel to be followed etc. The monitoring aspects of a RCC can be navigation, automatic reporting or maintenance schemes. Navigation in this context could refer to the position data of a vessel and automatic reporting could be fuel consumption or machinary status etc.

To be adequately efficient and reliable, sufficient information must be transferred from the vessel to the RCC, and vice versa, with appropriate frequency and data loss. This sets requirements on the type, volume and latency of the data transmitted. This is again, dependent on the system and will vary. For instance, a full scale container vessel in the ocean will have stricter requirements than a small vessel at the MC-Lab.

2.6.1 Network protocols

A network protocol is a set of rules that determine how data should be transmitted between different devices on the same network. This makes it possible for devices with different internal processes, structure, and designs to communicate with each other.

Some of the most used network protocols are User Datagram Protocol (UDP) and Transmission Control Protocol (TCP) [Jain, 2023]. TCP transmits data after a connection has been established. This allows for data to be transmitted both ways. UDP does not require a connection. This means that data is sent whether or not they are received by another unit. TCP on the other hand, checks for errors and guarantees that the data will be delivered in the correct order [Jain, 2023]. This protocol is therefore beneficial for transferring files, images, videos where it is more important that the transferred data is correct than the speed of the data transfer. As mentioned, UDP cannot guarantee that all data will be received. However, this protocol is much faster than TCP, due to not having to send an overhead for opening a connection, maintaining and then terminating the connection. This protocol is useful where real-time communication is required like broadcast and online gaming.

Another popular protocol is MQTT, which is an OASIS standard messaging protocol for the Internet of Things. This protocol uses lightweight publish and subscribe messages to connect remote devices with a small code footprint and minimal network bandwidth [mqtt.org, 2022]. This is very similar to the ROS message passing architecture in Figure 2.12.

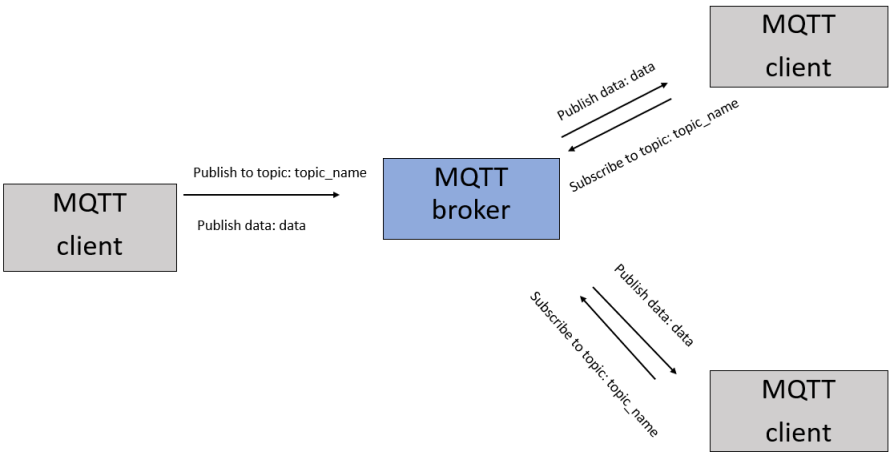


Figure 2.15: Example of MQTT message passing.

2.6.2 Remote Procedure Call

Remote procedure call (RPC) is a software communication protocol that can be used by a program that can request a service call from a program running on another computer on the same network. This procedure uses a client-server model, where the client requests data that is provided by the server [Rosencrance and Matturro, 2021]. One request call is shown in Figure 2.16.

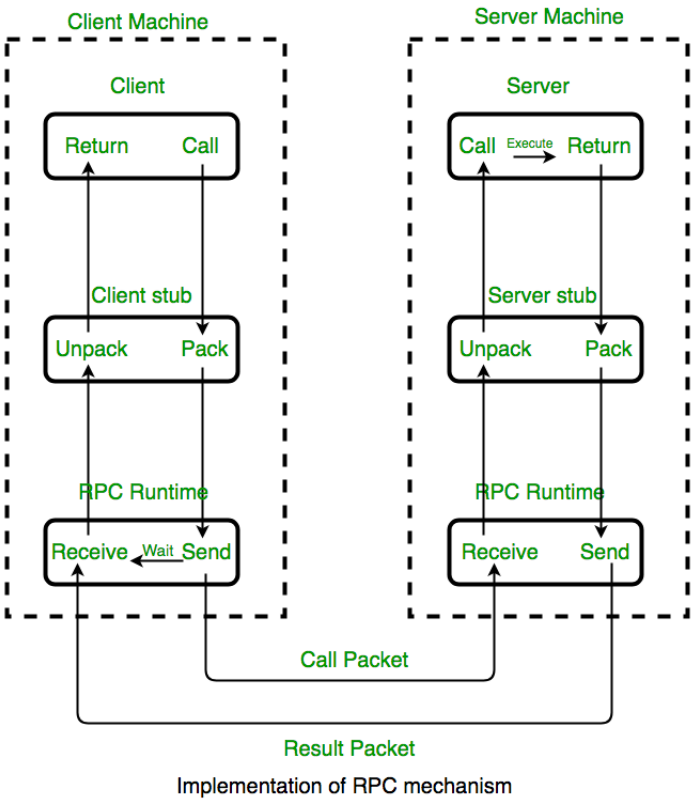


Figure 2.16: RPC framework, Courtsy: rpc [2023].

The client invokes a client stub procedure that handles the interface between a local client and the run-time system. The run-time system is the engine that translates the programming language into machine code. This machine code is sent to the server through the server run-timesystem and server stub. When the server received the request call, the server executes the call request and sends the requested data through the same sub systems [Rosencrance and Matturro, 2021].

Chapter 3

Problem formulation

The overall goal of this thesis is to recreate the MC-Lab and selected ROS-enabled cybership vessels, such as CSEI, CSS, and CSAD in Gemini's 3D simulated environment and emulate RCC operations of the vessels and their control systems. These cyberships will also be fitted with virtual sensors such as camera and LiDAR. This system consists of different subsystems that can be solved and tested separately. It includes the following sub-systems; virtual testing platform, the 3D models, and virtual sensors in Gemini, remote control and monitoring of the physical vessels at the MC-lab from OS-lab.

3.1 Digital twin

The MC-Lab needs to be recreated in Unity. This involves 3D models of both the cyberships and the environment. The models don't need to be very detailed to be visually pleasing, but the scale of the models need to be correctly proportional to the physical lab.

Virtual sensors need to be attached to each cybership. These sensors need to be adjusted to fit each ship. The LiDAR scanner needs to be placed and the minimum search distance needs to be adjusted to avoid the sides of the cyberships themselves being detected. The same applies for the cameras that need to be placed correctly, in order to avoid the deck of the cyberships covering large parts of the image.

The virtual testing platform will require ROS and Gemini. ROS enables the developer to implement their own code and test their systems. A DT of the MC-Lab will make use of this function to implement control systems for the vessels at the MC-lab and create virtual simulators that output the behaviour of the vessels with e.g., position, velocity, or acceleration. Gemini will visualize the position data as 3D models of the vessels and environment, and output virtual sensor data based on the virtual environment.

ROS is only fully supported on Linux. Gemini which runs on Unity is created based on a

Windows operating system. This implies that the necessary data between the operating systems need to be transferred from one machine to another. What type of data that will be transferred is up to the usage of the DT. The main data from ROS would be position coordinates, mainly surge, sway and yaw, in order to visualize the 3D model of the cyberships in Gemini. Gemini will need to capture LiDAR and camera data from the virtual sensors and send it to ROS in order to visualize them and use them for different use cases, for instance object detection.

3.2 Remote control and monitoring from OS-Lab

The MC-Lab is connected to the public NTNU network called "Eduroam" and a local network called "MCLab". The qualisys system that captures the positioning data runs on the local MC-Lab network.

A connection needs to be established between the machines at the OS-Lab and the local network at the MC-Lab. This will result in the OS-Lab machines to attain access to the 6 DOF qualisys data of the physical vessels in the basin. It will also open up the opportunity to be able to send data, e.g., command signals for remote control operations, from the OS-Lab to the MC-lab.

The ROS simulators output the positional data as Odometry messages which consist of a header, frame ID, position with covariance and velocity with covariance [ros.org, 2022]. Only the positions without covariance are of relevance for Gemini. These values need to be extracted from the odometry message and sent in a new message that consists of only position.

Data from the OS-lab can be sent to the MC-Lab with TCP or UDP. These protocols send streams of bytes. Messages consisting of bytes, need to be decoded and converted to the desired variable type. One example is an integer from one machine that needs to be converted into bytes and sent using TCP or UDP, and the receiving machine needs to decode this message and convert it back to an integer to be able to use it for the desired purpose.

3.3 Limitations and assumptions

With regards to time management and providing an adequate solution to the problem based on the scope of the thesis, some limitations have been identified. Some of the limitations are:

- The qualisys cameras at the MC-Lab reset the origin of the coordinate system every time the qualisys system is booted.

- The CSS has not been in use since last year. Fixing problems with the thrusters and running the physical vessel will consume a lot of time.
- The existing python simulators for CSEI and CSS are only 3DOF simulators regarding surge, sway, and yaw, that neglect heave, roll, and pitch. This means that comparing physical tests with the waves will not be relevant.

With regards to the limitations, some assumptions need to be made. These assumptions are:

- Since the origin of the coordinate system changes every time the system is run, the position of the virtual vessels placed roughly measured by eye will have sufficient accuracy.
- Physical tests will only be performed with one physical vessel, and will be a sufficient indicator that monitoring and remote control will work for the other vessels as well.
- The physical operations will be performed in a calm sea state with negligible waves, assuming zero pitch, heave and roll motions.

Chapter 4

Digital twin development of MC-Lab

This chapter presents the development of the DT of the MC-Lab. It includes the complexity and scope of the DT, the different subsystems and the information flow between them, and lastly the test cases that will evaluate the DT.

4.1 Scope of the Digital Twin

As mentioned in Section 2.1, the scope and complexity of a DT can vary. The applications of this DT is to visually represent the 3DOF (surge, sway, and yaw) behaviour of the physical vessels at the MC-Lab in a 3D environment. The DT's complexity is limited to the behaviour of the position of the hull of the vessels, and will not cover structural analysises of the hull or subsystems such as thrusters (e.g., angle of thrusters and RPM), electrical systems, or motor controllers.

4.2 Setup of the virtual platform

ROS runs on Linux, while Gemini runs on Windows. There are two options for running these two operating systems. One option is to run them on separate physical machines. Another option is to run a Windows-machine with a virtual machine, which is a compute resource that uses software instead of a physical computer to run programs. This virtual machine can run an operating system that is different from the physical machine. There are pros and cons to both options, and are presented in Table 4.1.

	Pros	Cons
Run virtual machine	Sensor data and positional data can be transferred at a higher frequency.	RAM and CPU usage is limited to one machine.
Run on two machines	The workload regarding RAM, GPU and CPU is distributed to two machines. This leads to better visualization in Gemini and ROS nodes running at a higher frequency.	Data transfer is limited to the network speed. Sensor data and positional data may need to be transferred at a lower frequency.

Table 4.1: Pros and cons of using virtual machines and two seperate machines

The general setup and data flow between ROS and Gemini is shown in Figure 4.1. ROS outputs 3DOF position and sends it to Gemini using RPC. RPC is also used to send sensor data from the virtual camera and LiDAR sensors in Gemini and is received by ROS.

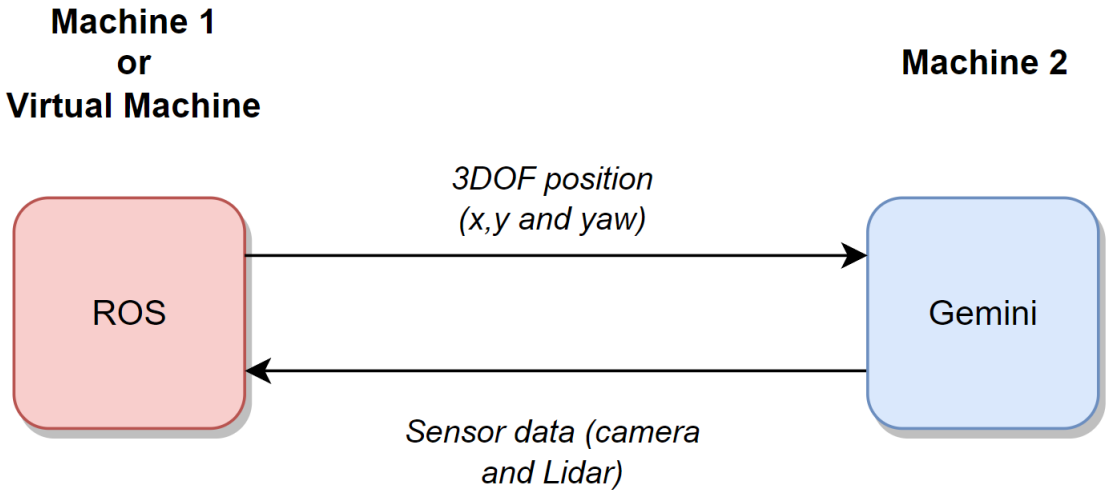


Figure 4.1: Setup of ROS and Gemini.

4.3 Gemini

The Gemini system contains the visual 3D models, and virtual sensors for the system. Scripts are included to change the position of the vessels in the visualization, but do not calculate any of the physical behaviour of the vessels.

4.3.1 3D models

The 3D models for the cybership vessels, such as CSEI, CSAD, and CSS are imported from git [2022] and shown in Figure 4.2, Figure 4.3 and Figure 4.4. These models are not as detailed as the physical vessels, but are sufficient to be recognizable by the hull shape and color.

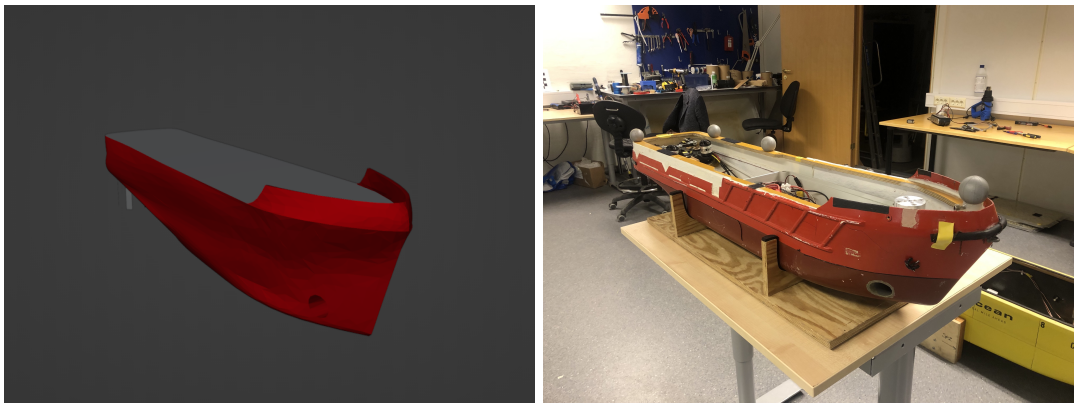


Figure 4.2: 3D model of CSEI and the physical vessel.

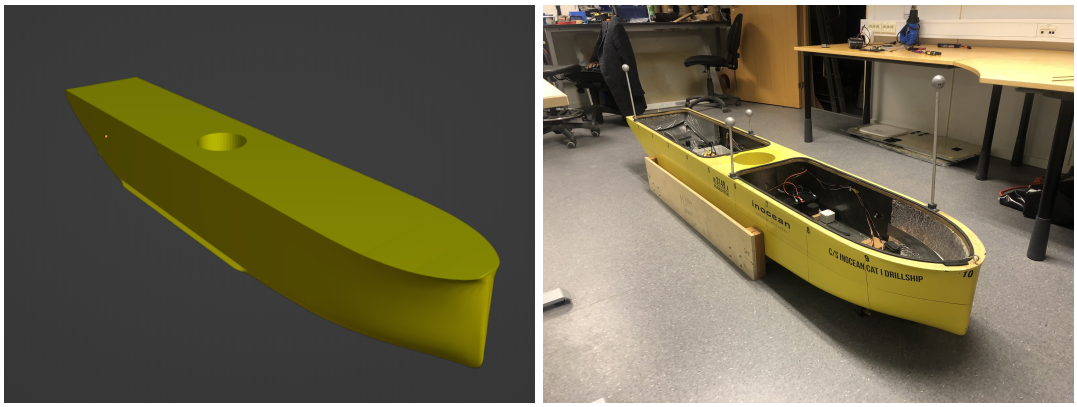


Figure 4.3: 3D model of CSAD and the physical vessel.

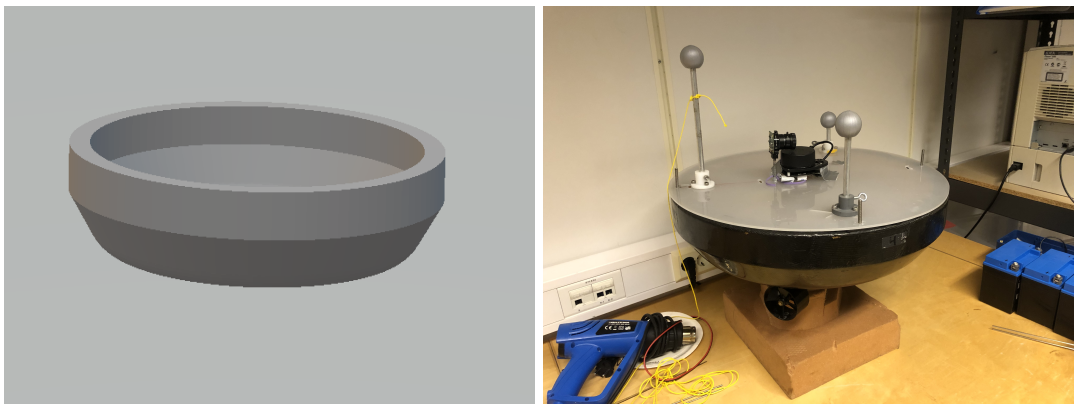


Figure 4.4: 3D model of CSS and the physical vessel.

These models are placed in a 3D model of the MC lab and scaled accordingly using the measurements from Section 2.5.1. The 3D model for the water has small irregular waves that do not affect the motions of the ships. This is only done to give a pleasing visualization of the water surface and is not an accurate representation of the waves in the real physical lab. Creating a realistic water surface with the correct wave height will be computationally heavy when running it in real-time. Since the ROS simulators are 3DOF simulators, a

correct visualization of the water surface is not important when assuming calm water with small to no waves.

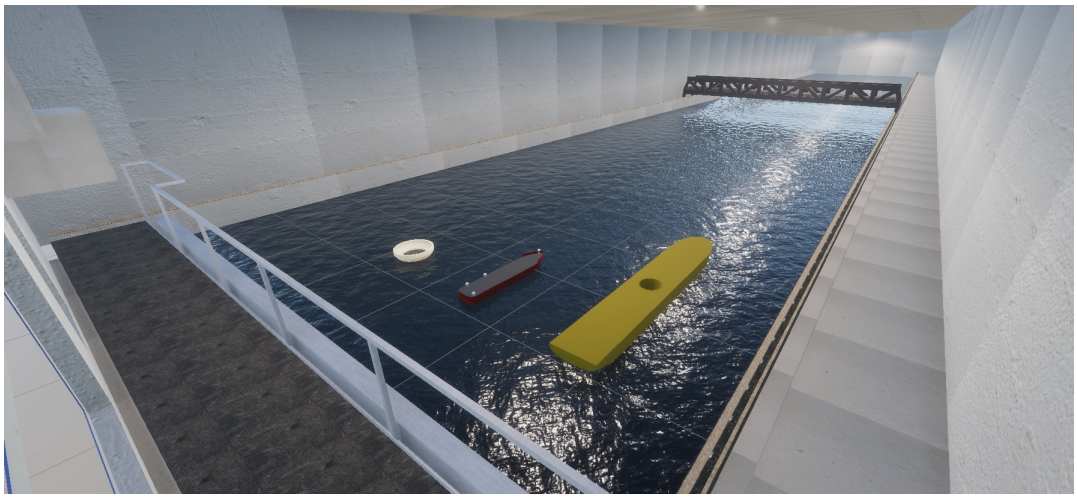


Figure 4.5: 3D model of the MC-Lab and cybership vessels.

4.3.2 Scripts

Several C# code scripts have been implemented to each vessel. These scripts add different features for the vessels that can easily be turned on and off by checking the boxes shown in Figure 4.6, both before running the simulator and during simulation in real-time.

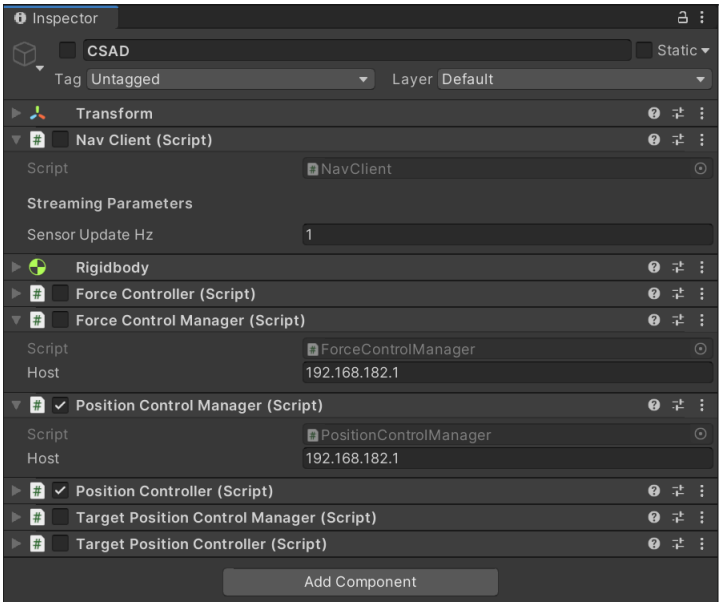


Figure 4.6: Inspector window for a vessel.

The “Nav Client” script sends the camera and LiDAR sensor data from Gemini to ROS. The streaming frequency can easily be chosen by the user.

The “Control Manager” scripts are the ones that allow ROS to send data to Gemini using

RPC. The "Force controller" script can add a constant force to the vessel using ROS. The force is added to the vessel in 6DOF (surge, sway, yaw, roll, pitch, and yaw) by publishing a list with 6 elements in ROS. This script uses the in-built physics in Unity, where the user can choose different parameters of the rigid body, such as mass, drag, and angular drag.

This may be sufficiently accurate for some objects in the real world, but not for vessels in water. An external simulator is needed to calculate the behaviour and output the position. This is where the "Position Controller" is used. Instead of forces in 6DOF, a list with 6 elements is sent from ROS describing the position of the vessels in 6DOF. The "Target Position Controller" script has the same feature as the "Position Controller", but is used for vessels that act as a reference vessel for other vessels.

4.3.3 Sensors

The camera and LiDAR sensors are placed on one single game object called "SensorRig" that can be moved around. This sensor rig consists of one 360° LiDAR and 4 cameras placed facing the front, back, port-side and starboard side of the vessel.

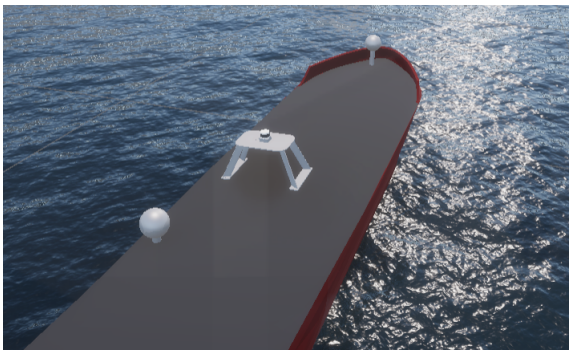


Figure 4.7: SensorRig on top of CSEI.

The LiDAR scanner can be customized by changing parameters such as minimum and maximum distances of the laser scans. The number of lasers can also be chosen, if the user wants one laser creating a 2D plot, or several lasers that can create a 3D scan of the environment. Real LiDARs can experience ray drop, where some signals are lost. This can also be customized in the virtual LiDAR by choosing the ray drop probability, where a ray drop probability of zero will give ideal LiDAR data with no missing data.

4.3.4 ROS architecture

Gemini is used for visualizing the vessels and simulating sensor data. The rest of the system is handled by ROS. The ROS system in this thesis can be divided into two subsystems;

a subsystem that handles everything from operator input to outputting the behaviour of the vessel, and a subsystem to handle the communication between ROS and Gemini.

There exists several ROS systems that handle the operator input and output the vessel behaviour. These systems serve different purposes and are interchangeable as long as they implement a simulator node that outputs 3DOF or 6DOF position data. Any ROS system will work, but this thesis is based on the system developed by Solheim [2022] for CSS. The focus in this system was to implement a LiDAR- and camera-based SA-system with control barrier functions (CBF) for safe motion control. The original ROS system uses real physical sensor data in order to detect objects and set reference positions to avoid objects. The new system removes the physical aspect of the sensors, and replaces it with virtual sensors.

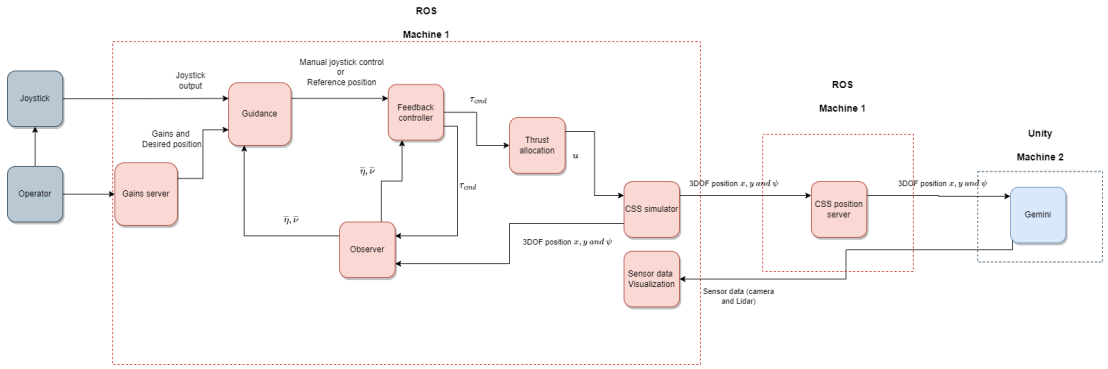


Figure 4.8: ROS architecture.

Figure 4.8 shows the node processes and message flow in ROS. $\bar{\eta}$ and $\bar{\nu}$ is the estimated position and velocity in surge, sway and yaw (x, y and ψ) in NED-frame, while τ_{cmd} is the desired forces and moments in BODY-frame and u is the allocated thrust force and angle for each thruster. Depending on the joystick input from the operator, the guidance node chooses if the control system uses manual control with joysticks or DP with reference positions. It is also possible to manually place obstacles with x and y position with a given radius, creating a path that the vessel needs to follow.

The CSS simulator can be exchanged with the physical vessel at the lab. In that scenario, the CSS simulator node in Figure 4.8 would simply be exchanged with the actuator drivers of the real vessel that receive the thrust allocation as input and qualisys that outputs the 6DOF position data.

4.4 Hybrid testing

A DT such as this, open up the possibility to do hybrid testing. This implies testing the system by connecting virtual and physical subsystems. The subsystems do not consider if the other subsystems are virtual or physical, as long as they provide the desired input

data. The connection between the virtual platform and the physical vessels are explained more in detail in Section 5.1. One example is to place virtual obstacles in the basin, and driving the real vessel. The virtual sensors provide the same type of data that a real sensor would do. This can enable obstacle avoidance systems to be tested on the physical vessels without damaging them if a collision was to occur. The control systems can also be run on the powerful machines on the OS lab and control the vessel remotely. Vessels that do not have sensors like LiDAR and Camera can simulate the sensors on Gemini while the real vessel is operating. This allows for control systems to get sensor data even without real sensors attached. It is possible to do so, since the environment in Gemini is created to represent the real MC-Lab.

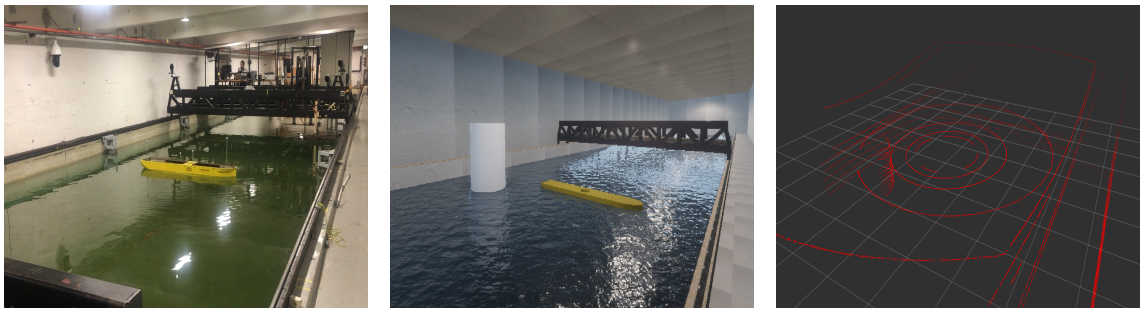


Figure 4.9: Figures of the physical vessel, virtual vessel with an object and the virtual LiDAR.

4.5 Cases

A DT such as this can have many different use cases. To asses the accuracy and efficiency of these sub systems, some test cases need to be executed, and are presented in chapter 6. These chosen tests are as following:

1. Test the accuracy of the virtual sensors data.
2. Joystick control of the vessel and visualize in Gemini.
3. Visualize a vessel following a reference vessel which follows a path that avoids an obstacle.

The first test is to make sure that the LiDAR- and camera-sensors give an accurate representation of the environment the virtual vessel is placed in. Different objects will be placed around the vessel and a side-by-side comparison should be done with the visualization in Gemini and the sensor data.

The second test will ensure that the manual joystick control in ROS works and that the vessel in Gemini follows the updated positions of the simulator. The focus of this test will

be that a change in x,y, or yaw in the simulator should result in a change in the vessel in Gemini.

Simulating an obstacle avoidance scenario with a reference vessel in the third test will show if the possibility of visualizing the position of two different vessels at the same time will be possible. Simulating the virtual sensors simultaneously will also ensure that the modification of the ROS system by Solheim [2022] using real sensors can be simulated with virtual obstacles.

4.6 Step by step guide

Running the DT requires one machine with Windows and a second machine with Linux. The Windows machine needs to download Unity and the required Gemini packages and models of the MC-Lab. The Linux machine needs to download ROS and the different workspaces needed. A more detailed guide to downloading and running the DT is given in Appendix A.

Chapter 5

Remote control center development

This chapter presents the development of the RCC function and includes the process of connecting the labs and the setup of the different subsystems. Lastly, the test cases for the RCC will be presented.

5.1 Connecting OS-Lab and MC-Lab

The two labs are connected using a Meraki SD-WAN by Cisco, to enable data transfer. Meraki SD-WAN is mostly used in companies with offices at different locations, to be able to set up one common company network.

Cisco firewalls are set up at both labs. These firewalls create a permanent VPN connection between the labs. There are also network switches that are connected to each firewall, enabling multiple devices to connect to the same network. The data transfer speed is dependent on the connection speed at the labs which is higher than the frequency of the ROS and Gemini system. This means that the transfer speed of the data between the labs will not affect the efficiency and accuracy of the remote control and monitoring applications using the DT.

5.2 ROS-architecture

The RCC function is based on the same ROS architecture and information flow shown in Figure 4.8. The difference lies in the data flow between the joystick node and guidance node when driving manual joystick control and between the operator and the gains server node when running DP scenarios. A flowchart of this submodule is shown in Figure 5.1.

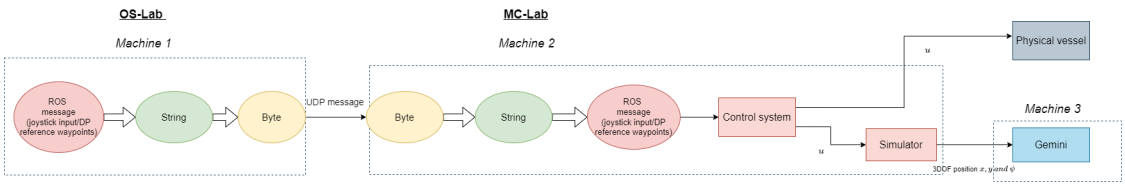


Figure 5.1: ROS-architecture of the remote control submodule.

TCP guarantees that the data is transmitted without loss, but this results in slower data transfer. Similar to online gaming, the real-time feature of the DT is prioritized higher than having a safe connection without any loss of data between the labs. In addition, UDP messages can be broadcasted such that multiple devices can receive the message and use it for different applications. Therefore, UDP messages are chosen to be the network protocol to send data between the labs.

UDP messages need to be sent in a byte format from the server machine. This means that, ROS messages firstly need to be formatted to strings and then to bytes. The client machine needs to receive this message and then convert it back to strings and finally publish them as ROS messages.

5.3 Remote monitoring

It is mentioned in Section 3.3 that the origin of the coordinate system is not fixed, and will change every time the QTM is run. Since new cameras will be fixed to the walls in the future, resulting in an origin that does not move, the calibration of the position of the vessel relative to the basin is done with measuring by eye. Due to the width of the basin being shorter than the length, the focus of the calibration is the position of the vessel with regards to the width of the basin.

The approach for calibrating the position for the tests in this thesis will be to move the vessel to one end of the basin and setting that as the origin (shown in Figure 5.2). This will be the approach with the best accuracy when measuring by eye. One downside of this approach is that the QTM does not cover the whole width of the basin. Position is lost if the vessel is too close to the wall. Therefore, the vessel needs to be placed slightly away from the wall until it is in range.

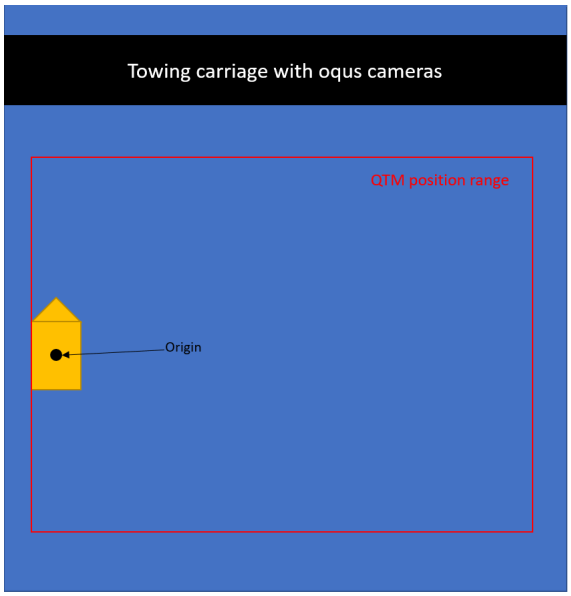


Figure 5.2: Calibration setup for the origin of the QTM coordinate system.

5.4 Interface for remote control

In Section 4.3.4 it is mentioned that the ROS system without the remote control when running a simulator can be divided into two submodules; one for outputting the behaviour of the vessel behaviour and one for the communication between ROS and Gemini. The ROS system with outputting vessel behaviour is interchangeble with different ROS systems for different vessels and application areas. For physical testing of the vessels, the output also changes from vessel behavior to for instance actuator commands. Different systems will have different inputs, and those inputs may be formatted differently. For instance, one systems that receive a list with three elements for 3DOF position (surge, sway, and yaw), may expect them as a 3 by 1 array, while another system expects the input to be a 1 by 3 array. There could also be differences in the order of the elements. Hence, two different ROS systems with remote joystick control and DP capabilities have been chosen to display the validity of this approach to RCCs. To show the validity on a simulator, the ROS system developed by Solheim [2022] for CSS is chosen. To show the validity on a physical vessel, the ROS system being developed by Midtun [2023] for CSAD is chosen.

5.4.1 Joystick control

The same DS4 controller is used to operate the CSS simulator and the physical CSAD vessel. Therefore the approach of the remote control operation with joystick control will be the same for both systems. A ROS message of a joystick input is built as such:

The axes input refer to the joystick on a DS4 controller and the two buttons on the back

Type	Name
Header	header
float32[ ]	axes
int32[ ]	buttons

Table 5.1: ROS message for joystick.

that control the surge, sway and yaw of the vessels. The axes input is a list of float between zero and one. Therefor the force input can be adjusted by the operator. The buttons input is a list of integers for the other buttons on the DS4 controller that are zero as default and one when they are pressed. These can be customized to utilize different functionalities, e.g., changing between joystick control and DP control by pressing the square button.

5.4.2 DP control

To showcase different options for the interface for a DP operator, different interface layouts have been chosen for CSAD and CSS.

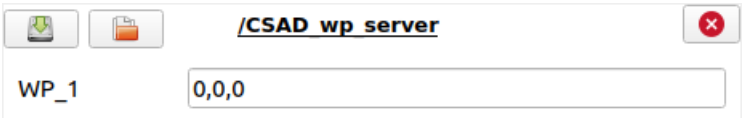


Figure 5.3: Interface for DP control of CSAD.

The interface for CSAD is shown in Figure 5.3. The three values, desired x- and y-position, and yaw angle, are set by the operator and sent to the controller. This interface is very simple and easy to use. The downside is that there is no descriptions for what the three values are in the interface. Therefor the operator must know beforehand, that the first value is the x-position, second value is y-position, and the last value is yaw angle.

Figure 5.4: Interface for DP control of CSS.

The interface for CSS, which is shown in Figure 5.4, has sliders that can be used to set different reference setpoints between a given interval. The size of the interval is not fixed to the values shown in Figure 5.4 and can be changed before running the system. There is also a possibility of setting the setpoints manually in the slots on the rights side without using the slider. This slider interface may seem like a better and more versatile option than the one for CSS, but has one clear downside. The reference positions are sent to the DP controller when a new setpoint is chosen. This means that, as soon as the value of

one variable is changed, the control system receives this change and tries to reach that value. Therefor, changing all the values for the three values at the same time will not be possible. There will be a delay between setting the x-position, y-position and the desired yaw angle depending on the speed of the operator that sets the desired value. Compared to this interface, the CSAD interface can update all three values at the same time without any delay. Depending on the DP operation, this delay may have significant effects on the safety of the operation.

5.5 Cases

The two main parts of an RCC is monitoring and remote control. To show the strength and limitations of the system, the chosen test cases are:

1. Visualize real-time odometry data from the physical vessel in the 3D environment
2. Remote joystick control of the CSS simulator and the physical CSAD vessel
3. Remote DP operation of a four corner test with the CSS simulator and the physical CSAD vessel

The goal of the thesis is to have the OS-Lab act as the RCC. Remote control and monitoring of the physical vessels can be done from any machine that is connected to the MC-Lab network. Due to the time physically traveling between the MC-Lab and OS-lab, and limited amount of people that are available to help out in executing these tests, only the remote joystick control of the physical CSAD vessel will be performed from the OS-lab. The rest of the tests will be done with one seperate machine at the MC-lab that acts as the RCC host and connects to the other machines at the lab.

Chapter 6

Results and discussion

The primary objective of the tests in this thesis is to assess the validity, efficiency and accuracy of the DT and the RCC. The details of the test cases have been presented in Section 4.5 and Section 5.5. This section chapter will present and discuss the results of the test cases that have been performed.

6.1 Digital twin demonstration

The tests in this section are done on one Windows machine with a virtual machine running Linux. The quality of the figures of the 3D models are dependent on the Graphics processing unit (GPU) of the machine. These tests are performed on a machine with an AMD Radeon GPU. Therefore, a machine with a better GPU will give a better resolution when running the same simulations.

6.1.1 Virtual sensors in Gemini

The virtual sensors are tested by placing different objects around the vessel. This is shown in Figure 6.1, where the CSEI is fitted with a LiDAR sensor and four cameras. The vessel is placed in the center of the basin with CSAD on the starboard side and a cylindrical object on port side.

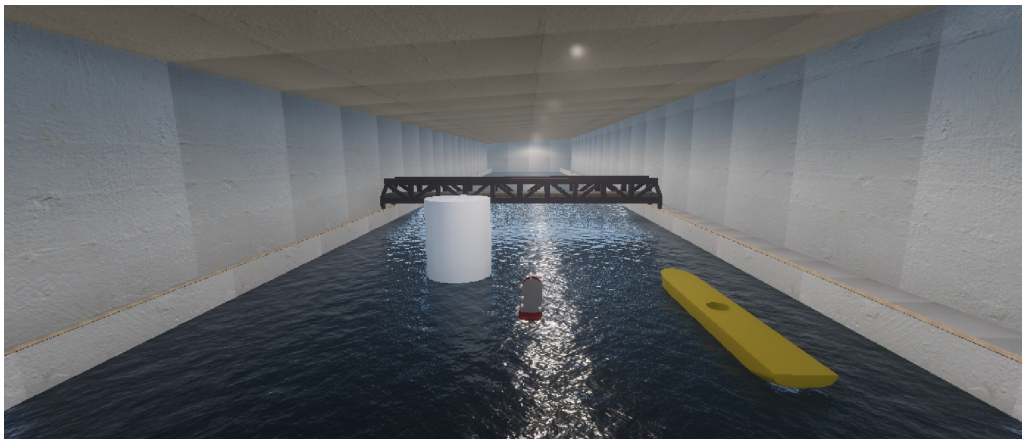


Figure 6.1: Bird's-eye view of the CSEI and environment.

In Figure 6.2 the LiDAR lasers can be seen after activating the sensor. The same laser data creates a 3D pointcloud visualization in ROS and is presented in Figure 6.3. The ray drop probability is set to 0.8 and the numbers of lasers is set to 12. The ray probability not being set to 1 shows the data loss in the pointcloud. The lines from the lasers are not fully continuous and contain points without data. Choosing a lower number of lasers results in more space between the laser closest to the vessel and the laser furthest away close to the ceiling. This can cause the sensor to unable to detect some obstacles, such as CSAD which is detected by only one of the lasers. Taking a closer look on the laser that detects CSAD, shows that the only difference between the lasers is the vertical angle. All lasers are given a minimum distance and maximum distance. As soon as those rays hit an object, they produce a red dot in the pointcloud. A taller object is therefore more probable to be detected by the LiDAR, than a wider object. This is also the reason for the circles close to CSEI to appear, due to hitting the water surface. Another important thing to notice are the lasers that hit the cylindrical object. The lasers placed at the same height as the sensorrig give parallel lines. The lasers higher on the cylinder are distorted and bent, due to the angle of the lasers. The same behavior can be seen on the walls, with close to parallel lines at the same height as the sensorrig and more bent lasers further up.

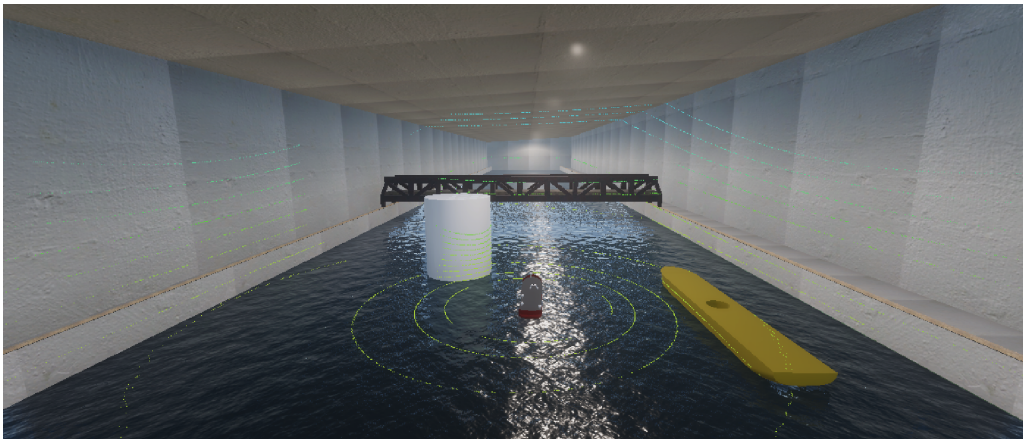


Figure 6.2: Bird's-eye view of the CSEI with LiDAR lasers.

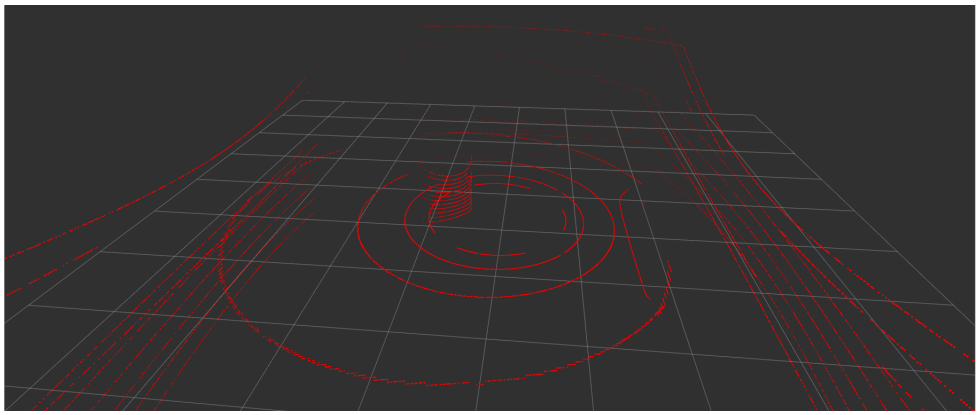


Figure 6.3: Pointcloud from the LiDAR sensor.

Images from the four cameras on the vessel are presented in Figure 6.4, Figure 6.5, Figure 6.6 and Figure 6.7. The placements of the cameras are chosen such that the deck of the CSEI is not shown. The downside with this placement is the large blindspots between the cameras. For instance, the cylindrical object is hidden and can't be found in any of the camera images. An object detection module purely based on the camera sensors, will not detect the cylinder and may cause the vessel to crash. To minimize the area of the blind spot, the space between the cameras should be reduced, even if it means that the deck of the vessel will occupy most of the image. Other settings on the camera can also be explored to increase the area the camera image can capture. It is also important to notice that this scenario is for a vessel standing still. A video of a moving vessel is presented in Figure 6.8 and will further discuss the sensor data.

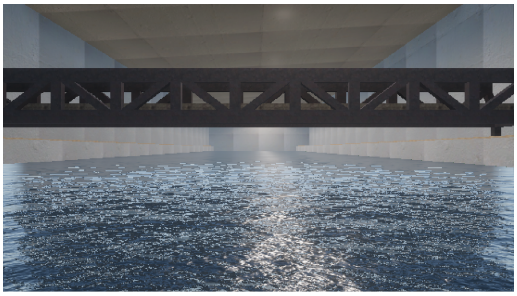


Figure 6.4: Front camera.

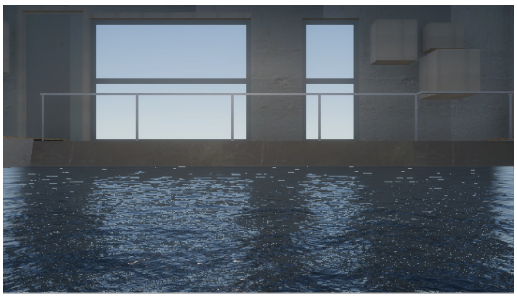


Figure 6.5: Back camera.

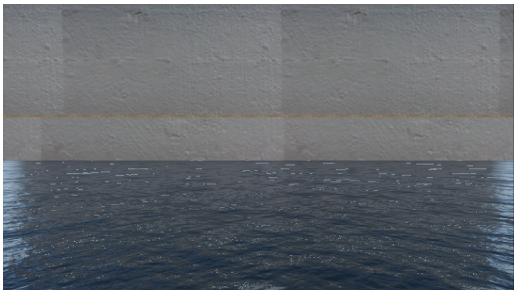


Figure 6.6: Port camera.

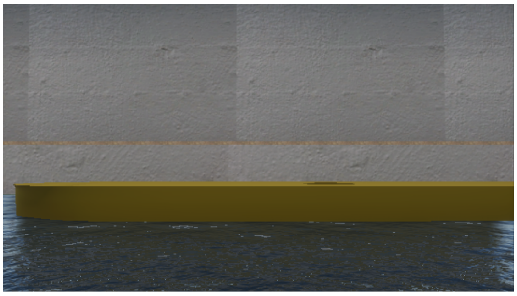


Figure 6.7: Starboard camera.

6.1.2 Obstacle avoidance test

The results for the test case is shown in Figure 6.9, Figure 6.10, Figure 6.11 and Figure 6.12. A video demonstrating the obstacle avoidance and reference vessel can be found through the link:

https://www.youtube.com/watch?v=Ar06t_qB898&ab_channel=JamesPremraj

or the QR-code:

Figure 6.8: QR-code for the video of the obstacle avoidance test.

Figure 6.10 shows the two different vessels, where the reference vessel has a lighter red color than the own vessel. The video in Figure 6.8 shows that the simulation has a higher frequency than the sensor data frequency. This is not optimal for high speed maneuvering, but is sufficient for low-speed operations. The frequency of the sensor data can be chosen, but is limited to the hardware of the machines and network speed. Turning off some of the cameras, can also reduce the size of data that is needed to be sent, resulting in the data to be able to be sent at a higher frequency. For example, in a bigger basin, the back camera and starboard camera would not be relevant since there is only one obstacle on the port side of the vessel. The LiDAR can also contain fewer laser scans. The obstacle is so tall that in this case, a single laser is sufficient to create a 2D plot of the obstacles around the vessel. Since the vessel is moving, the cylinder is not hidden in the blindspot during the whole simulation like in Section 6.1.1.

The positions and orientations of the vessel can be examined to be equivalent in Figure 6.9 and Figure 6.8. The reference vessel is not supposed to have any yaw motion, while the own vessel has some.

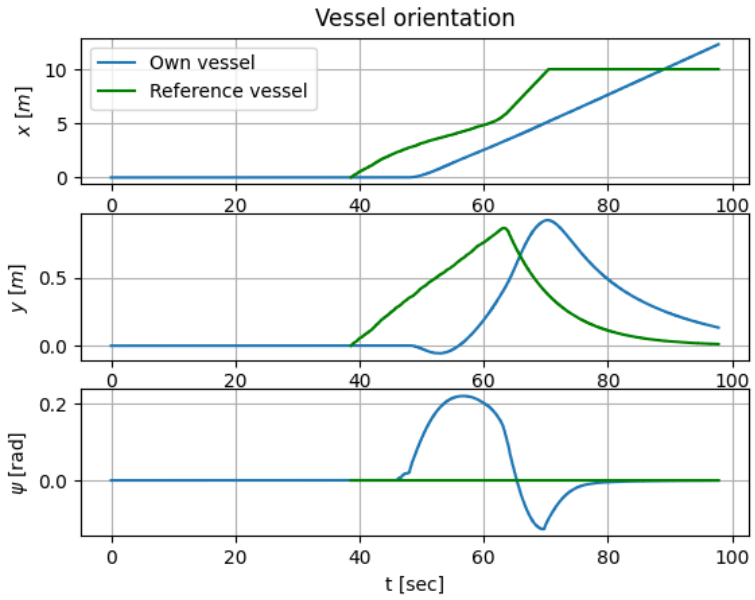


Figure 6.9: Vessel orientation in BODY-frame.

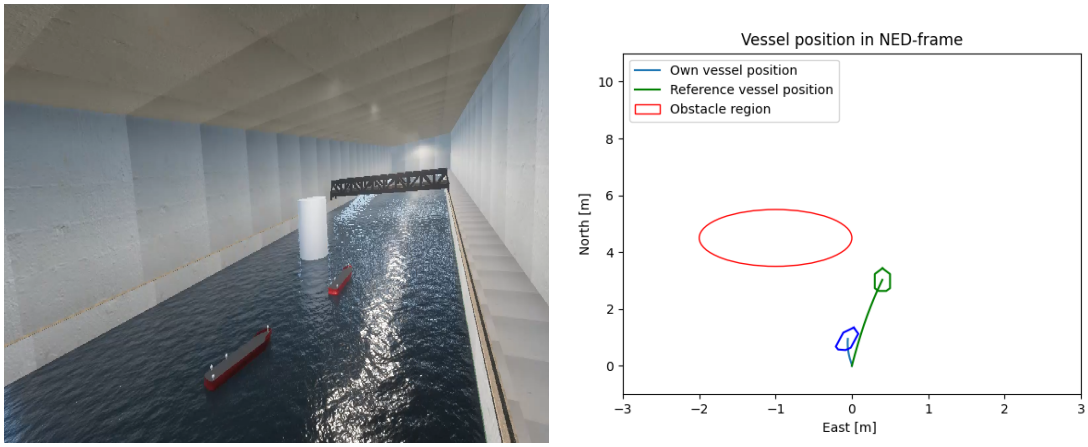


Figure 6.10: Vessel position in Gemini and NED-frame at time 55s.

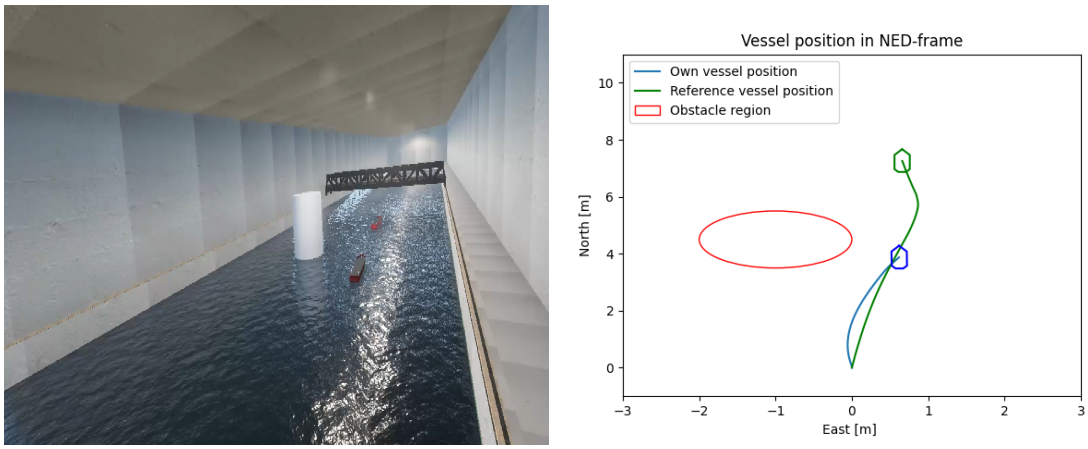


Figure 6.11: Vessel position in Gemini and NED-frame at time 63s.

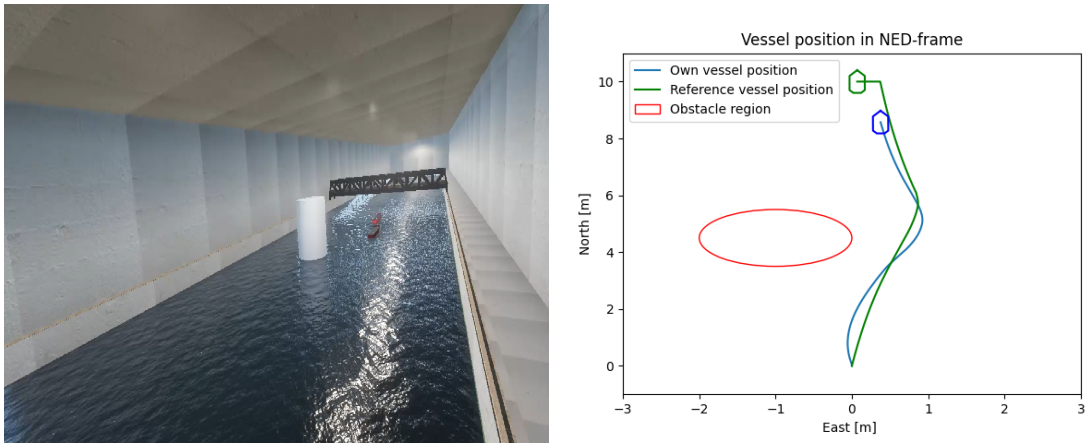


Figure 6.12: Vessel position in Gemini and NED-frame at time 75s.

6.1.3 Joystick control in simulation

The results of the virtual vessel controlled with a DS4 controller are shown in Figure 6.14, Figure 6.15 and Figure 6.16. A video of the vessel in Gemini can be found through the link:

https://www.youtube.com/watch?v=MVGyRjiWA6E&ab_channel=JamesPremraj

or the QR-code:

Figure 6.13: QR-code for the video of the virtual vessel controlled with a joystick.

The vessel position and orientation matches in both the video and Figure 6.16. The joystick inputs in x-direction result motion in x-direction in BODY-frame. The L2 button induces a yaw motion, which gives the vessel motions in both x- and y-direction.

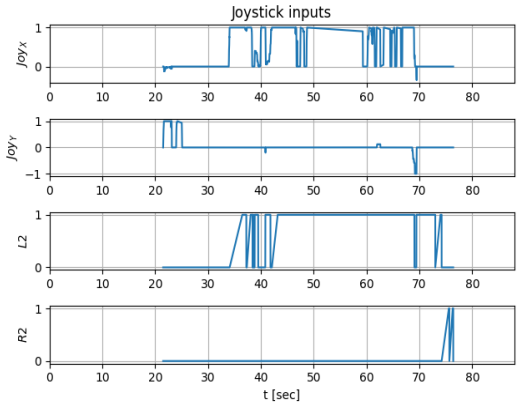


Figure 6.14: Joystick input.

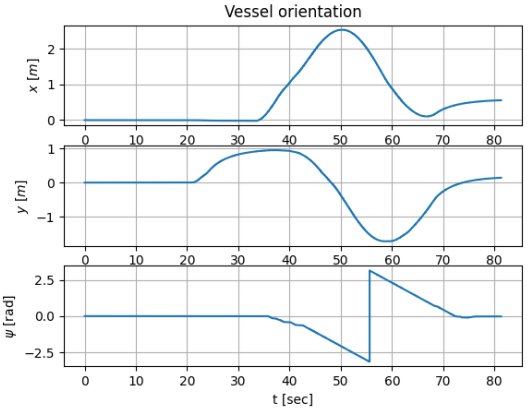


Figure 6.15: Vessel position in BODY-frame

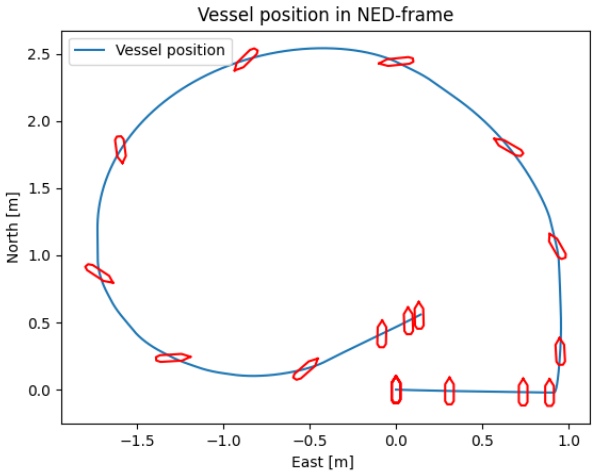


Figure 6.16: Vessel position in NED-frame.

6.2 RCC demonstration

It is important to note that the tests were performed on a machine at the MC-Lab connected to the network and acts as the host with the operator for the other machines. To showcase that the system works from locations further away from the lab, the test with joystick control of CSAD, in Section 6.2.2, was done with a machine at the OS-Lab as the host. The focus in these tests are not how good the thruster allocation or DP-system works, but rather how fast and accurate the signals from the operator are received by the machines or Raspberry Pis at the MC-Lab.

6.2.1 Visualizing real-time odometry in a 3D environment

A video demonstrating the remote monitoring function can be found through the link:

https://www.youtube.com/watch?v=gg_7djfzF4Q&ab_channel=JamesPremraj

or the QR-code:

Figure 6.17: QR-code for the video of the remote monitoring function.

The virtual vessel represents the position and motion of the physical vessel quite well, but it is not fully accurate. The motion of the vessel is correct, but the position is not identical due to the calibration. In Section 5.3, it is mentioned that the origin of the coordinate system changes every time the system is run, and the approach for calibrating the position of the vessel relative to the basin. In the video, it is possible to see that the virtual vessel is a little too close to the wall compared to the physical vessel. The approach of measuring by eye to calibrate the vessel is therefore not sufficient. However, the camera system for the QTM will soon be exchanged with fixed cameras after this thesis is written, which means that this will not be a problem in the future.

Adding and removing objects in the environment does not affect the vessel, which can be observed when removing the towing carriage in the video. The positional data from the physical vessel is not influenced by the virtual model, implying that virtual sensors on the virtual vessel can detect objects not present, and at the same time neglect unnecessary objects in the physical Lab.

One downside of the monitoring function is that there is a limitation to the frequency in the virtual environment. The positional data cannot be visualized at a high enough frequency in order to have a smooth and continuous simulation. This is due to the data being sent from the QTM machines at the MC-Lab through different machines and being formatted before being received by the Linux machine that runs Gemini. Increasing the frequency more than what is presented will lead to Gemini getting too many data sets and creating a queue. Consequently, the position of the vessel will be delayed and not give a real-time representation of the physical vessel.

6.2.2 Remote control with DS4 controller

The result of controlling the simulated CSS vessel with a DS4 controller is presented in Figure 6.18, Figure 6.19, and Figure 6.20. The motion of the vessel corresponds to the different joystick inputs. The L2 and R2 buttons result in the desired yaw movements, without any lag.

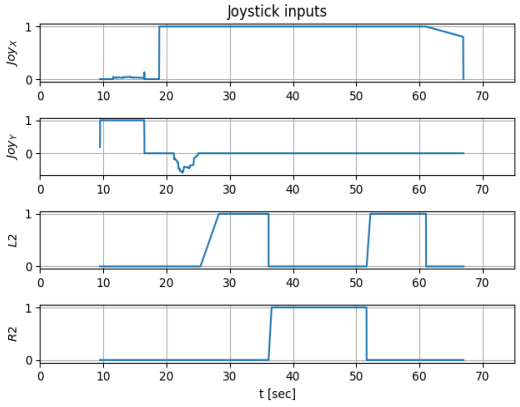


Figure 6.18: Joystick input for CSS using a DS4 controller.

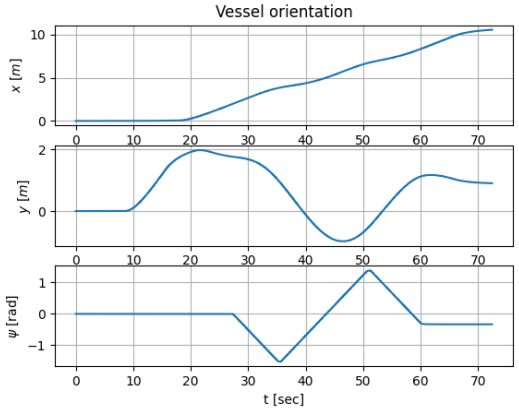


Figure 6.19: Vessel position of CSS in BODY-frame.

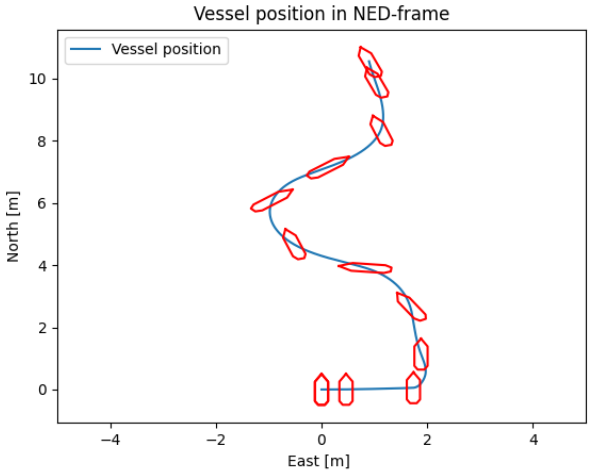


Figure 6.20: Vessel position of CSS in NED-frame.

The results of controlling the physical CSAD vessel with a DS4 controller are presented in Figure 6.21, Figure 6.22, and Figure 6.23. The vessel motion is not as smooth as of the simulation, but the joystick inputs match the motions of the vessel. The lag is also negligible even though the data is sent from the OS-Lab to the MC-Lab. This means that the data amount is small enough to not have any limitations regarding the frequency the signal is sent with.

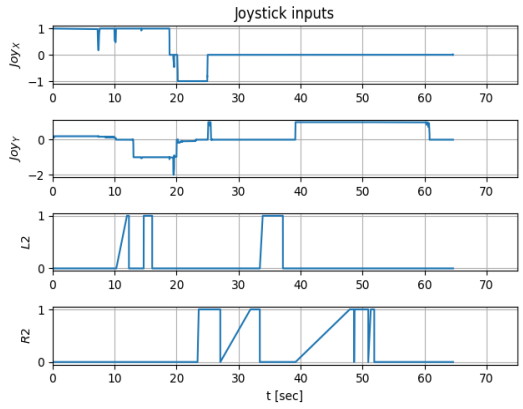


Figure 6.21: Joystick input for CSAD using a DS4 controller.

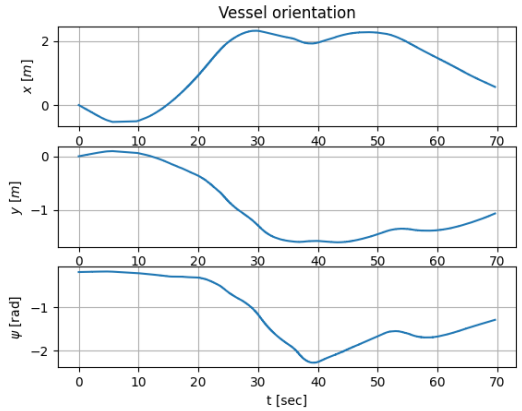


Figure 6.22: Vessel position of CSAD in BODY-frame.

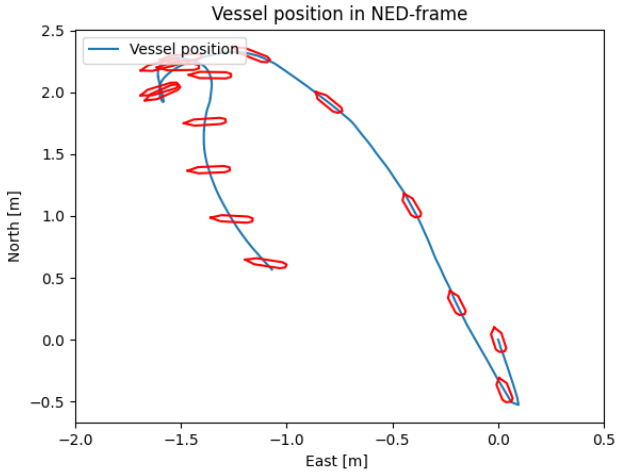


Figure 6.23: Vessel position of CSAD in NED-frame.

6.2.3 Remote DP operation

The results for the remote DP four corner test for the CSS simulator is shown in Figure 6.24, Figure 6.25, and Figure 6.26. There is minimal delay from when the operator chooses the reference positions and when the control system starts to move the vessel towards that position. Looking at the reference positions at around 300s in Figure 6.24, the desired x-position is chosen earlier than the yaw direction. This is due to the interface shown in Figure 5.4. It is not possible to set two positional parameters at the same time, but the delay can be minimized depending on the speed of the operator.

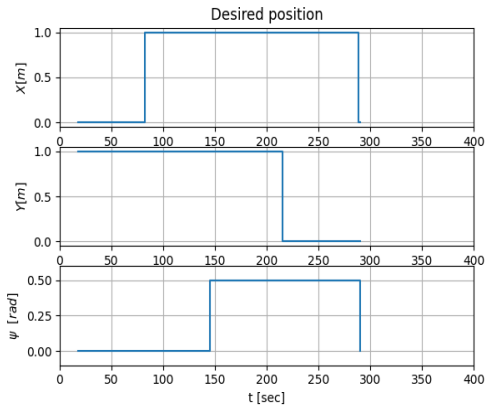


Figure 6.24: Reference position from operator.

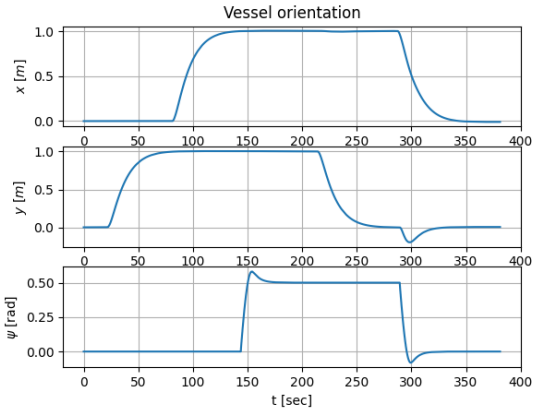


Figure 6.25: Vessel position of CSS in BODY-frame.

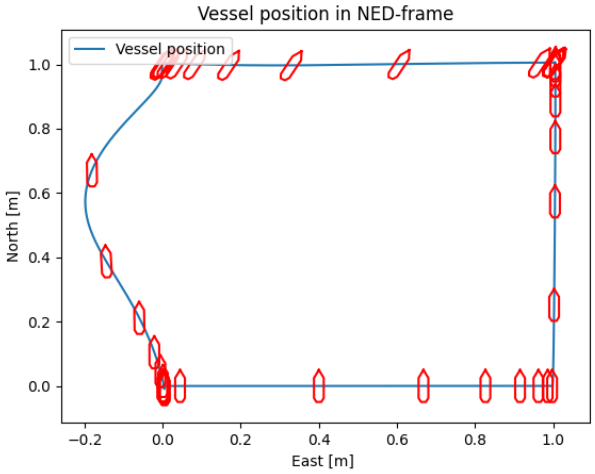


Figure 6.26: Vessel position in NED-frame.

The results for the remote DP four corner test for the physical CSAD vessel is shown in Figure 6.27, Figure 6.28, and Figure 6.29. The reference signals are received by the control system with a little delay (around 10-15s). This is due to the control system and thrusters on the vessel and not due to the network speed. At 500s, the y-direction and yaw-direction can be found to be set to zero at the exact same time. This is the difference between the interface for the CSS simulator and the CSAD vessel.

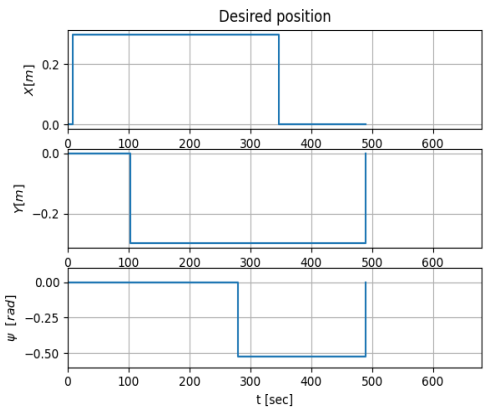


Figure 6.27: Reference position from operator.

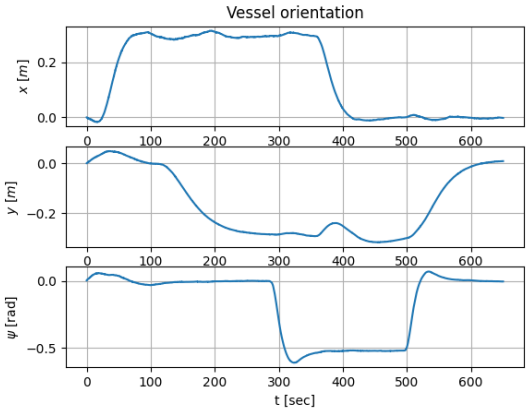


Figure 6.28: Vessel position of CSAD in BODY-frame.

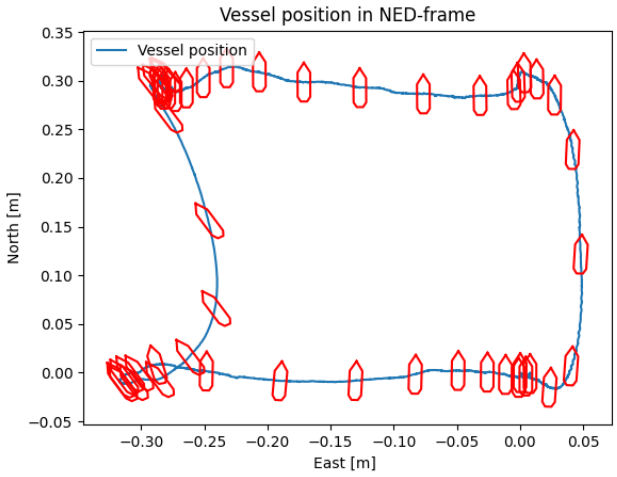


Figure 6.29: Vessel position in NED-frame.

6.3 Final discussion

The DT as a whole, is shown to be capable of visualizing 3D models of the cybership vessels and environment at the MC-Lab. The quality of the visualization is limited to the GPU of the machine that runs Gemini. Hardware limitation also affect the frequency that the sensor data are transferred with. The RCC function allows to monitor and control physical vessels at the MC-Lab from the OS-Lab. The network speed is not a limitation. The DP system for the physical CSAD vessel is slower than the CSS simulator to react to the updated waypoints from the RCC, but this is mainly due to the physical thrusters, control system and thruster mapping. This statement is backed up by the fact that the remote control with a DS4 controller does not experience any delay between the joystick input and the vessels motion. The only subsystem that is not sufficient enough is the position of the vessel when monitoring is not sufficient. There needs to be a better

approach to calibrating the position, but was not a priority when running the cases in this thesis, since the camera system will be exchanged with fixed cameras shortly after the semester this thesis will be written in.

Chapter 7

Conclusion

This thesis presents a DT of the MC-Lab and a RCC function, created with Gemini and ROS. 3D models of the cybership vessels, CSEI, CSAD, and CSS, are placed in a virtual model of the MC-Lab with attached LiDAR and camera sensors. C# scripts are also added to the vessels in order to be able to move them with the use of a DS4 controller and other inputs such as waypoints for a DP controller and external forces. Gemini is connected to a python simulator for CSEI and CSS in ROS, that sends the positional data of the vessels to be visualized in Gemini. The python simulators are 3DOF simulators in surge, sway, and yaw.

The RCC function is made up of two parts; remote control and monitoring. The host machine can be located anywhere as long as it is connected to the MC-Lab network. To access this network from the OS-Lab, a Meraki SD-WAN by Cisco is used. The data between the labs is transferred by using UDP messages. The desired data such as positional odometry messages when monitoring and joystick input during remote control needs to be formatted into bytes when sending and then reformatted back to the original datatype by the receiver.

Adding the RCC function to the DT, opens up the opportunity to be able to perform hybrid tests. The physical vessel can be driven in the MC-Lab by controlling it remotely from the OS-Lab. The virtual lab in Gemini can include virtual objects detected by the virtual sensors on the vessel. Control systems that use sensor data and situational awareness modules to avoid obstacles can for instance test their system on the physical boat without using too much time placing a physical obstacle or be afraid of crashing and damaging the vessel.

The quality of the visualization of the DT and the frequency of the sensor datatransfer from Gemini was found to be affected by the hardware of the machines running the system. Remote monitoring of the physical vessel was also limited to the hardware at the MC-Lab. The cameras of the QTM that capture the position of the vessel did not have a fixed

origin for the coordinate system every time it was run. Calibrating the origin of the vessel position was therefore done manually by eye, resulting in the virtual vessel position not being fully accurate. However, the calibration of the position was not a high priority, since the cameras will be exchanged with fixed cameras with a fixed origin point in the near future. The other subsystems such as the remote control are efficient in sending control commands from one lab to the other and have little to no delay when controlling the physical vessels. The virtual sensors also give promising results and are fully customizable when regarding placement and quality of the sensordata.

To summarize, the DT is a sufficiently detailed and accurate 3D representation of the MC-Lab and the cybership vessels. The vessels can be controlled using python simulators and visualized in Gemini. The RCC function allows for monitoring and remote control of the physical vessel at the MC-Lab from the OS-Lab. Thus, the system can perform the main objectives of this thesis.

7.1 Recommended further work

Designing a complete DT is not a small feat. This thesis is the initial attempt of creating such a system for the MC-Lab. A DT can be as simple and as complex as needed. Therefore, there are a lot of improvements that can be implemented for the existing features, and many new features that can be added to this DT.

Some of the suggestions for further work are,

- Create a better interface for the DP controllers that is more visually pleasing and easier to use.
- Include a python 6DOF python simulator for the vessels, such that the roll, pitch and heave motions can also be visualized. In this case, better 3D model of the water surface with the correct wave heights and period should be implemented.
- Calibrate the virtual sensors, such that they match the specifications of the real sensors available at the MC-Lab.
- Develop situational awareness systems or other object detection algorithms that use the virtual sensor data and compare it to physical sensors.
- Add other types of virtual sensors such as infrared (IR) camera or radar.
- When the fixed cameras have been set up at the MC-Lab, update the online monitoring system to visualize the exact position of the vessel.
- Add a feature to be able to use virtual reality (VR) glasses

- Create a more complex DT with focus on the individual thrusters, fuel consumption, and servo-motors.

Bibliography

Ntnu shore control lab. https://www.ntnu.edu/shorecontrol, 2022. Accessed: 2022-10-10.

Ntnu-mcs, 2022. URL https://github.com/NTNU-MCS.

Ocean systems lab. https://sites.google.com/view/os-lab/home, 2023. Accessed: 2022-10-13.

Njord-the-autonomous-ship-challenge, 2023. URL https://github.com/Njord-The-Autonomous-Ship-Challenge/Gemini.

Remote procedure call (rpc) in operating system. https://www.geeksforgeeks.org/remote-procedure-call-rpc-in-operating-system, 2023. Accessed: 2023-05-09.

Aaron D. Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. Control barrier functions: Theory and applications. In *2019 18th European Control Conference (ECC)*, pages 3420–3431, 2019. doi: 10.23919/ECC.2019.8796030.

Norwegian Maritime Authority. Rsv 12-2020 guidance in connection with the construction or installation of automated functionality aimed at performing unmanned or partially unmanned operations, 2020.

Jon Bjørnø. Thruster-assisted position mooring of c/s inocean cat i drillship. Master's thesis, NTNU, 2016.

C. Cabos and C. Rostock. Digital model or digital twin? , in 'compit'18 conference', pavone, 2018.

Edmund M. Clarke and Bernd-Holger Schlingloff. Chapter 24 - model checking. In Alan Robinson and Andrei Voronkov, editors, *Handbook of Automated Reasoning*, Handbook of Automated Reasoning, pages 1635–1790. North-Holland, Amsterdam, 2001. ISBN 978-0-444-50813-3. doi: https://doi.org/10.1016/B978-044450813-3/50026-6. URL https://www.sciencedirect.com/science/article/pii/B9780444508133500266.

R. T. H. Collis. Lidar. *Appl. Opt.*, 9(8):1782–1788, Aug 1970. doi: 10.1364/AO.9.001782. URL https://opg.optica.org/ao/abstract.cfm?URI=ao-9-8-1782.

DNV. Remote-controlled and autonomous ships. 2018.

DNV. Qualification and assurance of digital twins. 2021a.

DNV. Fleet-wide implementation of next-generation hull integrity monitoring, 2021b. URL https://www.dnv.com/expert-story/maritime-impact/Fleet-wide-implementation-of-next-generation-hull-integrity-monitoring.html.

L. Dubins. On curves of minimal length with a constraint on average curvature and with prescribed initial and terminal positions and tangents. *American Journal of Mathematics 79*, pages 976–516, 1957.

Thor I. Fossen. *Marine Craft Hydrodynamics and Motion Control*. Wiley, 2021.

Michael Grieves and John Vickers. *Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems*, pages 85–113. 08 2017. ISBN 978-3-319-38754-3. doi: 10.1007/978-3-319-38756-7_4.

John K Haas. A history of the unity game engine. *Diss. WORCESTER POLYTECHNIC INSTITUTE*, 483:484, 2014.

Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2 edition, 2004. doi: 10.1017/CBO9780511811685.

HeidiVision. Pinole camera model. https://hedivision.github.io/Pinhole.html#ref1, 2019. Accessed: 2022-12-02.

J. Heikkila and O. Silven. A four-step camera calibration procedure with implicit image correction. In *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 1106–1112, 1997. doi: 10.1109/CVPR.1997.609468.

Tor Kvestad Idland. Marine cybernetics vessel cs saucer:-design, construction and control. Master's thesis, NTNU, 2015.

IMO. Autonomous shipping. https://www.imo.org/en/MediaCentre/HotTopics/Pages/Autonomous-shipping.aspx, 2022. Accessed: 2022-12-03.

Palak Jain. Differences between tcp and udp, 2023. URL https://www.geeksforgeeks.org/differences-between-tcp-and-udp/.

X. Maldague. Ndt techniques: Thermographic. In K.H. Jürgen Buschow, Robert W. Cahn, Merton C. Flemings, Bernhard Ilschner, Edward J. Kramer, Subhash Mahajan, and Patrick Veyssière, editors, *Encyclopedia of Materials: Science and Technology*, pages 6036–6039. Elsevier, Oxford, 2001. ISBN 978-0-08-043152-9. doi: https://doi.org/10.1016/B0-08-043152-6/01065-2. URL https://www.sciencedirect.com/science/article/pii/B0080431526010652.

Eirik Midtun. Dp observer algorithms, tuning and testing. Master's thesis, NTNU, 2023.

mqtt.org. Mqtt: The standard for iot messaging. https://mqtt.org/, 2022. Accessed: 2022-12-12.

NTNU. Trondhjem biological station. https://www.ntnu.edu/biology/research/tbs, 2022. Accessed: 2022-10-10.

NTNU and SINTEF. Oceanlab, 2022. URL https://www.ntnu.edu/oceanlab.

James Premraj and Robert Opland. Talk about the mc-lab, Oct 2022.

Morgan Quigley. Ros: an open-source robot operating system. In *ICRA 2009*, 2009.

Morgan Quigley, Brian Gerkey, and William D. Smart. *Programming Robots with ROS: A Practical Introduction to the Robot Operating System*. O'Reilly Media, Inc., 1st edition, 2015. ISBN 1449323898.

Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *NIPS*, pages 91–99, 2015. URL http://dblp.uni-trier.de/db/conf/nips/nips2015.html#RenHGS15.

J. A. Robinson. Loveland donald w.. automated theorem proving. a logical basis. fundamental studies in computer science, vol. 6. north-holland publishing company, amsterdam, new york, and oxford, 1978, xiii + 405 pp. *Journal of Symbolic Logic*, 45(3): 629–630, 1980. doi: 10.2307/2273428.

L. Rosencrance and B. Matturro. Remote procedure call(rpc). https://www.techtarget.com/searchapparchitecture/definition/Remote-Procedure-Call-RPC, 2021. Accessed: 2023-05-09.

ros.org. Odometry message. http://docs.ros.org/en/noetic/api/nav_msgs/html/msg/Odometry.html, 2022. Accessed: 2023-04-26.

Håkon Nødset Skåtun. Development of a dp system for cs enterprise i with voith schneider thrusters. Master's thesis, NTNU, 2011.

Øyvind Smogeli. Introduction to autonomous marine systems. 2022a.

Øyvind Smogeli. Lecture notes in tmr06 simulation-based testing and verification. 2022b.

Mathias Netland Solheim. Integration between lidar- and camera-based situational awareness and control barrier functions for an autonomous surface vehicle. Master's thesis, NTNU, 2022.

Asgeir J. Sørensen. A survey of dynamic positioning control systems. *Annual Reviews in Control*, 35(1):123–136, 2011. ISSN 1367-5788. doi: https://doi.org/10.

1016/j.arcontrol.2011.03.008. URL https://www.sciencedirect.com/science/article/pii/S1367578811000095.

UnityTechnologies. Unity manual, 2015. URL https://docs.unity3d.com/510/Documentation/Manual/.

Kjetil Vasstein, Edmund Brekke, Rudolf Mester, and E Eide. Autoferry gemini: a real-time simulation platform for electromagnetic radiation sensors on autonomous ships. *IOP Conference Series: Materials Science and Engineering*, 929:012032, 11 2020. doi: 10.1088/1757-899X/929/1/012032.

Zeabuz. Ntnu and the milliampere ferries. https://www.zeabuz.com/milliampere, 2022. Accessed: 2022-12-04.

Appendix A

ROS manual

This manual is intended to provide the necessary information to download and run the digital twin of the MC-Lab. The goal of the digital twin is to be easily used by future students at NTNU. Therefor the objective is to create an overview of the startup procedure in order to save time by providing better documentation.

A.1 Downloading and network setup

A.1.1 ROS

Set up a virtual machine with Ubuntu 20.04 or any other Ubuntu version that supports ROS noetic. Download the two ROS workspaces, "catkin_ws" and "ws_saucer", that are attached to this thesis. After downloading the worskpaces,

1. Open
   ~/catkin_ws/src/ros_adapter/requirements.txt and change the content to:

   protobuf==3.17.0
   grpcio==1.41.1
   grpcio-tools==1.41.1
   PyYAML==5.4.1
   rospkg==1.4.0
   numpy
   opencv-python==4.2.0.32
   ipdb==0.13.11

2. Run in terminal:

pip install -r ros_adapter/requirements.txtinstall -r ros_adapter/requirements.txt

A.1.2 Gemini

Downloading gemini involves downloading the Gemini file and Unity Hub.

1. Install gRPC dependencies by right clicking on the "setup.ps1" file and "Run with powershell"
2. Download Unity Hub and choose version "2019.4.40f1"
3. Select the "Projects" tab and add the "Gemini-Unity" folder
4. Import the assets packages by clicking "Assets ->Import package"

After downloading and importing the neccessary files, you need to setup the network in order for ROS and Gemini to be able to communicate. On the windows computer:

1. Make sure the Position controller scripts have set the Host IP to the Windows PCs IP address which is running Unity
2. Go to **Gemini\Gemini-Unity\Assets\Gemini\Scripts\EMRSensors\Core\Sensor.cs** and change the IP address on line 18 to the IP address of the Linux/Ubuntu machine

On the Linux machine:

1. Go to **catkin_ws/src/ros_clients/config/config.yamlcatkin_ws/src/ros_clients /config/config.yaml** and change the IP address to the Windows PCs IP address
2. Go to **catkin_ws/src/ros_adapter/config/config.yaml** and change the IP address to the IP address of the Linux/Ubuntu machine

To get a more detailed understanding of the system, look at Njord's guide on https://njord.gitbook.io/njord/.

A.2 Run cases

A.2.1 Running CSS simulator in ROS and visualizing in Gemini

1. **Ubuntu/Linux** Open new terminal
   ```
   $cd catkin_ws
   $source devel/setup.bash
   $roslaunch ros_scenario launch_position_control.launch
   ```

2. **Ubuntu/Linux** Open new terminal
   ```
   $cd ws_saucer
   $source devel/setup.bash
   $roslaunch observer CBFobstacle.launch
   ```

3. **Ubuntu/Linux** Open new terminal
   ```
   $cd catkin_ws
   $source devel/setup.bash
   $rosrun ros_clients CSSpublisher.py
   ```

4. **Windows** Press play button in Gemini

A.2.2 Visualizing the physical vessel at the MClab in Gemini in real-time

1. **Ubuntu/Linux** Open new terminal
   ```
   $cd catkin_ws
   $source devel/setup.bash
   $roslaunch ros_scenario launch_position_control.launch
   ```

2. **Ubuntu/Linux** Open new terminal
   ```
   $python3 qtmdatascript.py
   ```

3. **Ubuntu/Linux** Open new terminal
   ```
   $cd catkin_ws
   $source devel/setup.bash
   $rosrun ros_clients qualisys_publisher.py
   ```

4. **Windows** Press play in Gemini

A.2.3 Visualizing sensordata in ROS

1. **Windows** Start any scenario where the ship is being visualized in Gemini in Unity and enable the "Nav_client" script

2. **Ubuntu/Linux** Run rviz

   (a) change "Fixed Frame" setting from map to "vessel_center"

   (b) Press "add", then "by topic" and choose the desired camera, for example "/Front/Image"

   (c) Press "add", then "by topic" and choose /lidar/ PointCloud 2"

A.2.4 Remote control of the physical vessel at the MC-Lab from the OS-Lab

1. Run your workspace for the vessel in the MC-Lab
2. **MC-Lab** Run "joystickSubscriber.py". Remember to change the IP address to your IP address on the Ubuntu pc and
3. **OS lab** Open new terminal
   $cd ws_saucer
   $source devel/setup.bash
   $rosrun ros_clients joystickPublisher.py

   Rember to change the IP address in the python file to the IP address of the ubuntu pc at the MClab

A.2.5 Simulating obstacle avoidance with a reference vessel

1. **Ubuntu/Linux** Open new terminal $cd catkin_ws
   $source devel/setup.bash
   $roslaunch ros_scenario launch_targetship.launch
2. **Ubuntu/Linux** Open new terminal
   $cd ws_saucer
   $source devel/setup.bash
   $roslaunch observer CBFobstacle.launch
3. **Ubuntu/Linux** Open new terminal
   $cd ws_saucer
   $source devel/setup.bash
   $rosrun rqt_gui rqt_gui -s reconfigure

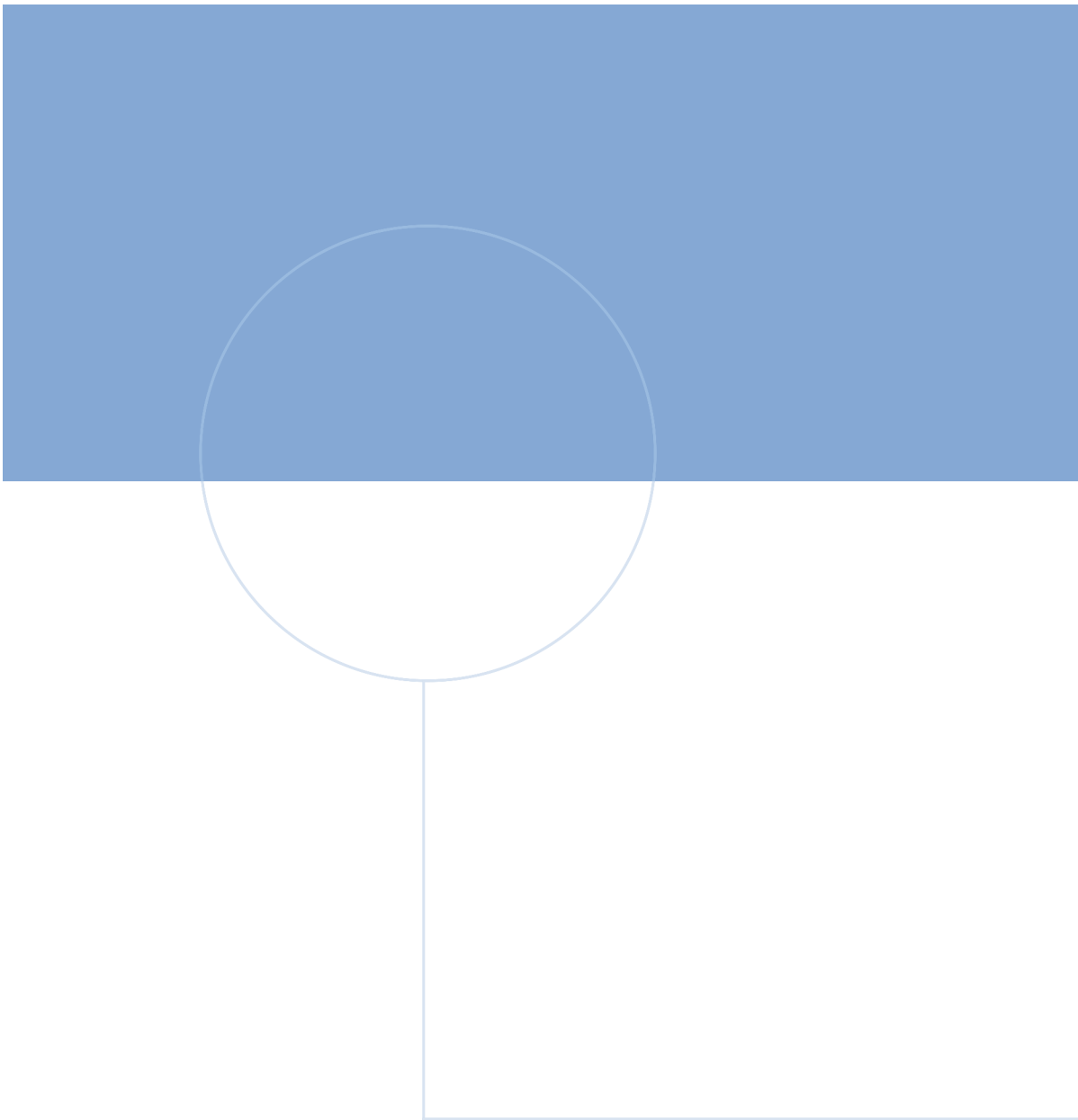
   Choose your desired U_ref and l_p
4. Press square on your controller to enable the path generation and then triangle to switch from manual mode to automatic mode
5. The visualization could be done in Gemini or by using PlotJuggler
   If gemini is chosen, remember to enable the targetposition script for the reference vessel and run the CSSTargetpublisher in ROS

Appendix B

Content in attached ZIP file

The attached ZIP-file contains,

- ROS workspace for linux machine
- Unity workspace for windows machine
- Videos of the results, if the QR code didn't work
- Poster for the master thesis



NTNU

Norwegian University of
Science and Technology