

Development and Verification of an Autonomous Personal Watercraft Testbed

A small marine craft testbed for making future research and testing within autonomous navigation in a coastal environment easily accessible

Master's thesis in Systems, Control and Mechatronics

NOEL DANIELSSON
VIKTOR LINDSTRÖM

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2021
www.chalmers.se

MASTER'S THESIS 2021

Development and Verification of an Autonomous Personal Watercraft Testbed

A small marine craft testbed for making future research and testing
within autonomous navigation in a coastal environment easily
accessible

NOEL DANIELSSON
VIKTOR LINDSTRÖM



CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Systems and Control
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2021

Development and Verification of an Autonomous Personal Watercraft Testbed
A small marine craft testbed for making future research and testing within autonomous navigation in a coastal environment easily accessible
NOEL DANIELSSON
VIKTOR LINDSTRÖM

© NOEL DANIELSSON, VIKTOR LINDSTRÖM, 2021.

Supervisor: Torsten Wik, Department of Electrical Engineering
Examiner: Petter Falkman, Department of Electrical Engineering

Master's Thesis 2021
Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Showing the scope of the project for developing and using the Marine Autonomous Research Vehicle (MARV) testbed for waypoint following.

Typeset in L^AT_EX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2021

Development and Verification of an Autonomous Personal Watercraft Testbed
A small marine craft testbed for making future research and testing within autonomous navigation in a coastal environment easily accessible

NOEL DANIELSSON
VIKTOR LINDSTRÖM

Department of Electrical Engineering
Chalmers University of Technology

Abstract

This work has focused on the design and implementation of an unmanned surface vehicle testbed platform called MARV. The purpose of this testbed is to make it easier for both researchers and students to try out new navigation algorithms for controlling a smaller vessel in a coastal environment. It has been developed in collaboration with the Swedish Sea Rescue Society and is supposed to support further research into how unmanned platforms can support their operations.

Both the hardware and software has been designed to allow for high level guidance and control tasks to be tested on the platform. To verify functionality of the testbed a demonstration implementation of waypoint following has been done. Successive waypoint following has been achieved through two independent regulators for surge and heading, receiving reference trajectories from a waypoint guidance system.

The waypoint following has been successfully tested both in simulation against a model developed for the craft and also running directly on the hardware during sea trials. This has also demonstrated the usability of the testbed for further research focused on more advanced applications.

A large focus has been to create a modular and extendable system. This has been done to not limit future research and allow for additions such as additional sensors and more compute power to be added to the system as required.

Keywords: marv, marine autonomous research vehicle, rescuerunner, ssrs, swedish sea rescue society, usv, unmaned surface vehicle, drive by wire, waypoint following, underactuated control, testbed

Acknowledgements

We would like to thank the staff at Swedish Sea Rescue Society for providing us with everything that we have needed from storing the personal watercraft to performing test drives of the system. We especially appreciate the help that we have gotten from Fredrik Falkman who has been our contact and made the cooperation between SSRS and Chalmers a possibility. We would also like to thank Torsten Wik for his time and interesting ideas during the project as our supervisor. At last we would also like to express our gratitude to our examiner Petter Falkman for giving us this opportunity to develop this platform from the ground up. With the hope that it will accelerate the research and learning within the area and maybe one day also aid SSRS in their mission.

Noel Danielsson and Viktor Lindström, Gothenburg, August 2021

Contents

List of Figures	xiii
List of Tables	xv
Acronyms	xvii
1 Introduction	1
1.1 Background	1
1.2 Purpose	2
1.2.1 Scope	2
2 MARV System Description	3
2.1 Purpose	3
2.2 Design Goals	4
2.3 System Architecture	4
2.3.1 Yamaha WaveRunner	4
2.3.2 Power Management	6
2.3.3 Drive By Wire System	6
2.3.3.1 Databus	6
2.3.3.2 Power Distribution Unit	7
2.3.3.3 Nozzle Control Unit	7
2.3.3.4 Throttle Control Unit	7
2.3.3.5 Utility Control Unit	7
2.3.3.6 Radio Control Unit	7
2.3.3.7 Operator Control Unit	7
2.3.3.8 Autonomous Control Unit	8
2.3.4 Navigation Sensors	9
2.3.4.1 SBG Inertial Navigation System	9
2.3.5 ROS 2 Nodes	10
2.3.5.1 ROS2 Bags	11
2.3.5.2 Power Management	11
2.3.5.3 Logger	11
2.3.5.4 Heartbeat	12
2.3.5.5 SBG ROS2 Driver	12
2.3.5.6 SBG Interface	12
2.3.5.7 Status Sender	12

2.3.5.8	CAN Bridge	12
2.3.5.9	Scenario Handler	12
2.3.5.10	Scenario	13
2.3.6	Communication Examples	14
2.4	Safety	14
2.4.1	Heartbeat and Unit State	15
2.4.2	12V Auto	15
2.4.3	The Dead Man’s Grip	16
2.5	Physical System Placement and Wiring	16
2.5.1	Front Mounting System	16
2.5.2	Combined Control Unit	17
2.5.3	Wiring	17
2.6	Modularity and Future Possibilities	18
2.7	Discussion	18
2.7.1	User Friendliness and Stability	18
2.7.2	System Safety	19
2.7.3	System Expansion and Maintenance	19
2.8	Future Work	20
3	Dynamics Modelling	21
3.1	Reference Frames and Notation	21
3.2	WaveRunner Physical Parameters	22
3.3	Manoeuvring model	23
3.4	Strip Theory	23
3.5	Surge Dynamics	24
3.6	Thruster Dynamics	26
3.7	Turning Dynamics	29
3.8	Discussion	31
4	Waypoint Following	33
4.1	Controller Design	33
4.1.1	Surge Controller	33
4.1.2	Heading Controller	35
4.2	Guidance Algorithm	38
4.2.1	Derivation of All Variables	39
4.2.2	Velocity Adaptation	39
4.2.3	Algorithm	41
4.2.4	Planning	42
4.3	Results	43
4.4	Discussion	47
5	Conclusion	49
5.0.1	Evaluation of Q1	49
5.0.2	Evaluation of Q2	50
5.0.3	Final Words	50
	Bibliography	51

A Appendix 1	I
A.1 Full waypoint pseudo algorithm	II

List of Figures

2.1	The MARV testbed system overview, showing how every part of the system is connected.	5
2.2	The Yamaha WaveRunner personal watercraft, used as the platform for the MARV testbed.	5
2.3	The OCU. Showing the operator interface that is located on the original glove box compartment directly in front of the operator.	8
2.4	The ACU. Based on the REACH platform this unit is the on board embedded Linux computer.	9
2.5	The collection of ROS 2 nodes, dashed circles, running on the ACU to provide higher level functions. The connecting arrows indicates the main streams of topics.	10
2.6	The Scenario Handler state behaviour.	13
2.7	How a message propagates through from a Scenario to the Steering.	14
2.8	How a message propagates through from the RCU to the Steering.	14
2.9	A rough placement of the physical enclosures of the system.	16
2.10	The FMS before and after mounting in the WR.	17
2.11	The GNSS Base Station Unit. Built for being placed on the leader/rescue boat, serving as a moving GPS base station.	20
2.12	The application idea of the GNSS Base Station.	20
3.1	Body frame coordinate system	21
3.2	CG based on estimated components and modelled hull shell [5].	22
3.3	Coast down testing.	25
3.4	The resulting coast-down points and regression polynomials.	26
3.5	Coast down data compared with simulation	26
3.6	The simulated system response using the logged coast down data as input.	28
3.7	The nonlinear and linear thruster force model, assuming a linear relationship between throttle signal and steady state velocity.	28
3.8	The travelled path (position data) data used for tuning the turning dynamics.	29
3.9	A comparison between logging data and simulation output after tuning both the sway and yaw angular velocity resistance polynomials.	30
4.1	Surge controller block diagram	34
4.2	Heading controller block diagram	36

4.3	In illustration showing the definition of all variables and relations used in the waypoint algorithm. Refer to Section 3.1 for definition of the body frame $\{b\}$ and world frame $\{n\}$	38
4.4	Showing the WaypointPlanner script's map, including the different components presented in the guidance algorithm.	42
4.5	The waypoint navigation results, for both simulation and practical test drive.	43
4.6	The waypoint navigation results, showing only the simulation and test drive results.	44
4.7	The waypoint navigation results, zoomed in.	44
4.8	A comparison between the waypoint reference velocity, guidance algorithm generated reference velocity and resulting output for both simulation and test drive.	45
4.9	A comparison between the guidance algorithm generated heading reference and resulting heading for both simulation and test drive. . . .	46
4.10	A comparison with the guidance algorithm scenario implementation bug, with and without.	46

List of Tables

3.1	Notation for marine vessels [18]	21
3.2	Mass of WR components. Positions relative to CO [5].	22
3.3	Pose of thruster and reverse gate relative to centre of mass [5]	22
3.4	List of strip theory parameters	24
3.5	List of surge resistance parameters	25
3.6	List of parameters for the thruster dynamics	27
3.7	List of turning dynamics tuning parameters	30
4.1	List of surge controller parameters	35
4.2	List of heading controller parameters	37

Acronyms

ACU	Autonomous Control Unit
CAN	controller area network
CCU	Combined Control Unit
CG	centre of gravity
CO	centre of origin
DB	Databus
DbW	drive by wire
DoF	degrees of freedom
ECM	Engine Control Module
EKF	extended kalman filter
FMS	Front Mounting System
GPS	global positioning system
ICU	Interface Control Unit
IMU	inertial measurement unit
INS	inertial navigation system
LQR	linear quadratic regulator
MARV	Marine Autonomous Research Vehicle
MCU	MARV Communication Unit
NCU	Nozzle Control Unit
NED	north east down
OCU	Operator Control Unit
PDU	Power Distribution Unit
PIU	Programming Interface Unit
RCU	Radio Control Unit

REACH Rugged Extensible Autonomous Control Hardware

ROS Robot Operating System

ROS 2 Robot Operating System 2

RTK real time kinematic

SSRS Swedish Sea Rescue Society

TCU Throttle Control Unit

UCU Utility Control Unit

USV unmanned surface vehicle

WR WaveRunner

1

Introduction

In cooperation with the Swedish Sea Rescue Society (SSRS) several bachelor and master theses have been arranged with the goal of creating an autonomous personal watercraft to support their sea rescue missions. The focus of this thesis has been to develop and evaluate a testbed named Marine Autonomous Research Vehicle (MARV) aimed at supporting further research in autonomous surface vehicles. Previous projects have been limited by not having a ready to go platform to test their work within guidance, navigation and control. The aspiration is that having access to a testbed with functioning hardware and software systems will accelerate future developments.

1.1 Background

Personal watercrafts are useful for rescue operations as they are highly manoeuvrable and have no exposed propellers that can injure people in the water. The main motivation for the autonomy goal is that operating the vessel during rough sea conditions is physically strenuous on the operator. This leads to rescuers arriving at the scene tired from the drive there. The solution of carrying the watercraft on board a lead boat has been evaluated but delays the arrival time to the destination.

Instead a concept has been proposed that allows the watercraft to autonomously follow a lead boat with an operator taking over manual control at the destination. An USV testbed would also open up possibilities for exploring other scenarios. One such idea is using autonomous craft for unmanned assistance in non critical situations such as boats becoming stranded.

This overall project has been under development for several years which means that there is a substantial amount of previous results and lessons learned. The architecture and much of the hardware design of the testbed builds upon the work done in [1] and [2]. For dynamics modelling of personal watercraft, guidance and control the work done in [3] has provided a solid reference. More recently the aspect of wave disturbances and path planning to minimise the impact of waves has been covered in [4]. Physical parameters for the WaveRunner (WR) craft have been measured in [5] but have not been verified against real world test data.

A common aspect of the previous work is that only a short amount of time has been used for physical testing at sea. Often the majority of resources have been needed for developing and implementing the system and in some cases work has been limited

entirely to simulation.

Looking at the broader research done in the field of unmanned surface vehicle (USV) [6] provides a good review of different course keeping control methods. The review also shows that physical testing was uncommon with only six out of the twenty six papers covered by the review conducting full scale trials. In [7] a good practical approach is given to model and develop a controller for a personal watercraft similar to the WR used for the MARV testbed. It covers aspect such as correlating output power with throttle command which is usually not a consideration in more theoretical work. A description and discussion of challenges related to successive waypoint tracking is made in [8], with the under actuated control problem being of particular interest given the thruster configuration of our test platform.

1.2 Purpose

The purpose of this project is to simplify the implementation and testing of future navigation algorithms of small marine vessels. More specifically research focusing on the area of search and rescue using an USV. The practical realisation of this is through the following two research questions:

- **Q1:** What aspects are important when developing a testbed for further research? That is, what aspects of the MARV system design are important to consider to enable future work?
- **Q2:** What control and guidance methods are useful for practical successive waypoint control during real world conditions?

The control and waypoint guidance is developed as a way to verify that the testbed is working as intended and as an example implementation that can be referenced.

1.2.1 Scope

The thesis does not include the basic design and construction of the hardware required for control of the craft. It does however include system integration, testing, logging implementation, ROS2 and scenario architecture.

The system should be as safe as possible and operate fully up to 10 – 15 m/s. The safety and function of the system above this limit is not taken into consideration.

The modelling and verification is done only in order to verify the system function. It is also done only for positive surge velocity in order to avoid the behaviour when the water jet is redirected by the bucket. The model is of maneuvering type which excludes the behaviour that arise when traveling at low or zero velocity. The modelling and control synthesis assumes calm waters and will such not take disturbances into account explicitly.

2

MARV System Description

The Marine Autonomous Research Vehicle (MARV) system, previously called the Autonomous WaveRunner, is the added system which is installed in the Yamaha WaveRunner (WR) platform. It consists of both hardware and software that makes it possible to externally control the vessel via manual remote operation or autonomously via the on-board computer. Together with the WR platform the system makes up the MARV testbed.

This chapter aims at providing basic knowledge of the MARV system in order to get started. It will provide an explanation of the purpose of the system, design requirements and chosen system architecture.

For a deeper understanding and technical documentation the interested reader should also take a look at the Github repositories [9]–[12], and the bachelor projects [1], [2]. The Github repositories contain the current circuit board design, manufacturing files and software. While the bachelor project report describes the work that has been previously done. It goes through the identification of problems and challenges combined with design and construction of the first version of the control system, which is the foundation for the MARV control system.

2.1 Purpose

The MARV system is supposed to make it easier for both researchers and students to try out new navigation algorithms for controlling a vessel in a marine environment. This should be done by simplifying future expansion, maintenance and the interaction with the system as much as possible. The user should not need any deep understanding of how the control commands are carried out and therefore be able to focus more on what control commands to send.

It should also provide a safer way of using a personal watercraft as a test platform. These vessels can achieve high speeds and are highly maneuverable. This is preferable since the MARV can be used as an agile testing platform. Although the versatility also increases the overall risk, in the worst case a malfunction can cause accidents within seconds. Therefore the MARV control system should also have a high focus on safety. It should constantly monitor the system for any deviation and in that case immediately interrupt the current task. In this way the main risk is left to the control algorithm itself which can also be limited by preventing it to perform too harsh control commands.

2.2 Design Goals

The purpose can be summarized into several important design goals which have been taken into consideration when designing both the hardware and software in the MARV testbed.

- **User friendliness:** The system should be as easy as possible to use for upcoming projects. The user should not need to understand how the whole system works, but only what is relevant for the task at hand.
- **System safety:** The system should be as safe as possible to use and precautions should be taken in order to minimise risks.
- **System expansion and maintenance:** It should be easy to expand the system and add more functions to or perform maintenance of existing units.
- **Stability:** The system should be stable and continue to operate without any major issues.

A subjective evaluation of how the system has met these goals is presented in the discussion at the end of this chapter.

2.3 System Architecture

The MARV control system consists of many different parts of hardware and software that are interacting with each other. This makes it difficult to unambiguously partition the system as different functionality overlaps. For the sake of providing an overview of the primary system functionality it has been divided in the parts that can be seen in Figure 2.1. A more detailed explanation of the units along with examples how the communication is carried out will be presented in this section.

2.3.1 Yamaha WaveRunner

The Yamaha WaveRunner VX Cruise HO is the personal watercraft that is used as the testing platform. On this version the steering is mechanical while the throttle is controlled by analogue signals. The choice of platform was based on that it shares almost the same driveline as the SSRS Rescuerunner vessels. If future projects result in a mature solution for autonomous control of these vessels, this would simplify the adaptation of the system to work with the Rescuerunner. An image of the WaveRunner can be seen in Figure 2.2.

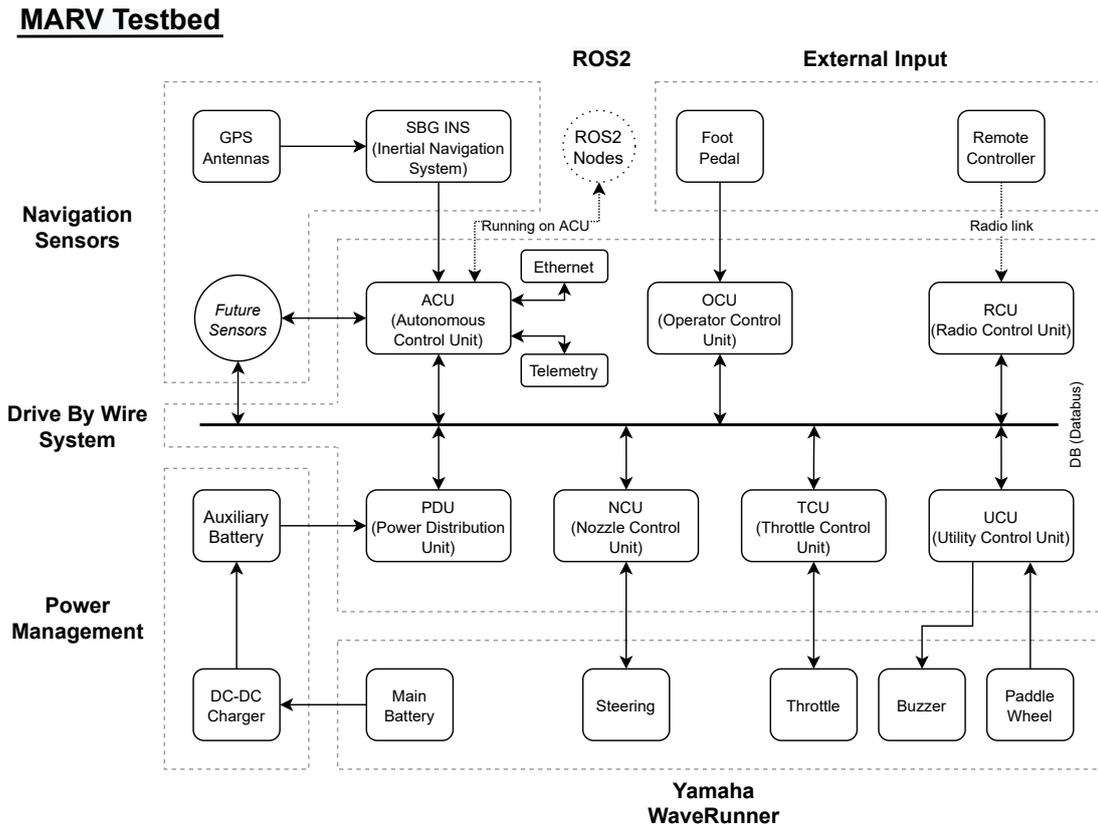


Figure 2.1: The MARV testbed system overview, showing how every part of the system is connected.



Figure 2.2: The Yamaha WaveRunner personal watercraft, used as the platform for the MARV testbed.

2.3.2 Power Management

The power system consists of a battery to battery charger and an auxiliary battery. The charger charges the auxiliary battery from the WR's main battery in a safe way. It limits the current, prevents overcharging of the auxiliary battery and cuts off the charging whenever the main battery voltage becomes too low. The auxiliary battery is used for several reasons. It prevents voltage drop when starting the WR's engine which could otherwise shut down units due to the sudden decrease in voltage level. It can also provide a high current for a short amount of time if needed. The auxiliary battery is protected from fully draining by the Power Distribution Unit (PDU). Although this is not the case if connecting external loads directly to the auxiliary battery which needs to be taken into consideration if the system is updated.

2.3.3 Drive By Wire System

The Drive by Wire (DbW) system is the main part of the MARV system. It includes all units that interact directly with the WR hardware and executes control commands. It also handles power distribution, computation and features that makes it easier for the operator to interact with the MARV system. Following is a brief descriptions of each separate part.

2.3.3.1 Databus

The system is connected together by the Databus (DB) which is a combination of both power and communication. The power distribution consist of both one 12V supply that is continuously enabled and another called 12V Auto that is controlled depending on the system mode. It is switched on or off depending on if the system is in *external* or *manual* mode.

When in external mode the system is allowed to take control and can then carry out control commands. This enables both power to stepper motor connected to the handlebar shaft and power to the relays that switch over the throttle control.

When the manual mode is activated the MARV system has no way of controlling the WR. The 12V Auto power is off which electromechanically disconnects control signals and power to actuators, disabling the systems ability to carry out control commands. Every unit can also sense the state of 12V Auto.

The communication part of the DB consists of a single CAN bus currently running at 1 Mbit/s. This enables robust and distributed communication between all the units of the DbW system.

The previous system used two separate databuses with the Interface Control Unit (ICU) in between. The ICU acted as a bridge between the two relaying incoming steering commands to the correct low level unit responsible for carrying it out. For this system the two databuses have instead been merged into one and the power management feature moved directly to the PDU. This has been done since there was no apparent advantage of having two buses, and the removal of the ICU reduces the complexity of the system.

2.3.3.2 Power Distribution Unit

The PDU provides power management for the DbW system. It contains the logic when to enable or disable external mode but can not active it by itself. If a system fault is detected external mode will immediately be disabled. Part of the power management features are under voltage protection for the auxiliary battery and electronically controlled fuses.

2.3.3.3 Nozzle Control Unit

The Nozzle Control Unit (NCU) controls the angle of the WR's waterjet nozzle. More practically it controls the stepper motor attached by a pulley system to the steering handle. It also monitors the steering angle through an absolute angle sensor. This unit can only control the system when in external mode.

2.3.3.4 Throttle Control Unit

The Throttle Control Unit (TCU) controls the throttle input to the WR's Engine Control Module (ECM). When in external mode this unit replicates the behaviour of the analogue signals normally coming from the throttle controls on the handle. When in manual mode the relays instead connects the handle throttle controls directly, which disables any signals coming out of this unit.

2.3.3.5 Utility Control Unit

The Utility Control Unit (UCU) handles extra features that are useful. It has the possibility to alert the operator if anything happens using the WR's buzzer to play different sound signals. There is also hardware that can read the paddle wheel which senses the velocity through water. It also has an accelerometer on the circuit board, for measuring acceleration up to 200g. If there are any non vital feature that needs to be added in the future, then this is the unit to upgrade.

2.3.3.6 Radio Control Unit

The Radio Control Unit (RCU) allows the MARV to be remote controlled. It decodes the incoming data from the built-in radio receiver and sends out both steering and throttle commands on the DB. This units main purpose is for testing since it is a good way of trying out the control of the system without overhead between the commands and their recipient.

2.3.3.7 Operator Control Unit

The Operator Control Unit (OCU) is a unit that has two main functions. It acts as the switch between manual and external mode. The operator can request to switch mode using the OCU. The PDU then determines if the request is valid and if so the mode is switched. It also contains an 2.7" oled display running a user interface in which the operator can get information about the system status. It also has many other functions which are to start and stop manual logging, start and stop

scenarios (implemented navigation algorithms) and also shutdown and restart the MARV system. A picture of the OCU can be seen in Figure 2.3.



Figure 2.3: The OCU. Showing the operator interface that is located on the original glove box compartment directly in front of the operator.

In order to switch to external mode the operator has to hold the foot pedal and press the blue button. If the request is accepted by the PDU then the mode will switch which is indicated by the blue button lighting up and a sound signal being played by the buzzer. Then to disable the system the operator only has to either release the foot pedal or press the red button. A confirmation sound will then be played by the buzzer and the blue light is turned off.

2.3.3.8 Autonomous Control Unit

The Autonomous Control Unit (ACU) is the on board embedded Linux computer. It is currently based on an REACH unit [13]. It is purpose built in-house to be both a robust and extensible platform for robotics and automation applications. It is based around the Nvidia Xavier NX development board that has been enclosed and passively cooled. It has four slots for expansion cards that can be used to add different features. In this case two slots are currently in use to enable the use of CAN bus and RS232 serial communication for communicating with the inertial navigation system (INS) unit. This version of the REACH unit is constructed with telemetry support which can be used for two way long range communication. It also has support for connecting up to two Raspberry Pi cameras. The REACH unit has both an Ethernet port and WiFi accessibility. A picture of the ACU can be found in Figure 2.4.



Figure 2.4: The ACU. Based on the REACH platform this unit is the on board embedded Linux computer.

On the ACU a collection of ROS 2 nodes are running. These provide the basic high level functions for collecting sensor data, handling scenarios and more. These functions are presented in detail in the ROS 2 nodes section.

2.3.4 Navigation Sensors

The navigation sensors part of the system includes any higher level sensors used for navigation. It is possible to add sensors directly to the ACU like for the INS. This can be done by creating an expansion card and installing in the REACH unit to communicate with the required protocol. It is also possible to connect a sensor by USB or install two Raspberry Pi cameras using an HDMI cable and the REACH HDMI to CSI adapter.

If the sensor supports CAN bus they could also be added directly to the DbW system, although this can affect stability and potentially the safety of the system.

2.3.4.1 SBG Inertial Navigation System

At the moment the only sensor is the SBG Ellipse2-D INS. This unit is both a GPS receiver and IMU in the same package. Thus it is able to provide both the position and orientation of the MARV. At the moment it gives a position accuracy down to 1.2 meters [14]. It has RTK support which gives an accuracy down to only a few centimeters. The INS is connected to the ACU using the RS232 serial communication made possible by an expansion card installed in the REACH unit.

Everything from the IMU data to the GPS position can be read, both raw and filtered through the Extended Kalman Filter (EKF). The filtered data is often more

reliable since it combines measurements with a marine vessel motion model in order to increase the accuracy. When running the system it is very important that the INS has mostly full status. The most important status variables is the *solution_mode*, which is the EKF filter status. This can vary from 0 to 4, where the last one means that the EKF filter should be able to output a reliable *navigation position*. On top of that the system should also show true *heading_valid*, *velocity_valid* and *position_status*. These can all be found in SBG's *SbgEkfStatus* message. They are also presented on the status page of the OCU where the *solution_mode* can be seen under *EKF*. The other variables can be seen under *INS (HVP)* where (HVP) stands for heading, velocity, position. This value will show a sum that varies from 0 to 111. If the heading is valid it will add 100, if the velocity is valid it will add 10 and for the position it will add 1. When all these are true the navigation system is fully operational. If the system does not manage to achieve full status it can help to power cycle the INS. When there is no connection with the INS the OCU will show the status as -1.

The INS needs to be configured in order to work. This can be done using the SBG Center application to set the output variables and frequency. The lever arms which are the distances between the different components used by the INS must also be set.

2.3.5 ROS 2 Nodes

On the ACU a collection of Robot Operating System (ROS) 2 nodes are running. They provide several different high level functions in the system which enables autonomous control and operator interaction to the MARV. The different nodes can be seen as dashed circles in Figure 2.5.

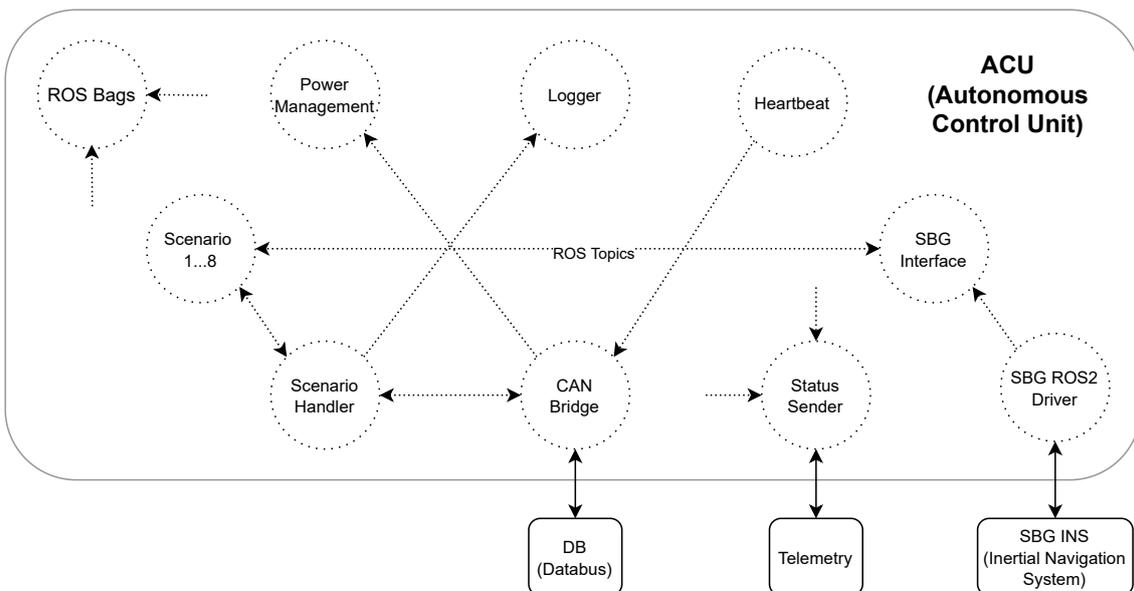


Figure 2.5: The collection of ROS 2 nodes, dashed circles, running on the ACU to provide higher level functions. The connecting arrows indicates the main streams of topics.

The nodes communicate with each other using topics (messages). The topics related to the MARV system are divided into four main categories:

1. *marv/nav/...* contains everything going out from the SBG Interface node. These contain everything that has to do with navigation which are for example the pose and the extended Kalman filter status.
2. *marv/status/...* which contains status messages from the system. It can be log data from the DbW system or other status messages coming from any ROS 2 node.
3. *marv/ctrl/...* which are the ones that can control the system in any way. These are for example the throttle, steering and sound signal commands.
4. Other topics that originate from sensor drivers or other subsystems that are not directly part of the MARV topic structure. These are for example the messages coming from the SBG ROS 2 driver node. They contain the raw topics coming directly from the INS.

2.3.5.1 ROS2 Bags

The ROS 2 bags logs the specified ROS topics which are messages sent in between the nodes. These bags can either listen to specific topics or all available, hence the unconnected arrows in Figure 2.5. This is the main way of logging what is happening in the system. Everything from the steering angle, battery voltage, throttle command to position and orientation are sent as ROS topics that can be logged. The data can then be extracted from the bags and used in for example MATLAB. On Github there is a Jupyter notebook [15] with example scripts for extracting data from the ROS bags. Any number of ROS bags can be started listening to different combinations of topics.

2.3.5.2 Power Management

The Power Management node handles shutdown and reboot functions for the ACU. It enables the use of the power button to request a shutdown but it can also be held down to force the power off. It also gives the possibility to cleanly shut down or reboot the ACU by receiving commands from the OCU. This is important since improper shutdown can corrupt data on disk. The ROS bags are also sensitive to incorrect shutdown of the system.

2.3.5.3 Logger

The Logger keeps track of which logging interval that is currently active. The system is always logging data since that is how the ROS bags work. For the sake of simplifying the manual or automatic logging this node send a message with the current interval that increases from zero by one every time the logging is started or stopped. When not running it is simply sending -1 to indicate that it is not activated. Note that the log marker count does not have persistence between node restarts.

2.3.5.4 Heartbeat

This node sends the current status of the ACU out on the DB. The status is displayed on the OCU and can be changed for the ACU by publishing the state topic that the node listens to.

2.3.5.5 SBG ROS2 Driver

The SBG ROS2 Driver node implements the communication protocol and functions for communicating with the INS. This node is provided by SBG Systems which is the manufacturer. In the configuration file the output and frequency of different data topics can be set. These needs to match the settings in SBG Center configuration, as they will only affect the ROS 2 side and not the INS itself.

2.3.5.6 SBG Interface

The SBG Interface node is a middle layer between the SBG ROS2 driver and the rest of the nodes. It has several purposes. The first one is to simplify the output, picking out the most wanted data and republishes it as new topics to the system using standard ROS 2 topics instead of custom messages. It also transforms the position data from geographical coordinates to local navigation coordinates by using a reference position that needs to be set. Additionally it also sends out information to the OCU about the INS status.

2.3.5.7 Status Sender

The Status Sender node is a temporary node that sends some debug information, such as the position, status of INS and more, over the long range telemetry link. This data can be received by the MARV Communication Unit (MCU) which has a telemetry adapter attached to it.

2.3.5.8 CAN Bridge

The CAN Bridge node connects the ROS network to the DB. It simply maps together and forwards messages both ways. This is done differently depending on how the message is incoming. For ROS to CAN the bridge listens to the specified topics, then looks up the CAN arbitration id, packs the message and then sends it. For CAN to ROS the node checks which CAN arbitration id that the incoming message has. Then it unpacks the data, packs it into a the corresponding ROS topic that is then published. Every message that should pass in between the two must be declared in the DBC file [16], found in the MARV-ROS repository [12]. The action upon receiving or sending the message must also be implemented in the bridge.

2.3.5.9 Scenario Handler

The Scenario Handler takes care of running the user implemented navigation algorithms, also called scenarios. It keeps track of which scenarios that have been started, which one are running and also the status of the running scenario. It acts

as a middle layer of safety between the scenario and the MARV. It also limits the amount of commands that can be sent per second and it stops the scenario if it does not behave as expected. The scenario handler allows the the scenarios to be updated dynamically on the OCU, allowing the user to customise which scenarios that are active. It also allows the user to send progress data, two variables and text notifications to show in the OCU during execution.

The Scenario Handler switches state depending on what is happening in the system. A flow chart of the behaviour can be seen in Figure 2.6 where the state is shown within parenthesis.

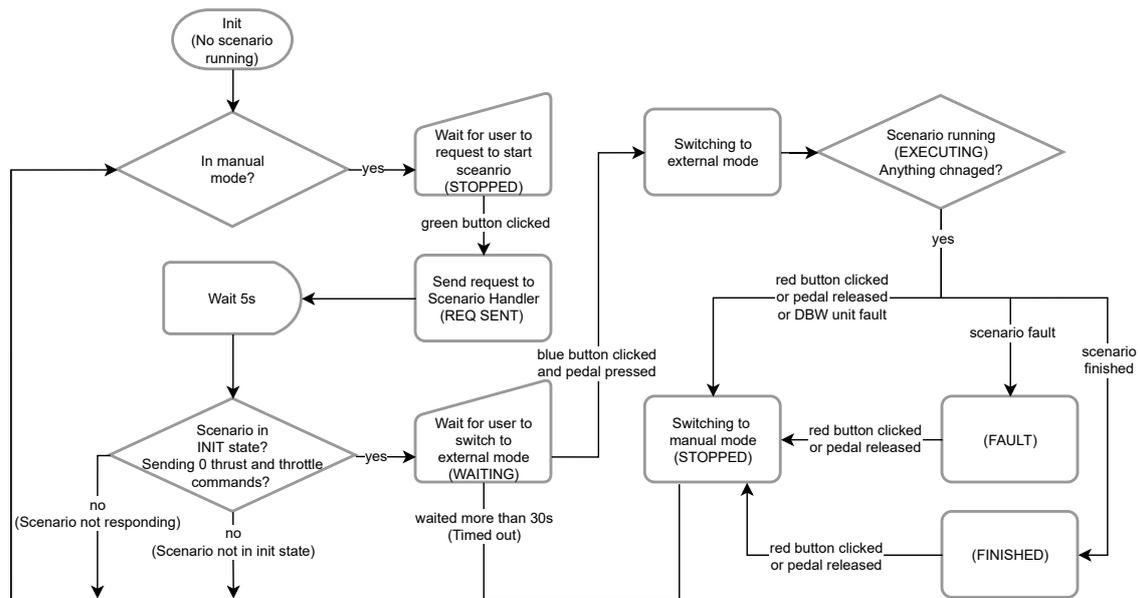


Figure 2.6: The Scenario Handler state behaviour.

2.3.5.10 Scenario

A scenario is a way of executing user written algorithms, such as waypoint following or thrust controller testing. They need to follow the basic structure of a scenario and inherit the scenario class function. This enables them to communicate with the scenario handler.

A scenario needs to behave as expected. This means that a setup can be run on startup but the node should not start by itself before getting the go ahead by the scenario handler. For a lot of examples using the correct structure take a look at the MARV-ROS repository [12]. The example structure can of course be modified and used freely as long as the following requirements are met:

- Inherit the functions from the Scenario class, these enables the features to send steering, throttle, scenario finished, update progress, data variables and more.
- Only initialize the necessary things at startup. This includes global variables, publishers and subscribers that will not trigger when the node is not active (started by the scenario handler). It also includes a timer that checks if the scenario should start.

- The Scenario state needs to be changed from stopped to init when a start is requested in order to be able to start. It should also begin to send 0 deg steering angle and 0 throttle commands, only then the PDU will allow switching to external mode.

For the documentation of using the scenario class, take a look at the MARV-ROS repository [12].

2.3.6 Communication Examples

When sending messages in the system the data will propagate through it using different methods and protocols. There is no guarantee that every message will be delivered but most of the functions do not require this. In Figure 2.7 an example of how a message propagates from a Scenario to the WR's steering.

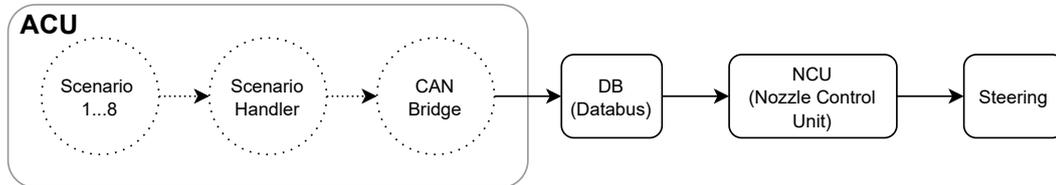


Figure 2.7: How a message propagates through from a Scenario to the Steering.

The message first begins as a ROS topic that are routed to the CAN Bridge node. There it is converted into a CAN message and then sent out on the DB. The NCU listens for the CAN messages and then executes the command to alter the steering angle.

Another example can be seen in Figure 2.8 where the RCU is the sender. In this case the message goes directly from the RCU out to the DB and then to the NCU. This is why the RCU is a great way of debugging the basic functions, since there are fewer involved subsystems.

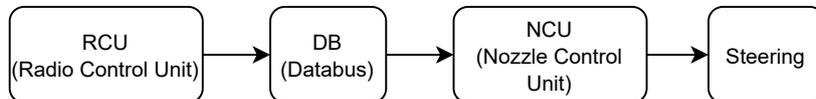


Figure 2.8: How a message propagates through from the RCU to the Steering.

2.4 Safety

It has previously been mentioned that the safety is an important aspect of the MARV control system. The WR requires both knowledge and sense to be handled in a safe way. When autonomous control is introduced this is even more crucial. Therefore the system has several layers of precautions in order to minimise the risk.

2.4.1 Heartbeat and Unit State

Every control unit in the DbW system continuously sends their heartbeat state. This is used to monitor the status of the unit. It can be either Not Connected (NC), Error (ER) or Okay (OK). Whenever a unit is connected and working properly it should be in OK state. From there on it can then switch to ER if something happens that interrupts the original functionality. This transition must be user specified and implemented in the program that is running since an error will not be detected automatically. It is also possible that the unit ends up as NC state. This will only occur if the connection to the unit is lost. The unit itself will not change its own state to NC but other units on the bus will deem the unit to be in NC state if it does not respond. Whenever a unit changes state from OK to ER or NC the user is also notified by a popup on the OCU and a sound signal.

In order to start a scenario the PDU requires the state of the TCU and NCU to be OK. It also needs the source unit of the commands to be in OK state. The source can be either the ACU or RCU but not both at the same time. If these requirements are not met then it is not possible to switch to external mode. In order to reset the system a full restart should be made.

The PDU can change to ER state for different reasons. The first is if the heartbeat is lost or changed to ER from any of the above mentioned units during execution in external mode. Except for the ACU and RCU, where it only cares about the source unit. Another reason is if the steering and throttle commands are suddenly stopped in external mode then the PDU will enter error state.

2.4.2 12V Auto

As previously mentioned the 12V Auto power feature plays an important role. It is simply a 12V power line that is controlled by the PDU. It is used for driving safety critical functions such as the steering stepper motor and the relays that switches between normal and autonomous control of the throttle signal. When this line is turned off the system cannot affect the WR in any way. If the power goes out in the MARV control system the control is immediately given to the operator. Mode switching is completely electromechanical to be as safe as possible.

The 12V Auto power is only enabled by the PDU if all requirements are met. Except for the state logic mentioned above the source must also send commands. For the RCU the Radio Controller only needs to be active by being turned on to do this. For the ACU the same basic requirements are needed. Although as mentioned under the scenario handler and scenario logic, the scenario both needs to be in init state and sending commands with 0 throttle signal and 0 steering angle.

In order to switch over to external mode and activate the 12V auto power the user must as previously described under the OCU Unit hold both the pedal and press the blue button. Then if either the pedal is released or the red button is pressed the system will directly switch over to manual mode again. In this way it is possible to switch of the system even if the user is not able to reach down to the OCU's red button.

2.4.3 The Dead Man's Grip

The system requires an operator to be on board at all times. This is both because of the fact that it is not easy to only steer the vessel by radio control, but also if the systems acts up. The WR has a normal dead man's grip. This is required in order to start and run the motor. If it is removed the motor will stop. This is the last layer of security if the operator falls off or if the MARV control system malfunctions. Despite the precautions taken above the operator needs to continuously monitor the situation as the MARV system has not gone through any extensive testing of the safety. It has however worked as expected during the initial testing and much thought has gone into making it as safe as possible.

2.5 Physical System Placement and Wiring

The physical enclosures of the system units are placed in different areas on both the inside and outside of the WR. A rough placement can be seen in the Figure 2.9.

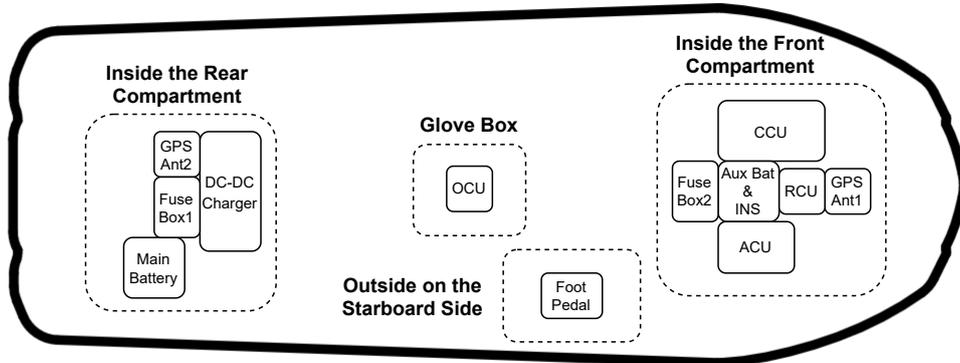
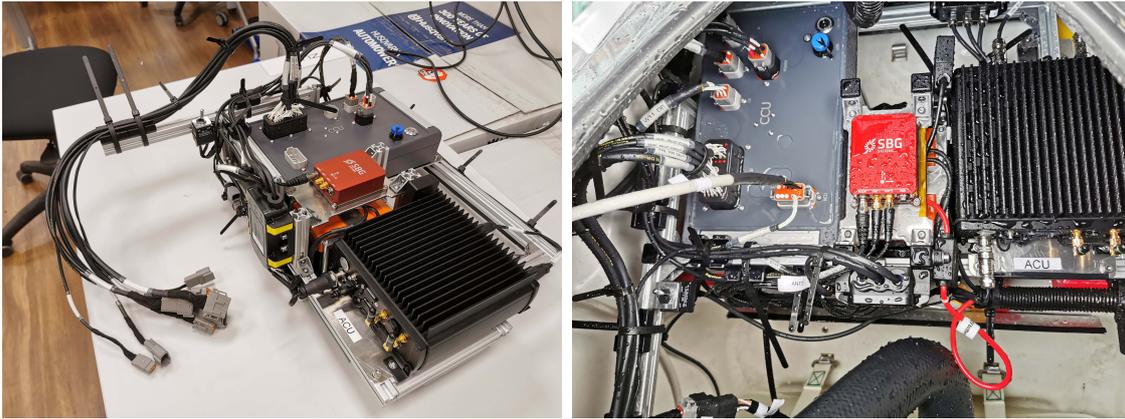


Figure 2.9: A rough placement of the physical enclosures of the system.

2.5.1 Front Mounting System

The Front Mounting System (FMS) is a modular mounting platform made up by aluminum 2020 extrusions and fixture plates. It is located inside the front compartment where it holds all the respective units seen in Figure 2.9. The whole structure is damped by studded dampers in order to reduce vibrations. In Figure 2.10 is a picture of the FMS when newly constructed and not yet installed, and then when it is installed in the WR's front compartment.



(a) The FMS after construction has been finished. (b) The FMS mounted inside the WR's front compartment.

Figure 2.10: The FMS before and after mounting in the WR.

2.5.2 Combined Control Unit

The Combined Control Unit CCU is a physical enclosure for the PDU, TCU, NCU and UCU. The CCU has been manufactured using an aluminium alloy enclosure to allow effective transportation of heat. The connector cutouts are CNC milled to allow for the use of non-circular connectors. Inside it has an aluminum fixture plate which all the circuit boards are mounted to. The whole unit is made to be as stiff as possible to minimize flexing that could otherwise damage electrical components.

Every on board unit can be selected and programmed using a single usb port. Using the same port it is also possible to get serial data for all units simultaneously. This is achieved by the Programming Interface Unit (PIU) that is also located inside the CCU.

2.5.3 Wiring

All the units are connected together using different wires and wire gauges depending on the application. Naturally the power management part therefore consist of thicker wire of 6-4 mm². For the DB a wire size of 0.75 mm² is used and recommended if the DB is expanded.

Most of the wires has been put together into cable harness assemblies, following a guide of making motor sport grade cable harnesses [17]. This has been done using mostly Deutsch connectors, shrink tubing for ingress protection, CAN bus cable and quality wires. This should increase the durability of the system since it reduces the risk of it getting damaged by the harsh environment over time.

The complete wiring diagram and specification can be found in the MARV-Hardware-Electrical repository [10].

2.6 Modularity and Future Possibilities

The MARV control system is built with a modular design in mind. It is possible to switch out, upgrade and add new units to the existing architecture, as new needs arise. This has mostly already been mentioned in this chapter but here are some key points.

- The DB currently runs from the Front Compartment, up the RCU located inside the hatch, then to the Glove Box where the OCU is located. From here it is possible to extend the bus which provides both power and communication for new units. It is also possible to extend the DB at other ends but then a new cable harness needs to be manufactured. If any new units are added to the DB, make sure that the system still functions as intended since the communication is crucial for the both safety and performance of the system.
- The units inside the CCU can be switched out if any upgrade is needed. Although they need to have the same dimensions, hole patterns and connector placement in order to fit inside the enclosure.
- It is possible to add new navigation sensors by connecting them directly to the ACU, either by a REACH expansion card or USB. The ACU is also prepared for the use of two Raspberry Pi cameras. New sensors can also be added to the DB as a new unit.
- The FMS is constructed using normal 2020 aluminium extrusions, with 5 mm grove size. It can easily be extended but make sure to use fasteners of stainless steel or aluminium since the environment is corrosive.
- If any new unit is added make sure to keep the same CAN message id structure and unit state logic.

2.7 Discussion

Presented in this section is a subjective evaluation of the MARV system design goals.

2.7.1 User Friendliness and Stability

The MARV control system has become quite complex. It includes a lot of physical units, many thousand lines of code, logic for both keeping the safety up and handling different actions. Despite this it has been quite easy to create and run several different scenarios. More than ten different scenarios has been created and tested, everything from steering tests to waypoint control. The important thing has been to keep the basic structure and abstractions. Testing out the scenario several times and trying out the behaviour on land before even taking it on to a real test drive. This is crucial as the scenarios are written in Python and some errors are only found during runtime. An improvement here would be automated integration and unit testing of the system to ensure that changes do not have corner cases that do not get discovered by running a simple test scenario.

The system itself has been operating as expected, with some deviations presented further down. This means that the user should only need the basic knowledge in order to operate it, as long as it works as expected. When problems do occur they are usually bound to coding errors in the scenarios, or intermittent system errors that are really hard find the source of. Although, these can mostly be fixed by a full restart.

The CAN Bus Problem is a problem with CAN bus inside the DB. Despite the fact that it is running with 1 Mbit/s units starts to generate error frames and then going of bus at around 300 messages per second when all units are connected. When a unit (RCU,ACU,TCU or NCU) goes of bus the PDU will detect this and switch over to manual mode, thus stopping any active scenario. This number of messages is not nearly at the limit of what it can handle but it seems to be due to the fact that every DbW unit except for the ACU and PDU runs without an external crystal. This probably causes timing issues on the bus, causing some units to report messages as corrupt. It can be possible to fix this by adding an external crystal for those units or by calibrating the internal oscillator. Currently a workaround has been implemented by keeping the number of messages to a minimum and disabling the PDU from transmitting (and thus not reporting any errors). This unfortunately removes the possibility add new units to the DB at the moment, without risking more instability. Although even if this still occurs the system can still be restarted but it will of course interrupt the current task.

2.7.2 System Safety

The system safety functions have worked as expected during testing. The complexity of the logic and single failure points have been kept to a minimum. The PDU is responsible for the decision to enable switching to external mode but it is always the operator that does the activation. The foot pedal which was introduced in this version of the system has been used during testing to quickly stop the system and also disable the system if the operator momentarily becomes unbalanced. The redundant safety design used ensured that the CAN problems discussed never affected the safety of the system.

2.7.3 System Expansion and Maintenance

Modifying or adding any unit to the DbW system is a challenge. It requires both knowledge about electronics design, embedded programming and the system architecture. This is however possible but it has not been the main focus. The most important thing is to be able to add new sensors and writing software on the ACU. This requires knowledge about the sensors, different interfaces, ROS 2 and the ROS 2 node structure, which is reasonable.

When instead considering maintenance the system has not become any easier to work with. This is mostly inherent to the system being installed in an inaccessible area. The stepper motor and pulley system takes significant time to take out and in. Each unit can however be disconnected and removed for any maintenance even if some effort is required. Each and every cable is labelled and mostly keyed to only fit

where it is supposed to. The system has also so far reached expectations regarding robustness, which should minimise the need for maintenance.

2.8 Future Work

Previous projects that have been focusing on following a lead rescue boat. For this purpose a GNSS Base Station unit has been constructed to act as a moving base. It is built upon an extended REACH unit with an Nvidia Jetson Nano inside. The unit can be seen in Figure 2.11.

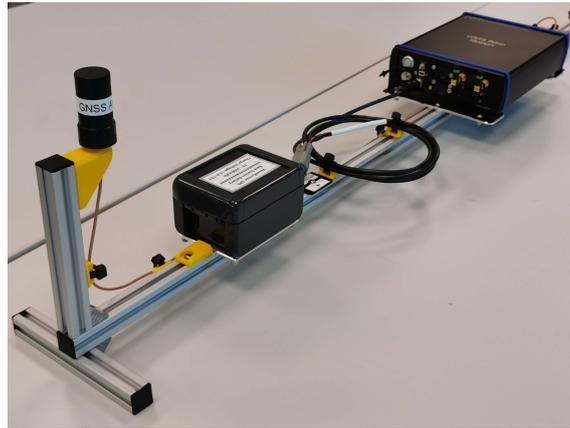


Figure 2.11: The GNSS Base Station Unit. Built for being placed on the leader/rescue boat, serving as a moving GPS base station.

It uses the Satlab UAV/RTK GPS unit with two antennas that can provide both the position and heading. This unit has only been assembled and has no software running on it. A protocol for communicating with the GPS device has to be written or implemented in order for the unit to function properly. The telemetry communication between the WR and GNSS base station also needs to be implemented. This can be done by sending data one way or by implementing a communication protocol for handling two way communication. A block diagram of the application concept can be seen in Figure 2.12.

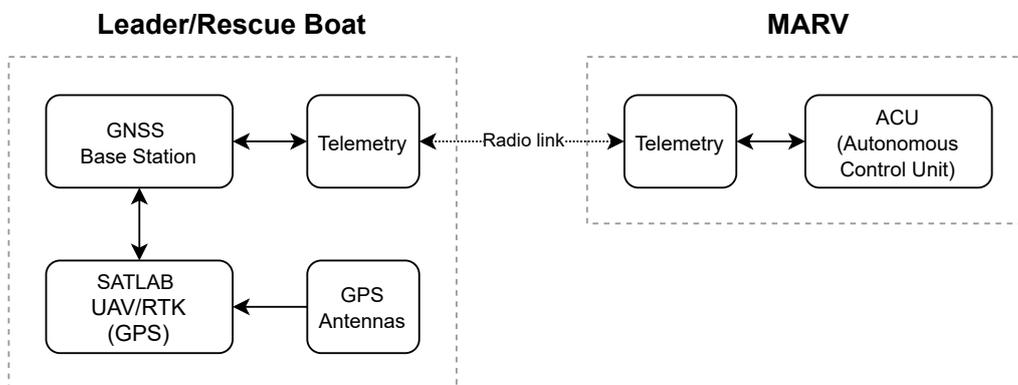


Figure 2.12: The application idea of the GNSS Base Station.

3

Dynamics Modelling

A dynamic model of the system has been developed to allow for both simulation of the system and model based control design. The focus has been on a simple model that is sufficiently accurate to be used for control design and waypoint guidance.

3.1 Reference Frames and Notation

Two reference frames are of interest for guidance, navigation and control. For control design the body frame $\{b\}$ is of main interest and the axes directions in relation to the vessel are presented in Figure 3.1. The body frame is used for all forces, moments and velocities unless otherwise noted. For navigation and guidance the north east down (NED) frame $\{n\}$ is used to describe positions and angles. The NED frame rotates with the earth but is assumed to be inertial.

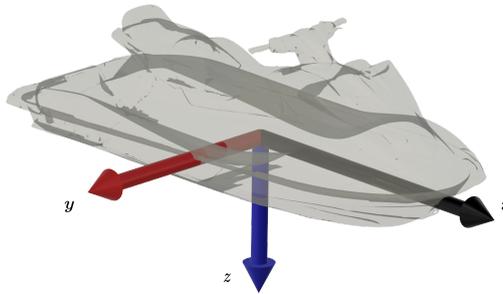


Figure 3.1: Body frame coordinate system

Table 3.1: Notation for marine vessels [18]

DoF	Forces and moments	Linear and angular velocities	Positions and Euler angles
1 motion in x (surge)	X	u	x
2 motion in y (sway)	Y	v	y
3 motion in z (heave)	Z	w	z
4 rotation around x (roll)	K	p	ϕ
5 rotation around y (pitch)	M	q	θ
6 rotation around z (yaw)	N	r	ψ

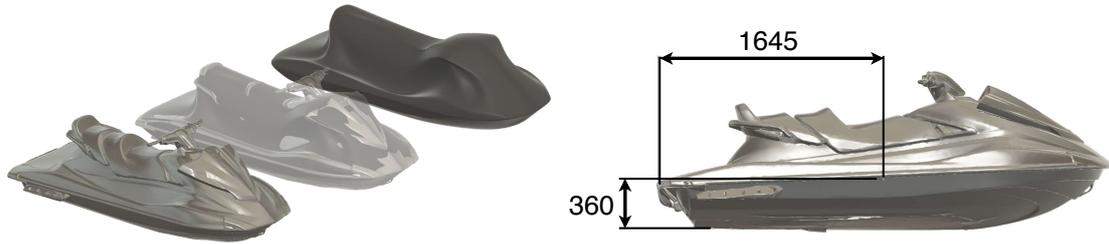
3.2 WaveRunner Physical Parameters

Here estimated and measured physical parameters of the WR will be presented. Earlier work has used methods such as 3D scanning and referencing technical documentation on the WR to accurately estimate relevant physical parameters [5].

Table 3.2: Mass of WR components. Positions relative to CO [5].

Component	Mass	Position (x, z) mm	Size (x, y, z) mm
Fuel	52.5 kg	(306,-460)	(400,550,300)
Engine	110 kg	(-494,-410)	(500,250,450)
"Hull"	238 kg	-	-
Operator	80 kg	-	-
Total	480.5	-	-

The mass of the components that contribute most to the vessel inertia has been compiled in Table 3.2. Here the hull component refers to the remaining mass after the engine and fuel has been subtracted. The distribution of the remaining hull mass has been estimated as a model thin shell around the vessel as can be seen in Figure 3.2a [5].



(a) Modelled hull based on 3d scan.

(b) Estimated CG of vessel.

Figure 3.2: CG based on estimated components and modelled hull shell [5].

Calculated inertia tensor around CG based on hull and components, excluding operator, listed in Table 3.2. The inertia tensor definition used is from [19, pp. 48-49].

$$\mathbf{I}_g = \begin{bmatrix} I_x & & \\ & I_y & \\ & & I_z \end{bmatrix} = \begin{bmatrix} 52.2 & & \\ & 238.1 & \\ & & 248.3 \end{bmatrix} \text{ kg m}^2 \quad (3.1)$$

Table 3.3 describes the position of the thruster nozzle and the reverse gates that allows the WR to redirect the water flow forward and to the sides [5].

Table 3.3: Pose of thruster and reverse gate relative to centre of mass [5]

Component	(x, y, z) mm	$(\phi, \theta, \psi)^\circ$
Main nozzle	(-1443, 0, -240)	$(180 \pm 24, 0, 0)$
Reverse gate starboard	(-1563, 140, -240)	(70, 0, 0)
Reverse gate portside	(-1563, -140, -240)	(-70, 0, 0)

3.3 Manoeuvring model

A maneuvering three DoF model was chosen for the development of a dynamical model of the craft. This limits the usefulness of the model to positive surge velocities from around 3 to 12 m/s but also significantly reduces the modeling complexities. Another simplification commonly seen in the literature is that the surge, sway and yaw dynamics are assumed to be independent from each other [20]. The CO of the body coordinates is also defined to coincide with CG of the vehicle. Water surface currents are assumed to be small in relation to the vehicle velocity. The dynamics, written using Fossen vectorial notation [19], then become

$$(\mathbf{M}_{RB} + \mathbf{M}_A)\dot{\mathbf{v}} + \mathbf{D}(\mathbf{v})\mathbf{v} = \boldsymbol{\tau} \quad (3.2)$$

with the state vector being defined as

$$\mathbf{v} = [u \quad v \quad r]^T. \quad (3.3)$$

The rigid-body system inertia matrix is based on the vehicle mass m and inertia around the z-axis I_z .

$$\mathbf{M}_{RB} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & I_z \end{bmatrix} \quad (3.4)$$

Combined with the mass contribution from the rigid body is the added mass matrix which is due to the water close to the vessel moving with it. The added mass is generally not constant during different operating conditions so additional scaling constant $k_{m\dot{v}}$ and $k_{m\dot{r}}$ have been introduced to allow for tuning of the theoretically calculated added mass that affects turning dynamics.

$$\mathbf{M}_A = - \begin{bmatrix} X_{\dot{u}} & 0 & 0 \\ 0 & Y_{\dot{v}}k_{m\dot{v}} & 0 \\ 0 & 0 & N_{\dot{r}}k_{m\dot{r}} \end{bmatrix} \quad (3.5)$$

Non linear damping matrix where each element expresses the resistance in the principal directions as a polynomial function of velocity in each direction respectively.

$$\mathbf{D}(\mathbf{v}) = - \begin{bmatrix} X_u(u) & 0 & 0 \\ 0 & Y_v(v) & 0 \\ 0 & 0 & N_r(r) \end{bmatrix} \quad (3.6)$$

External force inputs to the system are represented by the force and moment vector

$$\boldsymbol{\tau} = [X \quad Y \quad N]^T. \quad (3.7)$$

3.4 Strip Theory

To get a starting point for tuning strip theory was used to calculate a theoretical value for the added mass [21]. The added mass calculation is based on the geometry

of the submerged part of the vessel. Based on the work done by Bergholtz and Hernvall the added mass in three DoF can be parametrised [3].

$$\begin{aligned}
 -X_{\dot{u}} &= \frac{m}{20} \\
 -Y_{\dot{v}} &= \frac{L\pi\rho a^2}{2} \\
 -N_{\dot{r}} &= \frac{mb^3}{30} + \frac{L^3\pi\rho a^2}{24}
 \end{aligned} \tag{3.8}$$

The parameters presented in Table 3.4 are based on the 3d scans done in [5] and a section was done at the centre of the vessel to determine proper dimensions for the ellipse defined by a and b . The waterline was placed based on the vessel being stationary in the water with buoyancy forces carrying the entire weight of the craft.

Table 3.4: List of strip theory parameters

Parameter	Description	Value
a	Submerged depth	0.35 m
$2b$	Width at waterline	1.10 m
L	Total vessel length	3.35 m
ρ	Salt water density [22]	1025 kg/m ³

3.5 Surge Dynamics

To complete the surge dynamics component of the manoeuvring model the resistance function $X_{\dot{u}}$ needs to be defined. Looking at the complete expression

$$\dot{u}(t) = -\frac{X_u(u)}{m - X_{\dot{u}}}u(t) + \frac{X(t)}{m - X_{\dot{u}}} \tag{3.9}$$

and if there is no input force $X(t)$ the only parts that remains are

$$\dot{u}(t) = -\frac{X_u(u)}{m - X_{\dot{u}}}u(t) \iff -X_u(u)u(t) = \dot{u}(t)(m - X_{\dot{u}}). \tag{3.10}$$

This means that if $-X_u(u)u(t) = F_d$, where F_d is the drag force polynomial, it is possible to determine the dampening by a conventional coast-down test [23].

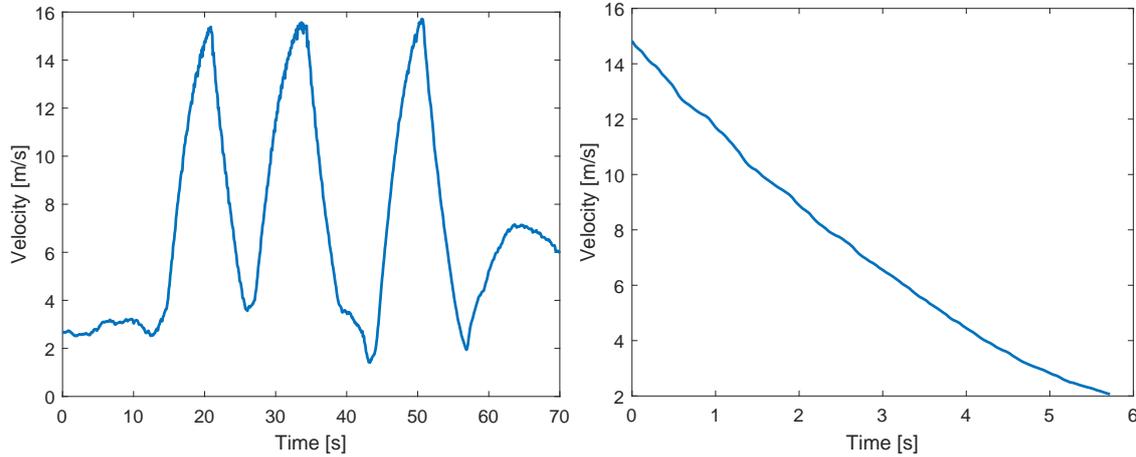
A coast-down test uses Newton's law

$$F = ma = \frac{md(v)}{d(t)} \tag{3.11}$$

in order to derive a curve of how much force is applied at a specified velocity. It is simply done by releasing the throttle at a high velocity, thus not applying any input force, and then logging the velocity and time during the reduction in velocity. Then

the only forces present are the drag/dampening force by elements such as wind and water.

The logged data can be seen in Figure 3.3, which shows three tries of first accelerating up to around 15 m/s and then releasing the throttle lever. It also shows the log data for one of the slopes of which the data can be used for the coast-down calculations.



(a) Logged data during coast-down test- (b) The log data interval that is used for coast-down calculations.

Figure 3.3: Coast down testing.

Choosing the interval with the largest decrease in velocity it is possible to perform numerical smooth robust differentiation as presented in [24] on the log data over time. The dampening force is given as a point for each time interval which could then be used to perform regression to acquire the dampening polynomials. Both a first and a third degree polynomial are computed and shown together with the points in Figure 3.4. The first degree polynomial is only derived using the interval velocity $\in [0, 10]$ m/s. This interval is specified since the model should be best in the area in which the MARV control system will operate. The first degree polynomial will later be used for controller design. The resulting polynomials are

$$F_d^{1st} = F_d^* \approx 146.4u \quad \text{and} \quad F_d^{3rd} \approx 0.972u^3 - 26.2u^2 + 285.7u. \quad (3.12)$$

Since $X_u(u) = -F_d/u(t)$ this results in the resistance functions as seen in Table 3.5.

Table 3.5: List of surge resistance parameters

Parameter	Description	Value
X_u^*	Surge resistance, linear	-146.4
$X_u^{3rd}(u)$	Surge resistance, polynomial	$-(0.972 u ^2 - 26.2 u + 285.72)$

It is then possible to use the newly calculated $X_u^{3rd}(u)$ to simulate coast-down tests. This is done picking out a coast-down logging data interval, inputting the initial velocity into the simulation and then running the test. The comparisons for two tests can be seen in Figure 3.5. The first is performed on the data used to calculate

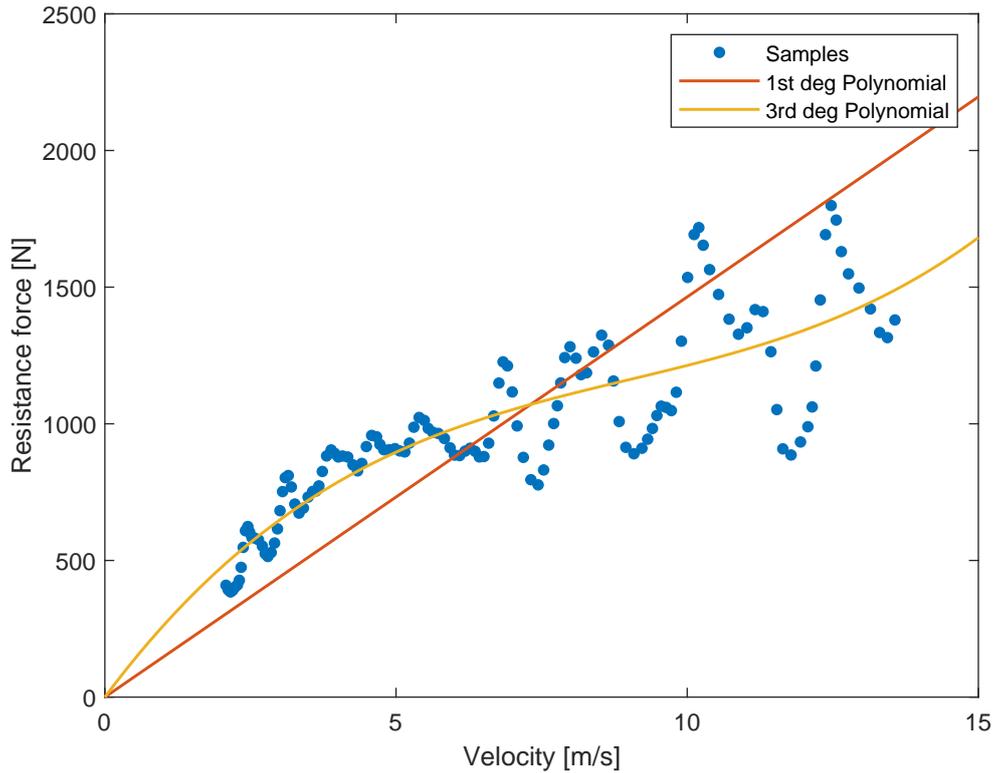
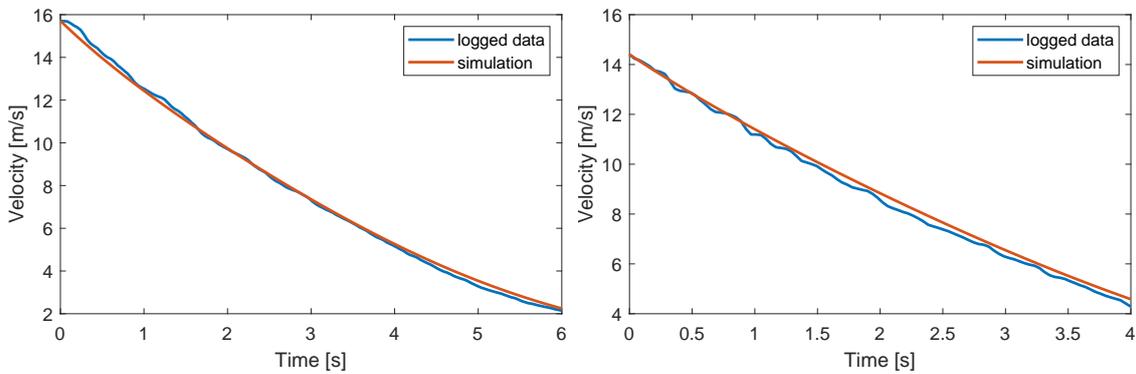


Figure 3.4: The resulting coast-down points and regression polynomials.

the polynomials and the other is fresh data. In this way the result should not be biased.



(a) Simulated coast-down result, using the third slope that is also used for calculating the polynomials. (b) Simulated coast-down result, using the second slope to avoid bias since it was not used for calculating the polynomials.

Figure 3.5: Coast down data compared with simulation

3.6 Thruster Dynamics

The thruster dynamics for the water jet is based on the jet discharge propulsion formula described in [2]

$$F_t(t) = \rho q (v_2 - v_1) k_{ft} \tag{3.13}$$

where F_t is the thruster force induced by an incompressible jet flow and k_{ft} is a scaling factor for tuning. The variable q is the volume flow of water

$$q = Av_2 = \left(\frac{d}{2}\right)^2 \pi v_2 \quad (3.14)$$

where d is the nozzle opening diameter, v_2 is the water velocity out of the jet

$$v_2 = \left(\frac{2(p_1 - p_2)}{\rho}\right)^{1/2} \quad (3.15)$$

where p_1 and p_2 is the pressure before and after the water jet respectively, and at last v_1 is the jet (and vessel) velocity.

The pressure before and after the jet varies depending on both the vessel's and water's velocity in relation to each other. This means that there are several parameters at a time that affects the outgoing force. In order to simplify the expressions the pressures based on the work in [2] are,

$$p_1 = h\rho g + u_{rel}u_{tr} \quad \text{and} \quad p_2 = h\rho g \quad (3.16)$$

where h is the water jet's submerged depth, g the gravitational constant, u_{rel} is a constant for the relation between throttle signal u_{tr} and the resulting pressure before the jet. In this way the pressure after the jet is kept at the constant pressure at h water depth, while the pressure before the jet is the one that causes the propelling force.

It is known that the Yamaha WaveRunner VX Cruise HO (High Output) model has around 180 hp [5] and an assumed max velocity at around $v = 25$ m/s. With an efficiency from the motor to the impeller of 0.8 [5] and a scaling factor of 0.75 [25] for using low speed mode (L-MODE). This would provide the maximum force F_{tm} as

$$F_{tm} = \frac{180 \cdot 736 \cdot 0.8 \cdot 0.75}{25} \approx 3180 \text{ N} \quad (736 \text{ is the conversion from } hp \text{ to } W). \quad (3.17)$$

Using the parameters seen in Table 3.6 and assuming movement at maximum velocity with the highest throttle signal ($u_{tr} = 100$), to solve the equation for the constant unknown constant provides $u_{rel} \approx 8224.92$.

Table 3.6: List of parameters for the thruster dynamics

Parameter	Description	Value
h	Water jet submerged depth	0.35 m
g	Gravitational constant	9.82 m/s ²
d	Nozzle diameter	0.08 m

For $\tau = F_t$ the manoeuvring model, surge dynamics and the thruster dynamics can then be used for simulating the system response using the logged coast down data.

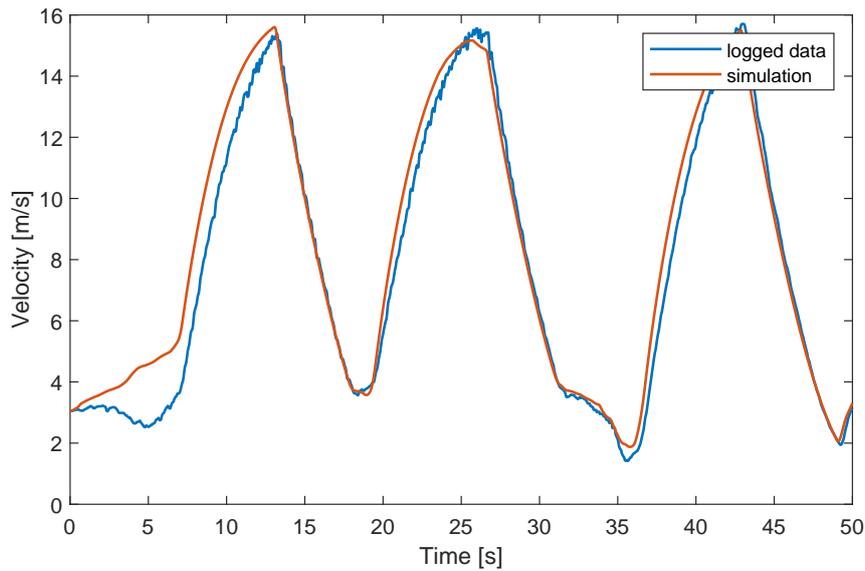


Figure 3.6: The simulated system response using the logged coast down data as input.

This gives the following results seen in Figure 3.6 when scaling down the thruster force by a tuning factor of $k_{ft} = 0.39$.

Assuming that the velocity in steady state has a linear relationship to the throttle signal it is then possible to derive a linear thruster force model. The linear polynomial is derived using linear regression in the throttle signal interval $u_{tr} \in [0, 60]\%$. This interval is chosen since it is most likely that the MARV control system will be limited to only use a maximum of 60 – 70%, for safety reasons. The results together with the nonlinear thrust model are shown in Figure 3.7.

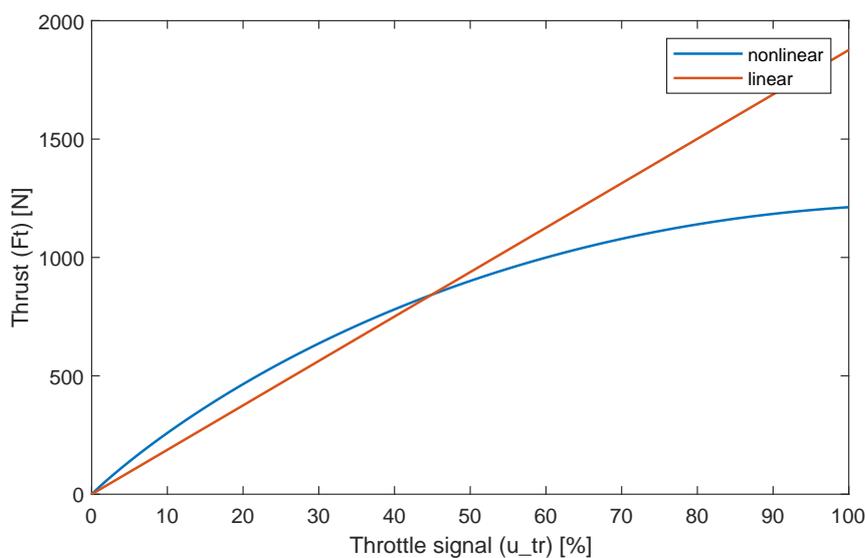


Figure 3.7: The nonlinear and linear thruster force model, assuming a linear relationship between throttle signal and steady state velocity.

3. Dynamics Modelling

Using the tuned parameters found in Table 3.7 the following velocity plots can be seen in Figure 3.9. The surge velocity over the same data interval is also included since it's behaviour could also change on some occasions even though the parameter $X_u(u)$ was not modified during the process.

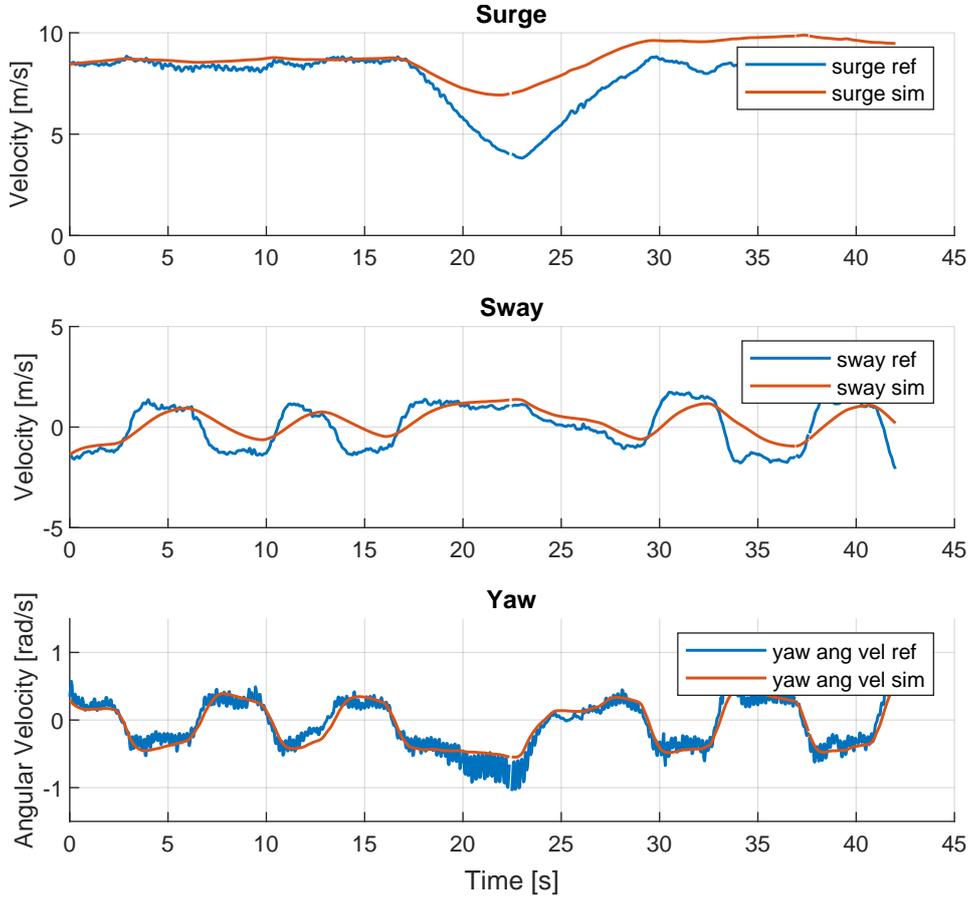


Figure 3.9: A comparison between logging data and simulation output after tuning both the sway and yaw angular velocity resistance polynomials.

Note in Table 3.7 that the expressions for resistance uses the absolute value of the velocity when evaluating the polynomial and combines it with the sign of the velocity.

Table 3.7: List of turning dynamics tuning parameters

Parameter	Description	Value
$Y_v(v)$	Sway resistance	$-(300 v + 20)$
$N_r(r)$	Yaw resistance	$-(1500 r + 130)$
$k_{m\dot{v}}$	Added mass scaling in sway	0
$k_{m\dot{r}}$	Added mass scaling in yaw	0.3

3.8 Discussion

While some significant discrepancies remaining between the simulation and test drive data the model was accurate enough for use in model based control design. One reason for the inaccuracies could be the assumed independence of the sway and yaw dynamics but also from mistakes done during the manual tuning process of the turning parameters. The surge model was also assumed to be independent from the turning dynamics and gave accurate results in both acceleration and coast down testing. However, during sharp turning maneuvers this assumption breaks down which can be seen in Figure 3.9 where the surge velocity is estimated significantly higher in the simulation.

For improving the dynamics model a recommendation would be to not assume that the states are completely independent. This would probably lead to a more accurate simulation but only if the correct parameters can be identified which would become harder to do manually when they are no longer independent from each other. Using more advanced system identification methods as in [7] would help with accurate identification based on the collected testing data.

Another aspect that could be looked at in future work is modeling for slow and even reversed speeds. The current model is limited to positive surge velocities which is enough for following waypoints but more precise navigation such as docking will require models that also include thrust reversal.

4

Waypoint Following

To evaluate the performance of the MARV control system together with the derived model and control scheme, a waypoint following algorithm has been developed and implemented as a scenario. The derivation of the controllers, guidance algorithm, simulation and real test drive results is presented below.

To simplify the design process a decision was made to split the waypoint following in two different subsystems. One guidance part that is responsible for trajectory generation and a control system that follows the trajectory. This is a common way to implement waypoint control and a similar method was used in [3].

4.1 Controller Design

Based on the independence of the surge and turning dynamics of the manoeuvring model the decision was made to use separate controllers for following a heading trajectory and a surge trajectory respectively. This has been a common control strategy in previous work and has been used in [3] and [4]. Another contributing reason is that the thruster configuration of the WR makes the system under actuated. This makes it impossible to control the full state space of the system which requires limiting the control to the surge and yaw state as discussed in [19].

The controller selection was also limited to linear controllers with the possible use of gain scheduling if required. This made a model based controller an attractive choice as it would make gain scheduling easier to implement if it became required. Based on the overview of course keeping control done in [6] linear quadratic regulators (LQRs) were selected for both the surge and heading controller. This choice was supported by the navigation system supplying full state feedback negating the need for designing observers.

4.1.1 Surge Controller

Separating out the surge dynamics from the maneuvering Equation 3.2 gives

$$\dot{u}(t) = \frac{X_u(u)}{m - X_{\dot{u}}} u(t) + \frac{X(t)}{m - X_{\dot{u}}} \quad (4.1)$$

where $X(t)$ is the force from the waterjet thruster in the surge direction.

Based on the thruster model in Eq. 3.18 that assumes proportionality between surge speed and throttle input a linear approximation for thruster force is introduced as

$$X^*(u_{tr}(t)) = F_k u_{tr}(t). \quad (4.2)$$

The full thruster force is assumed to lay in the surge direction which means current steering angle can be disregarded.

A linear approximation of surge resistance is also developed in Section 3.5 by fitting a linear curve against $X_u(u)$ with focus on a good fit within the interval 0 to 10 m/s which gives the linearised dynamics as

$$\dot{u}^*(t) = \frac{X_u^*}{m - X_{\dot{u}}} u^*(t) + \frac{F_k}{m - X_{\dot{u}}} u_{tr}(t). \quad (4.3)$$

After adding an augmented integral state the dynamics in Eq. 4.3 can be written in state space form as

$$\dot{x}_u(t) = \begin{bmatrix} 0 & 1 \\ \frac{X_u^*}{m - X_{\dot{u}}} & 0 \end{bmatrix} x_u(t) + \begin{bmatrix} 0 \\ \frac{F_k}{m - X_{\dot{u}}} \end{bmatrix} u_{tr}(t) \quad (4.4)$$

where

$$x_u(t) = [\int u_e \quad u_e]^T \quad (4.5)$$

and the u_e states denote surge error to allow reference tracking.

Based on the linear model, state cost matrix Q_u and output signal cost R_u an LQR is designed using MATLAB *lqr* function. Anti-windup of the integral state is also added via the back calculation method presented in [26] with the feedback connected according to Figure 4.1. Reference shaping is also applied using both a

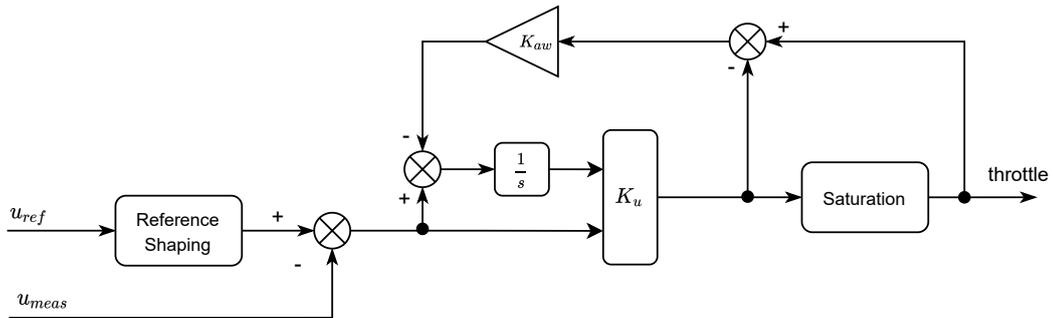


Figure 4.1: Surge controller block diagram

low pass filter, as recommended in [19], with the transfer function

$$\frac{\omega_u^2}{s^2 + \zeta_u \omega_u s + \omega_u^2} \quad (4.6)$$

and with a rate limiter with a maximum velocity change u_{reflim} per second.

A discrete version of the controller is also calculated using *lqrd* and corresponding finite impulse response filters for the reference shaping. The integrator is approximated using the forward Euler method.

Table 4.1: List of surge controller parameters

Parameter	Description	Value
$F_k(u_{ss})$	Throttle to force scaling	18.76 N/%
\mathbf{Q}_u	LQR state cost matrix	diag $\left(\left[\begin{smallmatrix} 300 & 1000 \end{smallmatrix}\right]\right)$
R_u	LQR output cost	1
ω_u	Surge reference filter cut off frequency	3.6 rad/s
ζ_u	Surge reference damping ratio	0.8
u_{reflim}	Surge reference rate limit	± 1.5 m/s ²
$u_{trsatsat}$	Saturation limit of throttle	70 %
K_{aw}	Anit-windup feedback gain	-3

4.1.2 Heading Controller

Based on the presented manoeuvring model in Section 3.3 the yaw dynamics can be written separately as

$$\dot{r}(t) = \frac{N_r(r)}{I_z - N_{\dot{r}}k_{m\dot{r}}}r(t) + \frac{N(t)}{I_z - N_{\dot{r}}k_{m\dot{r}}}. \quad (4.7)$$

To apply linear control system design both the yaw resistance $N_r(r)$ and moment applied by the thruster $N(t)$ need to be linearised. The moment applied by the thruster is modeled as

$$N(t) = -\sin(\Psi)F_{ss}(u_{ss})l \quad (4.8)$$

based on Equation 3.19 but with the thrust F_{ss} being the force required to maintain a steady state surge velocity.

Linearising equation 4.8 around a steering angle Ψ of zero degrees and an assumed steady state surge velocity u_{ss} gives

$$N^*(u_{ss}) = -lF_{ss}(u_{ss})\Psi(t). \quad (4.9)$$

A linear approximation of the resistance N_r^* is also found by applying *polyfit* to the resistance polynomial focusing on a good fit in the interval of 0 to 0.6 rad/s as that was the approximate range of angular velocities observed in the test drives.

Based on Equations 4.7 and 4.9 the yaw dynamics can now be written in linearised form as

$$\dot{r}^*(t) = \frac{N_r^*}{I_z - N_{\dot{r}}k_{m\dot{r}}}r^*(t) - \frac{lF_{ss}(u_{ss})}{I_z - N_{\dot{r}}k_{m\dot{r}}}\Psi(t). \quad (4.10)$$

The state that needs to be controlled however is the heading angle $\psi(t)$ which is simply the integral of $r(t)$. Combining the heading angle, with Equation 4.10 and an additional augmented state providing integrated heading angle gives the system on state space form as

$$\dot{x}_\psi(t) = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ \frac{N_r^*}{I_z - N_{\dot{r}}k_{m\dot{r}}} & 0 & 0 \end{bmatrix} x_\psi(t) + \begin{bmatrix} 0 \\ 0 \\ -\frac{lF_t(u_{ss})}{I_z + N_{\dot{r}}k_{m\dot{r}}} \end{bmatrix} \Psi(t) \quad (4.11)$$

where

$$x_\psi(t) = \begin{bmatrix} \int \psi_e(t) & \psi_e(t) & r(t) \end{bmatrix}^T. \quad (4.12)$$

Note that the two heading states have been expressed as error states that will take a heading reference as input. The yaw state does not have a reference so here the controller will instead act to drive this state towards zero.

Based on the state space model, state cost matrix \mathbf{Q}_ψ and output signal cost R_ψ an LQR is designed using MATLAB *lqr*. Gain scheduling based on steady state surge velocity was considered but the controller gains did not change significantly over the speeds of interest so instead u_{ss} was set constant at 10 m/s.

The controller is complemented with saturation limiting the maximum and minimum commanded steering angle Ψ_{sat} . To prevent the integral state from increasing a back calculation anti-windup method presented in [26] has been used. The anti-windup feedback gain K_{aw} has been tuned manually together with the LQR costs.

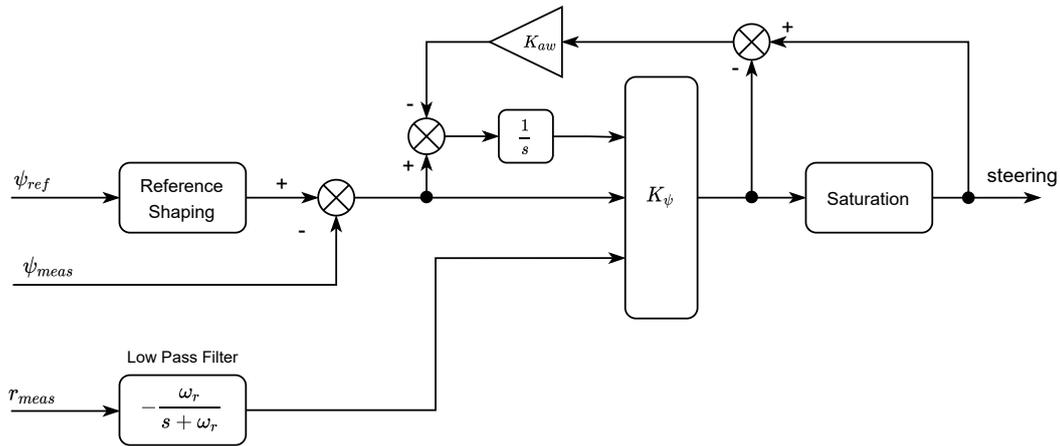


Figure 4.2: Heading controller block diagram

To prevent rapid changes in the controller input signal filtering to both r_{meas} and ψ_{ref} are applied. For the yaw measurement this is done with a first order low pass filter with the transfer function presented in Figure 4.2. For the heading reference a second order low pass filter, as recommended in [19], with the transfer function

$$\frac{\omega_\psi^2}{s^2 + \zeta_\psi \omega_\psi s + \omega_\psi^2} \quad (4.13)$$

is combined with a rate limiter allowing a maximum of ψ_{reflim}°/s . For the reference shaping to be correctly applied the heading angle first needs to be unwrapped and then wrapped back to the interval $[-\pi, \pi]$.

To allow for implementation in the MARV system the controller has also been discretized. For the LQR this was done through *lqrd* that uses the zero order hold

method. The impulse response of the reference shaping filters was sampled to create finite impulse response filters and the anti-windup feedback uses the forward Euler method to approximate the integration.

Table 4.2: List of heading controller parameters

Parameter	Description	Value
N_r^*	Linear yaw resistance	850 Nm/s
u_{ss}	Steady state velocity in surge	10 m/s
$F_t(u_{ss})$	Constant force at steady state surge	1052 N/m
\mathbf{Q}_ψ	LQR state cost matrix	diag $\left(\begin{bmatrix} 200 & 1 & 1 \end{bmatrix} \right)$
R_ψ	LQR output cost	100
ω_ψ	Heading filter cut off frequency	3 rad/s
ζ_ψ	Heading filter damping ratio	2
ψ_{reflim}	Heading reference rate limit	$\pm 20^\circ/\text{s}$
ω_r	Measured yaw filter cut of frequency	10 rad/s
Ψ_{sat}	Saturation limits of steering angle	$\pm 24^\circ$
K_{aw}	Anit-windup feedback gain	5

4.2.1 Derivation of All Variables

Several variables are already known. The variables \mathbf{p} , ψ are known since they are gathered directly from the INS. While $\mathbf{p}_{wp,n}$, $\mathbf{p}_{wp,n+1}$, r_n , $b_{v,n}$, $b_{v,n+1}$ and $v_{c,n}$ are planned in advance and is a part of the waypoint path planning data. The last known variable is S , which is set to be the previous waypoint, or the MARV's current location at the start. The rest of the variables are calculated as follows.

To begin with, the reference heading vector is

$$\mathbf{h}_{wp} = \begin{bmatrix} h_{wp_x} \\ h_{wp_y} \end{bmatrix} = \mathbf{p}_{wp,n} - \mathbf{p} \quad (4.14)$$

and this provides the reference heading angle as

$$h_r = \text{atan2} \left(\frac{h_{wp_y}}{h_{wp_x}} \right). \quad (4.15)$$

Then heading error is given by

$$h_e = \psi - h_r \quad \text{where } h_e \text{ must be wrapped and } \in [-\pi, \pi]. \quad (4.16)$$

Continuing with the current bridge vector which is given as

$$\mathbf{b}_n = \mathbf{p}_{wp,n} - S \quad (4.17)$$

and then the next bridge vector in the same way

$$\mathbf{b}_{n+1} = \mathbf{p}_{wp,n+1} - \mathbf{p}_{wp,n} \quad (4.18)$$

which makes it possible to calculate the angles between the bridges as

$$\alpha = \text{acos} \left(\frac{\mathbf{b}_n \cdot \mathbf{b}_{n+1}}{|\mathbf{b}_n| |\mathbf{b}_{n+1}|} \right). \quad (4.19)$$

4.2.2 Velocity Adaptation

As mentioned the MARV should be able to adapt its velocity in order to improve the travelled path. This is done in two ways.

The first way is to reduce the velocity if the current heading deviates too much from the reference heading. This is useful since it is harder to correct larger reference heading errors when the velocity is higher. This is both limited by the physical system, where the stepper motor driving the pulley system is not strong enough to overcome the forces presented at higher speeds. It is also because of the fact that the higher the velocity, the longer time it takes for the reference heading error to converge to zero. This is done by the following calculations.

First it is necessary to define an angle error tolerance, v_{diff} , which within the MARV should travel at the pre-planned current bridge velocity $b_{v,n}$. Outside of this tolerance the algorithm should reduce the velocity depending on how large h_e is.

The reduction should at most lower the velocity to the minimum specified velocity u_{min} . The minimum specified velocity is needed since it is not possible to steer if there is no movement.

The velocity should be the minimum if $h_e = -\pi$ or π , since this means that the MARV have to turn around. Using the nature of the sine function the velocity reduction v_{red} can be done in a smooth manner and is then calculated as

$$v_{red} = \sin(0.5|h_e| - v_{diff})(b_{v,n} - u_{min}) \quad (4.20)$$

and the new velocity reference to use v_r as

$$v_r = b_{v,n} - v_{red} \quad (4.21)$$

or else when within the tolerance simply as

$$v_r = b_{v,n}. \quad (4.22)$$

The second action builds on the same idea but is now using the angles between the bridges. The velocity should be reduced depending on how large the angle is when the vessel is about to turn to the next waypoint. The larger the angle, the less the vessel need to slow down. The extreme case is when the two bridge vectors are parallel, pointing in the same direction. Then it should keep the current bridge velocity. The opposite situation is when the angle is zero and they are pointing in different directions. Then the velocity should be at the minimum. The turning velocity is given by

$$v_{turn} = \sin(0.5|\alpha|)(b_{v,n} - u_{min}). \quad (4.23)$$

When the vessel is inside the velocity change circle, $|\mathbf{p}_{wp,n} - \mathbf{p}| < v_{c,n}$ the velocity should be begin to reduce. This is done in a single reference step, from the current bridge velocity to the turn velocity reference. It will take a certain amount of time depending on the system's dynamics for the velocity to be changed. This also depends on the time it takes between the velocity reference change and system response δ_t . When the vessel reaches the inner circle where $|\mathbf{p}_{wp,n} - \mathbf{p}| < r_n$ it should be at the turning velocity in order to be able to perform a correct turn. Since the waypoint radius is pre-planned and known this only leaves the outer velocity change radius to be calculated. Using the known surge rate limitation s_r , defined in the surge controller, it becomes

$$v_{c,n} = r_n + \delta_t b_{v,n} + \frac{v_{turn}^2}{2s_r}. \quad (4.24)$$

The expression is simply based on the normal distance formula when accelerating. The new velocity reference is then set to

$$v_r = v_{turn}. \quad (4.25)$$

4.2.3 Algorithm

Combining the previously derived expressions it is now possible to present a pseudo version of the guidance algorithm. Following is a simplified version of the pseudo algorithm, the complete one can be found in appendix A.1. The difference is that the full one also takes care of the edge cases that arise when on the last waypoint.

Algorithm 1: Simplified MARV waypoint pseudo guidance algorithm

initialise necessary variables;

while *there are waypoints left* **do**

 set start position to the previous waypoint, if on first, set once to current position;

 calculate heading vector and angle, \mathbf{h}_{wp} and h_r ;

if *reached last waypoint* **then**

 | set finished state, stop the vessel;

else

 | calculate bridges and angle between, \mathbf{b}_n and \mathbf{b}_{n+1} , then α ;

 | calculate turning velocity and change radius, v_{turn} and $v_{c,n}$;

if *inside of velocity change radius* $v_{c,n}$ **then**

 | lower vel ref, set to, $v_r = b_{v,n} - v_{turn}$;

else

 | calculate heading error h_e ;

 | wrap h_e to be $\in [-\pi, \pi]$;

if *outside of angle error tolerance*, $h_e > v_{diff}$ **then**

 | calculate the velocity reduction v_{red} ;

 | set vel ref, $v_r = b_{v,n} - v_{red}$;

else

 | set vel ref, $v_r = b_{v,n}$;

if *reached waypoint, i.e. within waypoint radius*, r_n **then**

 | keep track that the waypoint $W_{p,n}$ has been reached, aim towards the next waypoint;

 | calculate the next reference heading vector, pointing towards the

 | next waypoint, $\mathbf{h}_{wp_next} = \mathbf{p}_{wp,n+1} - \mathbf{p}_{wp,n}$;

 | update the heading reference angle to point towards the next

 | waypoint, $h_r = \text{atan2}\left(\frac{h_{wp_next_y}}{h_{wp_next_x}}\right)$;

if *has previously been inside of the waypoint radius, and once again outside of*, r_n **then**

 | reset the keep track variable, increase waypoint counter to target the next waypoint.;

 | calculate the same \mathbf{h}_{wp_next} and h_r as above.;

4.2.4 Planning

In order to plan a path made of successive waypoints a Matlab script has been written, the WaypointPlanner. The script can be found in the MARV-Software repository [27]. It uses maps taken from Lantmäteriet's map service [28] that has been configured using the WaypointPlanner_ConfigureMap script, using parameters such as the map image pixel density, scale and geographic coordinates reference point. This provides a map that can be used by the WaypointPlanner to plan a path with points in the NED coordinate system. The script allows editing and exporting functions for both the simulation and implemented scenario. An image of the map and user interface can be seen in Figure 4.4. The script requires the user to enter u_{min} , δ_t , s_r and v_{diff} parameters in order to correctly show the computed path.

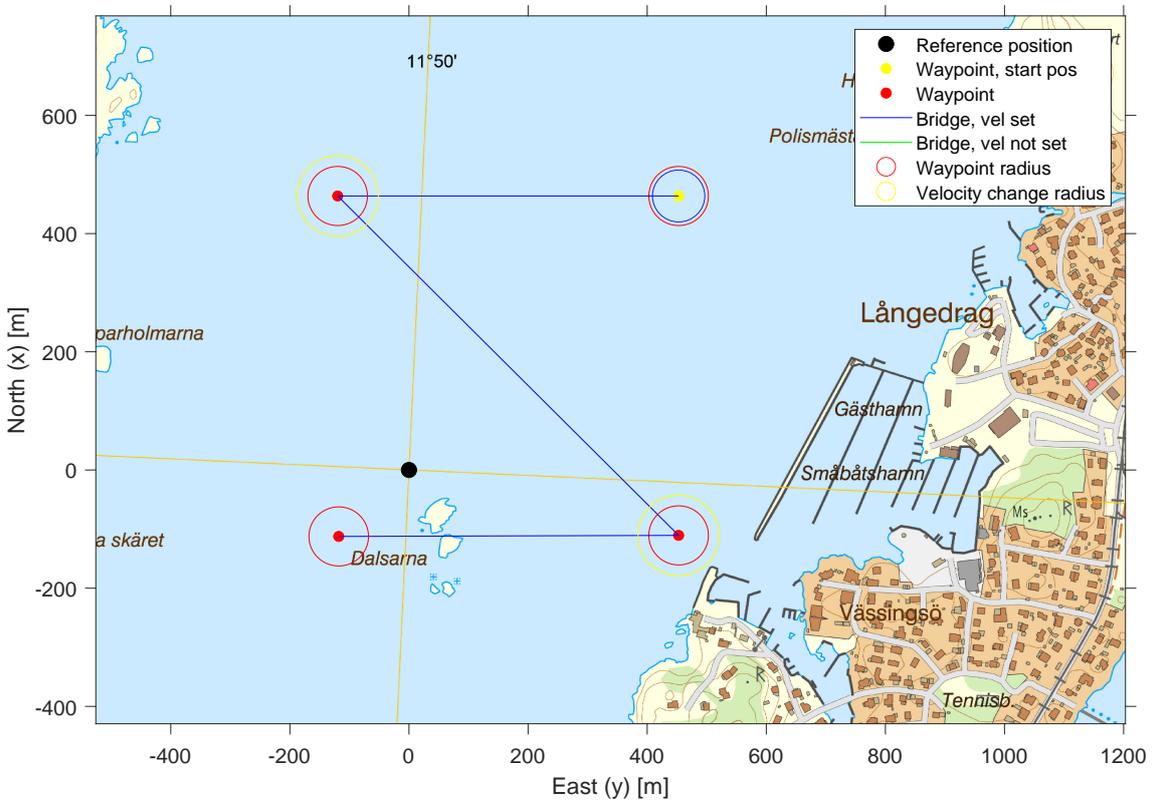


Figure 4.4: Showing the WaypointPlanner script's map, including the different components presented in the guidance algorithm.

4.3 Results

Using the derived controllers, guidance algorithm and the following parameters $u_{min} = 3.0 \text{ m/s}$, $\delta_t = 1.0 \text{ m}$, $s_r = 1.5 \text{ m/s}^2$ and $v_{diff} = 5.0^\circ$, the following results are achieved. The path following is seen in Figure 4.5 for both simulation and a practical test drive. The weather during the test drive presented wind of about $6 - 8 \text{ m/s}$ up to 12 m/s in the gusts. There was also a fair amount of waves.

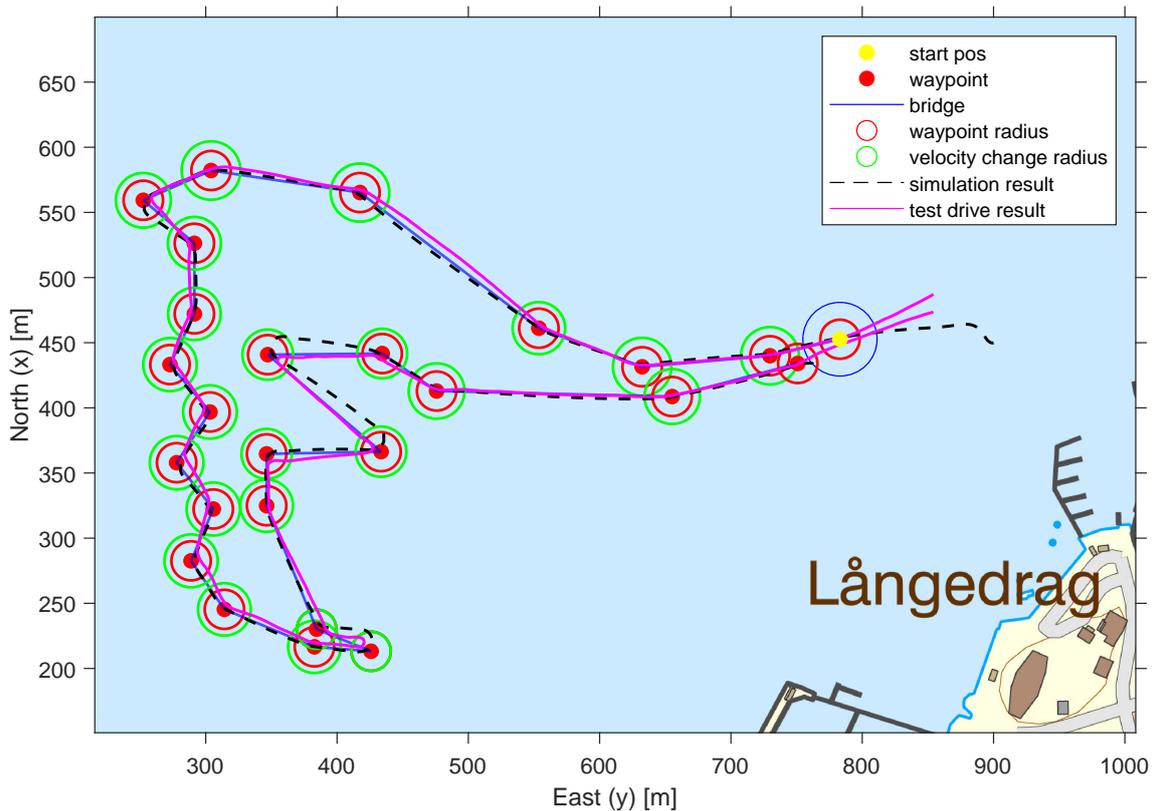


Figure 4.5: The waypoint navigation results, for both simulation and practical test drive.

Also showing only a comparison of the results without the planned path in Figure 4.6.

It can be seen that the two paths are close to each other. This means that, the model, synthesised controllers and guidance algorithms works as they are supposed to. There is however an apparent difference. Looking at Figure 4.7 is can be seen that the simulation tends to overshoot a bit, deviating from the connecting bridges. The test drive result tends to do the opposite by sometimes turning earlier.

4. Waypoint Following

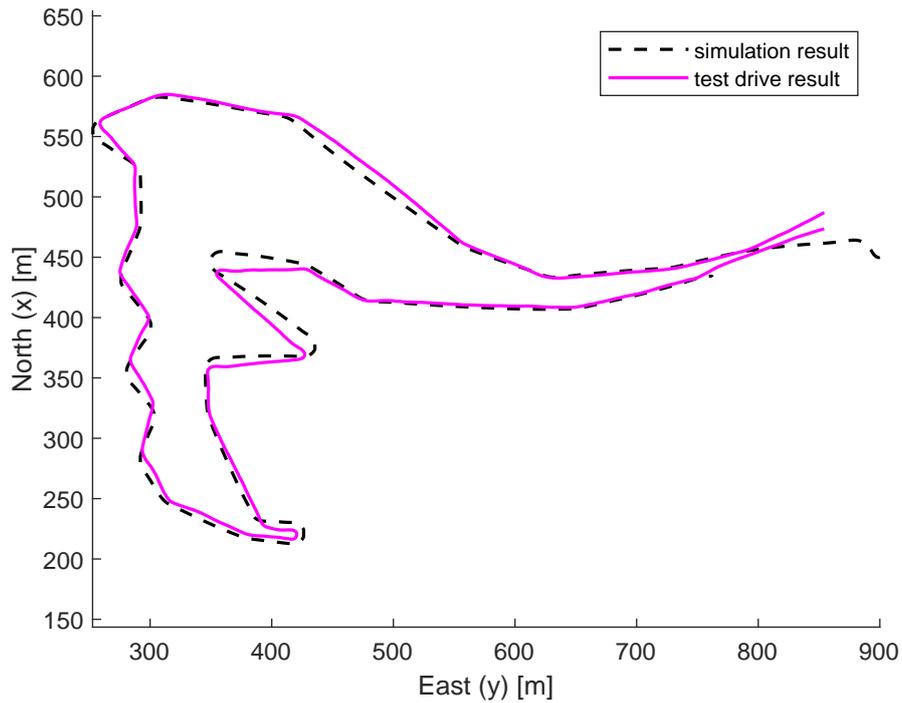


Figure 4.6: The waypoint navigation results, showing only the simulation and test drive results.

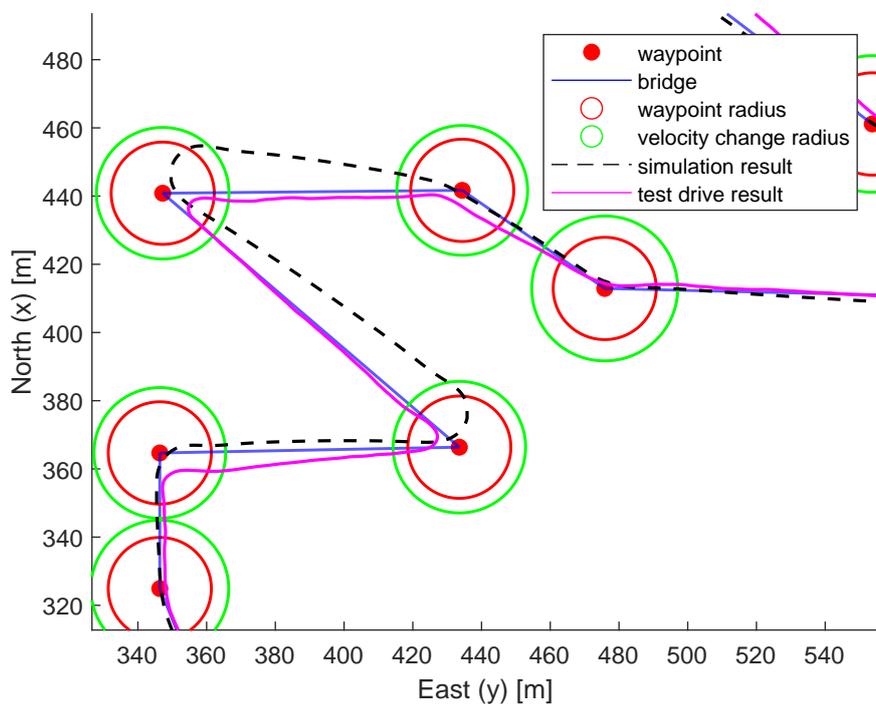


Figure 4.7: The waypoint navigation results, zoomed in.

Moving on to the results between reference signals and resulting output. In Figure 4.8 a comparison between the waypoint reference (b_v), guidance algorithm output (v_r) vessel velocity can be seen for both simulation and test drive. The guidance

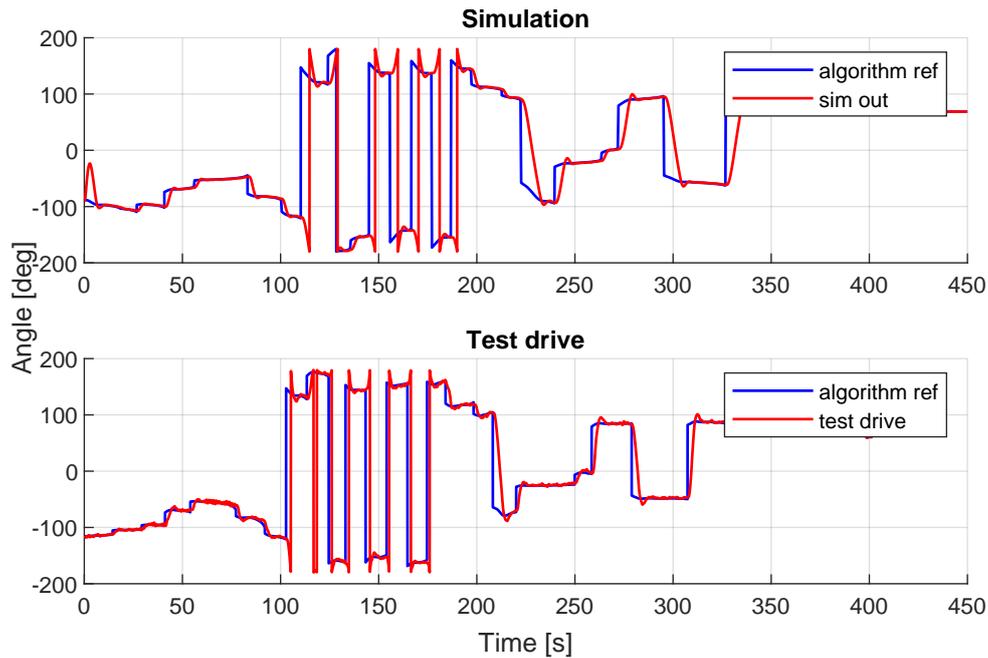


Figure 4.9: A comparison between the guidance algorithm generated heading reference and resulting heading for both simulation and test drive.

The guidance algorithm velocity reference should not pass above the waypoint reference at any time. This was caused due to a scenario implementation bug where some values came in as degrees instead of radians. This caused the velocity reduction v_{red} to be the maximum possible, as soon as the heading error h_e became larger than the error tolerance v_{diff} . This caused the system to travel at u_{min} velocity. In the same way it causes the reference to be higher than the waypoint reference. Fixing the bug and running the algorithm again using the recorded test drive data yields the following result seen in Figure 4.10.

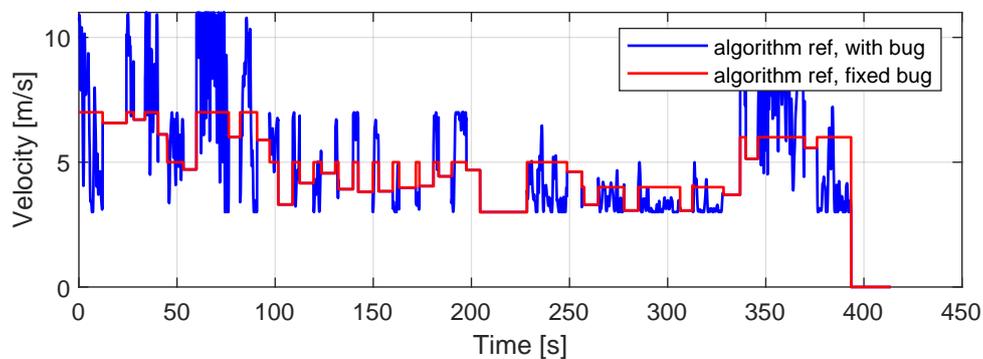


Figure 4.10: A comparison with the guidance algorithm scenario implementation bug, with and without.

4.4 Discussion

Overall the results shows successful implementation of successive waypoint tracking but some unexpected behaviour was detected which will be discussed in this section.

In the test drive the velocity varied more than expected. This can partly be due to the implementation bug which lowered and increased the velocity reference incorrectly. Another possible reasons is that the current velocity estimate propagated high frequency disturbances to the surge controller leading to an unnecessarily active throttle signal. During the test drive there was strong winds and high waves which possibly contributed to large disturbances.

Another observed difference was the model tends to under-steer in low velocities during large heading changes. This is somewhat expected based on the model tuning results where sharp turns resulted in deviations from the recorded test drive data. This could probably be significantly improved if a more complete model was developed without the simplifications of independent state variables. The dynamical model also over estimates the surge velocity in tight turns which can be seen in Figure 3.9 which could explain the larger turning radius.

5

Conclusion

This thesis has aimed to cover several parts related to the MARV testbed. It began with the system description that was supposed to give an introduction to the system as well as a basic understanding. It then continued on to the verification part by creating a model of the craft. The model was then used to synthesize the LQR controllers, which together with the guidance algorithm successfully managed to achieve waypoint following.

5.0.1 Evaluation of Q1

The successful behaviour of the system has been achieved by taking care of several important design aspects which enables future work and therefore increases the longevity of the platform. The most important aspect is the robustness which ensures that the system performs as expected at all times. This is however not something that is achieved in isolation. It is affected by the whole system which includes a well planned architecture, stable communication, reliable safety mechanisms, robust hardware and functional software. It still remains to be seen how the platform holds up over time but the verification shows an almost fully working system.

On top of the robustness aspect it has been equally necessary to provide the future users with an interface that is easy to understand with good abstractions. This is because the system itself has become quite complex and hard to understand without significant time investment. Time that can instead be used for the task at hand.

Another important aspect of the design has been to allow for complete logging of system signals. This is important since it allows for a more thorough analysis of the real behaviour of an algorithm and system dynamics. It is possible to go into the details and compare data at specific times.

The last aspect is the system extensibility and modularity. The system must be ready for more advanced research in the future. With a modular design methodology it is possible to both add new units and sensors as needs arise. Depending on how these additions are made the current system's performance could be affected. Because of the CAN problem it can be problematic to add new units directly to the DbW system. This is something that needs to be taken care of.

5.0.2 Evaluation of Q2

For successive waypoint following during real world conditions it has been proven that the derived model, controllers and guidance methods are functional. The modified and implemented pure pursuit guidance algorithm seemed to be applicable for this area. The model and controllers do however have their drawbacks.

The derived model of the system is simplified and relies on several assumptions. The two LQR controllers performed good enough for allowing successful maneuvering of the craft. The turning dynamics model did however tend to under steer in low velocities and the velocity controller over compensated the throttle signal for disturbances in velocity changes.

In order to further improve the behaviour of the controllers a better model should therefore be derived which captures a broader band of the real world behaviour, including disturbances. Due to the system's non-linear nature it may also be interesting to try out gain scheduled or non-linear controllers.

5.0.3 Final Words

The testbed is designed with robustness and safety in mind and many precautions has been taken in order to increase the quality of the outcome. It is really exciting to see that it turned out as well as it finally did. Except for the minor issues that can be improved it generally functions as originally envisioned. It still remains to see how of the MARV platform will be used in the future. Hopefully it will help accelerate and support further research within the area, and maybe even in the future realise SSRS's vision of a fully autonomous rescue vehicle.

Bibliography

- [1] E. Alfredsson, N. Danielsson, A. Gunnarson, V. Lindström, B. Lundgren, and J. Sjöberg, “Autonomous waverunner-drive by wire,” B.S. thesis, Chalmers University of Technology, 2019. DOI: 20.500.12380/257413.
- [2] E. Ingemarsson, F. Kerstis, P. Lindesson, M. Löfgren, H. Male, and F. Milqvist, “Autonomous waverunner-follow the leader,” B.S. thesis, Chalmers University of Technology, 2019. DOI: 20.500.12380/257414.
- [3] A. Bergholtz and C.-A. Hernvall, “Waypoint-based path following system for a jet ski,” M.S. thesis, Chalmers University of Technology, 2015. DOI: 20.500.12380/218514.
- [4] A. Alkaysi and J. Voigt, “Autopilot for a personal watercraft,” M.S. thesis, Dept of Automatic Control, Lund University, 2020. [Online]. Available: <http://lup.lub.lu.se/student-papers/record/9024752>.
- [5] E. Alfredsson, N. Danielsson, V. Lindström, and B. Lundgren, “Estimating dynamic model of a personal watercraft using first principles and numerical simulation,” unpublished, 2020.
- [6] M. N. Azzeri, F. A. Adnan, and M. Z. Md. Zain, “Review of course keeping control system for unmanned surface vehicle,” *Jurnal Teknologi*, vol. 74, no. 5, May 2015. DOI: 10.11113/jt.v74.4635.
- [7] C. H. Svendsen, N. O. Hoick, R. Galeazzi, and M. Blanke, “L1 adaptive manoeuvring control of unmanned high-speed water craft,” *IFAC Proceedings Volumes*, vol. 45, no. 27, pp. 144–151, 2012, 9th IFAC Conference on Manoeuvring and Control of Marine Craft, ISSN: 1474-6670. DOI: 10.3182/20120919-3-IT-2046.00025.
- [8] N. Wang and H. R. Karimi, “Successive waypoints tracking of an underactuated surface vehicle,” *IEEE Transactions on Industrial Informatics*, vol. 16, no. 2, pp. 898–908, 2019.
- [9] *Marv hardware mechanical*, <https://github.com/CASE-Lab/MARV-Hardware-Mechanical>. (accessed Sep. 12, 2021).
- [10] *Marv hardware electrical*, <https://github.com/CASE-Lab/MARV-Hardware-Electrical>. (accessed Sep. 12, 2021).
- [11] *Marv firmware*, <https://github.com/CASE-Lab/MARV-Firmware>. (accessed Sep. 12, 2021).
- [12] *Marv ros*, <https://github.com/CASE-Lab/MARV-ROS>. (accessed Sep. 12, 2021).
- [13] *Reach hardware*, <https://github.com/CASE-Lab/REACH-Hardware>. (accessed Sep. 12, 2021).

- [14] *Miniature high performance inertial sensors*, https://www.sbg-systems.com/wp-content/uploads/Ellipse_Series_Leaflet.pdf. (accessed Sep. 12, 2021).
- [15] *Marv test data*, <https://github.com/CASE-Lab/MARV-Test-Data>. (accessed Sep. 12, 2021).
- [16] *Dbc format*, http://socialledge.com/sjsu/index.php/DBC_Format. (accessed Aug. 1, 2021).
- [17] RB-Racing, *Professional mil-spec motorsport ecu wiring harness construction*, https://rbracing-rsr.com/wiring_ecu.html. (accessed Sep. 12, 2021).
- [18] SNAME, “Nomenclature for treating the motion of a submerged body through a fluid,” *The Society of Naval Architects and Marine Engineers, Technical and Research Bulletin*, no. 1–5, 1950.
- [19] T. I. Fossen, *Handbook of marine craft hydrodynamics and motion control*. John Wiley & Sons, 2011. DOI: 10.1002/9781119994138.
- [20] Z. Liu, C. Yuan, and Y. Zhang, “Linear parameter varying adaptive control of an unmanned surface vehicle,” *IFAC-PapersOnLine*, vol. 48, no. 16, pp. 140–145, 2015, 10th IFAC Conference on Manoeuvring and Control of Marine Craft MCMC 2015, ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2015.10.271.
- [21] T. I. Fossen, *Guidance and control of ocean vehicles*. Wiley, 1994, ISBN: 0471941131.
- [22] *Seawater*, <https://en.wikipedia.org/wiki/Seawater>. (accessed Jul. 28, 2021).
- [23] B. Šarkan, T. Skrucany, S. Semanova, R. Madleňák, A. Kuranc, M. SEJKO-ROVÁ, and J. Caban, “Vehicle coast-down method as a tool for calculating total resistance for the purposes of type-approval fuel consumption,” *Scientific Journal of Silesian University of Technology. Series Transport*, vol. 98, pp. 161–172, Mar. 2018. DOI: 10.20858/sjsutst.2018.98.15.
- [24] P. Holoborodko, *Smooth noise robust differentiators*, <http://www.holoborodko.com/pavel/numerical-methods/numerical-derivative/smooth-low-noise-differentiators/>, 2008.
- [25] Yamaha Motor Co., Ltd, “Waverunner vxs, vxr service manual,” unpublished.
- [26] J.-W. Choi and S.-C. Lee, “Antiwindup strategy for pi-type speed controller,” *IEEE Transactions on Industrial Electronics*, vol. 56, no. 6, pp. 2039–2046, 2009. DOI: 10.1109/TIE.2009.2016514.
- [27] *Marv software*, <https://github.com/CASE-Lab/MARV-Software>. (accessed Nov. 8, 2021).
- [28] *Kartutskrift - lantmäteriet*, <https://kartutskrift.lantmateriet.se/>. (accessed Nov. 8, 2021).

A

Appendix 1

A.1 Full waypoint pseudo algorithm

Algorithm 2: Full MARV waypoint pseudo guidance algorithm

initialise necessary variables;

while *there are waypoints left* **do**

 set start position to the previous waypoint, if on first, set once to current position;

 calculate heading vector and angle, \mathbf{h}_{wp} and h_r ;

if *reached last waypoint, i.e. on last waypoint AND within waypoint radius r_n* **then**

 | set finished state, stop the vessel;

else if *not on last waypoint* **then**

 | calculate bridges and angle between, \mathbf{b}_n and \mathbf{b}_{n+1} , then α ;

 | calculate turning velocity and change radius, v_{turn} and $v_{c,n}$;

if *inside of velocity change radius $v_{c,n}$* **then**

 | lower vel ref, set to, $v_r = b_{v,n} - v_{turn}$;

else

 | calculate heading error h_e ;

 | wrap h_e to be $\in [-\pi, \pi]$;

if *outside of angle error tolerance, $h_e > v_{diff}$* **then**

 | calculate the velocity reduction v_{red} ;

 | lower vel ref, set to, $v_r = b_{v,n} - v_{red}$;

else

 | set vel ref, $v_r = b_{v,n}$;

if *reached waypoint, i.e. within waypoint radius, r_n* **then**

 | keep track that the waypoint $W_{p,n}$ has been reached, aim towards the next waypoint;

 | calculate the next reference heading vector, pointing towards the next waypoint, $\mathbf{h}_{wp_next} = \mathbf{p}_{wp,n+1} - \mathbf{p}_{wp,n}$;

 | update the heading reference angle to point towards the next

 | waypoint, $h_r = \text{atan2}\left(\frac{h_{wp_next_y}}{h_{wp_next_x}}\right)$;

if *has previously been inside of the waypoint radius, and once again outside of, r_n* **then**

 | reset the keep track variable, increase waypoint counter to target the next waypoint.;

 | calculate the same \mathbf{h}_{wp_next} and h_r as above.;

else

 | calculate h_e ;

 | wrap h_e to be $\in [-\pi, \pi]$;

if *outside of angle error tolerance, $h_e > v_{diff}$* **then**

 | calculate the velocity reduction v_{red} ;

 | set vel ref, $v_r = b_{v,n} - v_{red}$;

else

 | set vel ref, $v_r = b_{v,n}$;

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se



CHALMERS
UNIVERSITY OF TECHNOLOGY