Bachelor's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of ICT and Natural Sciences

Marcus Ilstad, Jørgen Rottem, Joakim Sander Løken

Building a competitive Autonomous sea drone (ASD)

Using YOLO algorithm to navigate a sea drone

Bachelor's thesis in Electrical engineering, Automation engineering
Supervisor: Ottar Laurits Osen
May 2023

NTNU
Norwegian University of Science and Technology

Marcus Ilstad, Jørgen Rottem, Joakim Sander Løken

Building a competitive Autonomous sea drone (ASD)

Using YOLO algorithm to navigate a sea drone

Bachelor's thesis in Electrical engineering, Automation engineering
Supervisor: Ottar Laurits Osen
May 2023

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of ICT and Natural Sciences

NTNU
Norwegian University of
Science and Technology

NTNU
Kunnskap for en bedre verden

Department of ICT and Natural Sciences

Bachelor's Thesis

IELEA2920 - Bachelor Thesis Automation

Building a competitive Autonomous sea drone (ASD)

Authors:
Marcus Ilstad, Joakim Sander Løken and Jørgen Rottem

Date
21st May 2023

Supervisor: Ottar Laurits Osen

Preface

We put ourselves to the test by choosing this bachelor's thesis, which required us to put together the culmination of three years of work while also teaching us a great deal of new material. We can consider it an accurate reflection of life, in which we never stop gaining new knowledge.

This thesis' primary objective is to compete in the AutoDrone competition. The AutoDrone competition is an opportunity for us to apply the knowledge and abilities we've gained throughout our studies. We hope that our participation will not only highlight our skills, but also encourage others to pursue careers in this field. We also believe that future will have an increasing demand for the development of autonomous ships.

This report was composed at NTNU in Ålesund by three automation students with prior experience in the electrical industry. There may be unfamiliar terms in the report, but we recommend that you review the nomenclature, where we have attempted to list all abbreviations and other terms that may be difficult to comprehend. We also assume that you the reader of this thesis have basic knowledge in ROS and Linux.

Ålesund, 21st May 2023

Marcus Ilstad | Jørgen Rottem | Joakim Sander Løken

Acknowledgement

We would like to thank all the contributors, friends, and family members who have made this project possible. Throughout the process, their support and encouragement have been invaluable to us. Particularly, we would like to thank:

- Our supervisor Ottar Laurits Osen, without whose guidance and expertise this project would not have been possible. We are grateful for his unwavering dedication and commitment to our success.
- Laboratory engineers Anders Sætermoen and his team for assisting us in ordering parts and lending equipment.
- The members of the bachelor group from the previous year, Markus Grorud Gaasholt and Magnus Stava. Their willingness to share their experiences with us is greatly appreciated.

Executive summary

The purpose of this project is to build a capable ASD so that it can take part in the Autodrone competition in Horten. Machine vision will be utilized in order to allow the ASD to successfully navigate through the missions. The YOLOv8 algorithm was utilized to train a small YOLO model, which was then used for buoy detection.

The project produced an autonomous surface vessel (ASD) that is able to detect buoys, convert the detections to GPS locations, and navigate based on this information. Because of certain issues with the ZED2-camera's depth perception, the ASD was unable to perform its duties in a reliable manner.

In order to create an ASD that is both more effective and reliable than previous designs, the ASD's physical design and electrical design were both completely rethought from the earlier designs done at NTNU in Ålesund.

Table of Contents

List of Figures

List of Tables

Nomenclature

Physics Constants

R Radius of earth

Abbreviations

AI Artificial intelligence

API Application Program Interface

ASD Autonomous Sea Drones

AUV Autonomous underwater vehicle

CAD Computer-aided design

COLREGs Convention on the International Regulations for Preventing Collisions at Sea

DOF Degree of Freedom

ESC Electronic speed control

FOV Field Of View

FPS Frames Per Second

G-code Geometric Code

GNSS Global navigation satellite system

GPIO General-purpose input/output

GPS Global position system

I^2C Inter-Integrated Circuit

IMO International maritime organization

IMU Inertial measurement unit

IOU Intersection over union

mAP mean Average Precision

MASS Maritime Autonomous Surface Ship

PETG Polyethylene terephthalate glycol

RC Radio control

ROS Robot operation system

ROV Remotely operated vehicle

RTK Real-time kinematic

SD card Secure Digital card

SDK Software Development Kit

USB Universal Serial Bus

USV Unmanned surface vessel

wp Waypoint

YOLO You Only Look Once

Other Symbols

λ Degrees per pixel

θ Bearing angle

A Ampere

Ah Ampere-hour

I_B Load current

I_N Rated current

I_Z Current-carrying capacity

kgf kilogram-force

V Volts

hfov Horizontal Field Of View

lat Latitude

lon Longitude

T Translation from image center

Chapter 1

Introduction

1.1 Background

Autodrone is an annual competition in which national competitors operate autonomous sea drones. This event aims to promote technological progress and drone technology advancements, as well as strengthen maritime higher education in Norway.

1.2 Project introduction

Through the spring of 2023, an ASD was developed in order for NTNU in Ålesund to compete in the Autodrone competition in Horten against other universities. The team has amassed a wealth of knowledge in computer vision, computer programming, electrical design, and product design in general. This thesis will introduce the project and detail the design and construction of the ASD. In addition, it will highlight the obstacles and solutions encountered by the group throughout the duration of the project.

1.3 Aim and objectives

This thesis intends to design and develop an ASD that meets the requirements of the Autodrone competition. The project will involve the investigation, testing, and design of numerous ASD components, such as the propulsion system, control algorithms, and communication protocols. The primary goal is to develop an ASD capable of autonomously navigating a variety of obstacle courses and challenges. Listed below are the objectives of this thesis:

1. Designing an ASD that can serve as a foundation for future projects and meets the requirements of the competition, which are:
   (a) Speed mission
   (b) Collision avoidance
   (c) Obstacle channel
   (d) Docking
2. Developing a modular design
3. Having a completely electric ASD

1.4 Report content

This bachelor thesis is structured in the following way:

Chapter 2 - Theory - Presents necessary theory for the project.

Chapter 3 - Materials - Presents the necessary materials needed to carry out the project.

Chapter 4 - Method - Presents how the project was approached.

Chapter 5 - Results - Presents the results attained during the project.

Chapter 6 - Discussion - Discusses the approach, results and the group comes with suggestions for further development.

Chapter 7 - Conclusion - Concludes the project.

Chapter 2

Theory

2.1 Autonomous sea drones

ASD are model boats that can move on their own with the help of advanced sensors and algorithms. Batteries provide power for the boat's electric propulsion and steering. The future notion of autonomous ships, as described by IMO as "maritime autonomous surface ships", is quite popular. Smaller model vessels can serve as effective test beds for the development of innovative technologies and applications in the maritime environment. One of the most difficult features of ASD is their capability to navigate autonomously and avoid collisions when operating in severe weather and varying sea conditions [4].

2.2 Competition requirements

The competition requirements, as stated by AutoDrone, are as follows [29]:

- **Autonomy:** Drone shall be fully autonomous and shall have all autonomy decisions made onboard the ASD.
- **Communication:** The drone cannot send or receive any control information to and from Operators Control Station while in autonomous mode.
- **Deployable:** The ASD should be manually deployable.
- **Energy source:** The drone must be battery powered. All batteries must be sealed to reduce the hazard from acid or caustic electrolytes. The open circuit voltage of any battery (or battery system) may not exceed 60Vdc.
- **Kill Switch:** The drone must have at least one red button located on the drone that, when actuated, must instantaneously disconnect power from all motors and actuators.
- **Wireless Kill Switch:** In an emergency situation the operator control station must be able to actuate the kill switch on board the ASD.
- **Propulsion:** Any propulsion system may be used (thruster, paddle, etc.). However, all moving parts must have protection. For instance, a propeller must be shrouded.
- **Remote-controllable:** The drone must be remote-controllable from an operator control station.
- **Safety:** All sharp, pointy, moving or sensitive parts must be covered and marked.
- **Towable:** The drone must be towable.

- **Visual Feedback:** Teams are required to implement a visual feedback system, indicating status of their ASD. Additional information on this is available in Appendix 15.4 Visual Feedback.
- **Weight:** The entire maritime system (including UAV) shall weigh less than 70 kg.
- **Payload:** The drone must have a place to mount a go-pro action camera with an unobstructed view from the front of the drone.

More details about the requirements can be found in "Rules and Task Descriptions" [29].

2.3 Mission tasks

To demonstrate the autonomy of the ASD, the mission tasks are carried out. These tasks include obstacle avoidance, waypoint navigation, and docking at a designated location.

2.3.1 Obstacle channel

The ASD must pass through gates containing green and red buoys without touching them. Along the course there are placed yellow buoys that the ASD must be able to avoid. Se figure 2.1 for obstacle channel overview.

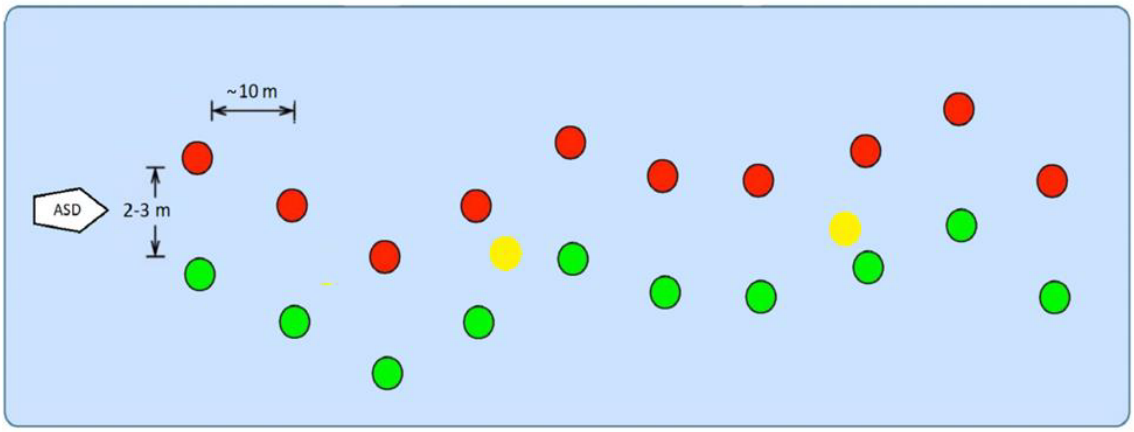


Figure 2.1: Obstacle channel overview from "Rules and Task Descriptions" [29].

Five points are awarded for each gate successfully passed, and ten points are deducted for failing to pass the yellow buoys. The starting score is 100 points, and 5 points are deducted for each passing minute.

2.3.2 Collision avoidance

The ASD must be able to safely traverse port and starboard traffic. This must be accomplished while following COLREGs at less than 4 knots and completing the mission task within 10 minutes. See Figure 2.2 for an illustration of collision avoidance.

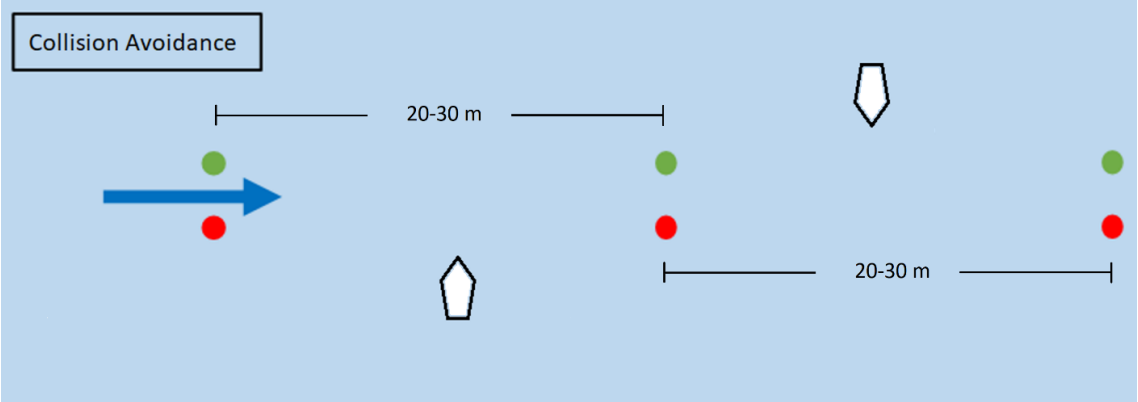


Figure 2.2: Collision avoidance illustration from "Rules and Task Descriptions" [29].

Ten points are awarded for each successfully traversed gate, and 30 points are deducted for collisions. Additionally, 30 points are awarded for compliance with each COLREGs rule.

2.3.3 Visual docking

The ASD must be able to dock with either the starboard or port side facing the pier. Additionally, the ASD must be positioned between a red and green buoy with a two-meter gap between them. Figure 2.3 depicts an example of this mission objective.

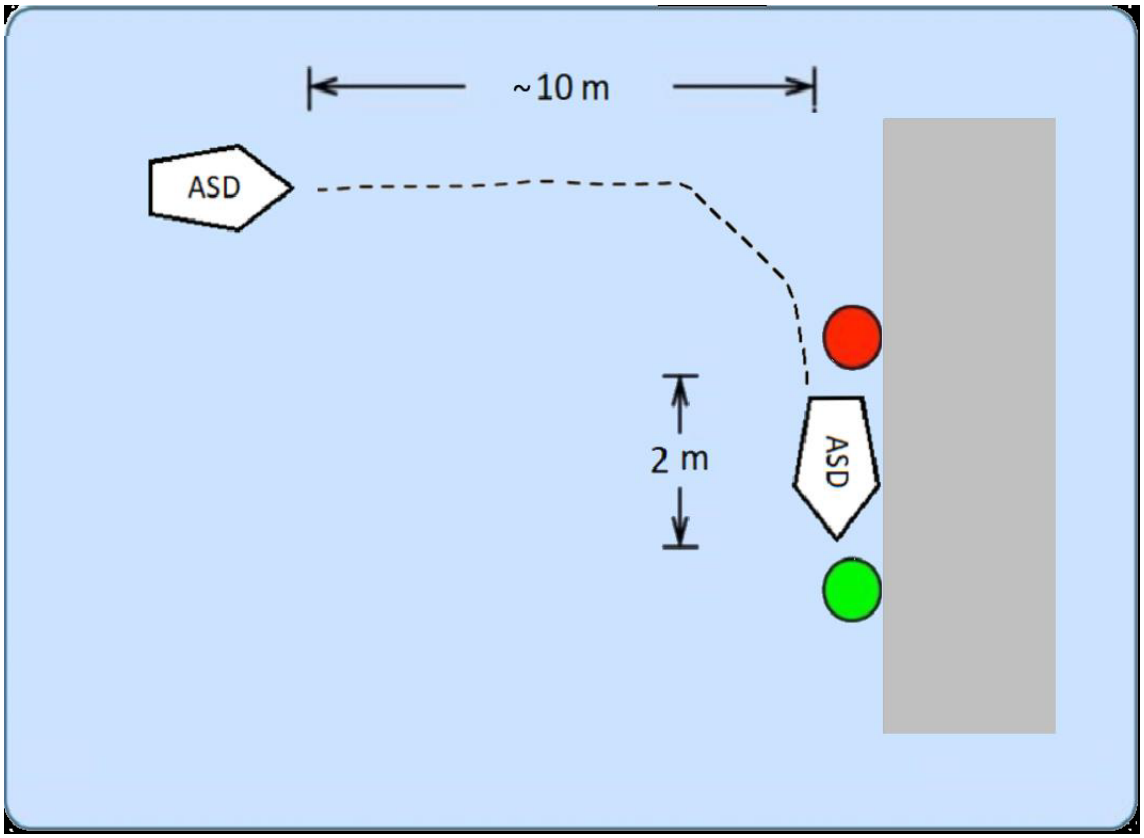


Figure 2.3: Visual docking illustration from "Rules and Task Descriptions" [29].

There are 10 points for reaching the dock and an additional 30 points for docking correctly. There are also given 2 points for each second docked, with a maximum of 60 points. The task must be

completed within a time limit of 10 minutes.

2.3.4 Speed gate

The drone must pass through the gate, circle the marker buoy, and then pass through the gate see figure 2.4 for a visual representation of the mission task.

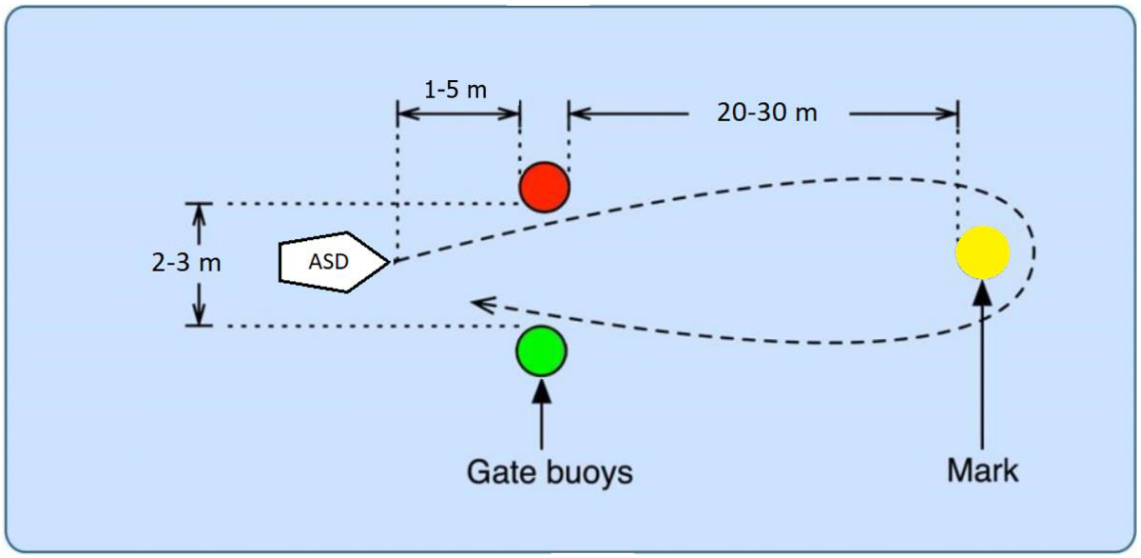


Figure 2.4: Speed gate illustration from "Rules and Task Descriptions" [29].

Twenty points are given to those who finish the course in less than ten minutes, and the three fastest ASDs get extra points.

2.4 RTK GPS

GNSS is the collection of satellite systems put out by various countries such as the EU's Galileo or Chinas BeiDou. These satellites allows for location estimation using position triangulation and timing data [31].

GPS triangulates position using satellites, but the accuracy can depend on weather conditions, signal interference and multipath errors. Real-time kinematic GPS is an addition to the GPS system used for improving the accuracy of the position estimate by using a stationary ground base as a reference point.

The ground base uses a technique called carrier phase tracking, where it measures the carrier phase of the GPS signal in addition to the code phase. The carrier phase is the actual phase of the GPS signal and varies by time, so by tracking it the base can determine the exact distance to the GPS satellites. The base then transmits this information to the moving GPS receiver that can use it for precise localization [1].

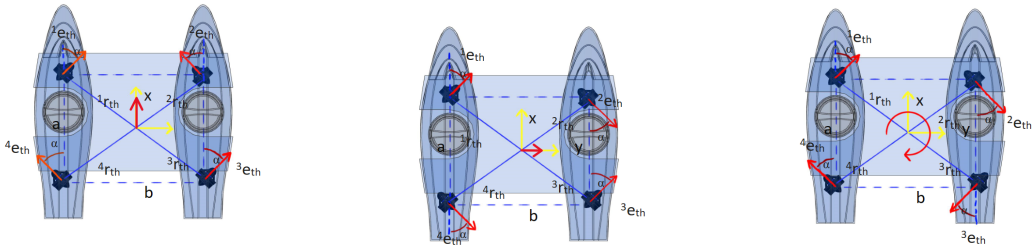
2.5 Dimensioning of cables and components

The fundamental principle of component dimensions is that circuit should always be cut before temperatures reach a point where cabling or parts are damaged. This is what is known as general demand 1 given by NEK [13]. The conductivity of the cables must be greater or equal to the

tripping current of the circuit breaker, which must be greater or equal to the anticipated current draw of the entire circuit, as shown in equation 2.1.

$$I_B \leq I_N \leq I_Z \tag{2.1}$$

2.6 Thruster configuration



(a) Surge (b) Sway (c) Yaw

Figure 2.5: Thruster directional configuration.

The forces in the different degrees-of-freedom as seen in figures 2.5a, 2.5b and 2.5c in a catamaran with thrusters configured in a X-configuration can be calculated in the following way:

The positional vectors for the thrusters in X-configuration are described in equation 2.2.

$$^1r_{th} = \begin{bmatrix} a/2 \\ -b/2 \\ 0 \end{bmatrix}, ^2r_{th} = \begin{bmatrix} a/2 \\ b/2 \\ 0 \end{bmatrix}, ^3r_{th} = \begin{bmatrix} -a/2 \\ b/2 \\ 0 \end{bmatrix}, ^4r_{th} = \begin{bmatrix} -a/2 \\ -b/2 \\ 0 \end{bmatrix} \tag{2.2}$$

The orientation vectors for the thrusters in X-configuration are described in equation 2.3.

$$^1e_{th} = \begin{bmatrix} cos\alpha \\ sin\alpha \\ 0 \end{bmatrix}, ^2e_{th} = \begin{bmatrix} cos\alpha \\ -sin\alpha \\ 0 \end{bmatrix}, ^3e_{th} = \begin{bmatrix} cos\alpha \\ sin\alpha \\ 0 \end{bmatrix}, ^4e_{th} = \begin{bmatrix} cos\alpha \\ -sin\alpha \\ 0 \end{bmatrix} \tag{2.3}$$

Given the thrusters are in a fixed position the total vector of propulsion forces and moments is given in equation 2.4, and the matrix in equation 2.5 can be derived.

$$\tau = \begin{bmatrix} \tau_X \\ \tau_Y \\ \tau_Z \\ \tau_K \\ \tau_M \\ \tau_N \end{bmatrix} = \sum_{i=1}^{p} {}^i\tau = \sum_{i=1}^{p} \begin{bmatrix} {}^ie \\ ({}^ir \times {}^ie) \end{bmatrix}^i T \tag{2.4}$$

$$\tau = \begin{bmatrix} cos\alpha & cos\alpha & cos\alpha & cos\alpha \\ sin\alpha & -sin\alpha & sin\alpha & -sin\alpha \\ 0 & 0 & 0 & 0 \\ cos\alpha \times a/2 & cos\alpha \times a/2 & cos\alpha \times -a/2 & cos\alpha \times -a/2 \\ sin\alpha \times -b/2 & -sin\alpha \times b/2 & sin\alpha \times b/2 & -sin\alpha \times -b/2 \\ 0 & 0 & 0 & 0 \end{bmatrix} Ku = TKu \tag{2.5}$$

Introducing substitution in equation 2.6 where **B** is the control matrix. **K** the force coefficient matrix and **u** the control vector. The substitution rewrites equation 2.4 to equation 2.7.

$$B = TK \tag{2.6}$$

$$\tau = Bu \tag{2.7}$$

Given that all thrusters are identical parameter **A** in equation 2.9 are given as equation 2.8. The zeroes in the control matrix seen in equation 2.9 are uncontrollable DOF.

$$A = (a/2)sin\alpha + (b/2)cos\alpha \tag{2.8}$$

$$\begin{bmatrix} cos\alpha & cos\alpha & cos\alpha & cos\alpha \\ sin\alpha & -sin\alpha & sin\alpha & -sin\alpha \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ A & -A & -A & A \end{bmatrix} K \tag{2.9}$$

Given the x-configuration as seen in figure 2.5 with the controlable DOF: surge, sway and yaw the thruster motion can be described as in equation 2.10. Given the thursters having different force coefficient in forward and reverse direction there is used K_1 and K_2 to describe this.

$$\tau = \begin{bmatrix} \tau_X \\ \tau_Y \\ \tau_N \end{bmatrix} = \begin{bmatrix} K_1 cos\alpha & K_1 cos\alpha & K_1 cos\alpha & K_1 cos\alpha \\ K_1 sin\alpha & -K_2 sin\alpha & K_1 sin\alpha & -K_2 sin\alpha \\ K_1 A & -K_2 A & -K_2 A & K_1 A \end{bmatrix} \begin{bmatrix} {}^1u \\ {}^2u \\ {}^3u \\ {}^4u \end{bmatrix} \tag{2.10}$$

2.7 Machine vision

Machine vision is a set of mathematical techniques used to make a machine or a computer perceive the world through vision like humans does. Through analysis of images and videos, patterns and objects can be determined and used for object detection, depth perception, 3D reconstruction and more [37].

2.7.1 Depth

When using a stereo camera, the depth of an object can be calculated by comparing the two images and calculating the disparity. The disparity of the object is inversely proportional to the distance of the object. That means if the object is close to the camera the disparity will be large, while if the disparity is low the object will be far from the camera [37].

2.7.2 YOLO

The You Only Look Once algorithm is a real time object detection algorithm, that applies a full image to a neural network. The image is divided in to regions by the network, and each region is used to predict and classify an object [26].

A common metric for evaluating performance for YOLO-models is the mean Average Precision, mAP. The mAP determines the mean average precision of the model, from the average precision of all the model's object classes. To calculate the average precision of each class, the precision-recall metric and Intersection over Union, IOU, is used. Precision is a measure of the model's accuracy for positive predictions, while recall is a measure of the proportion of positive cases the model is able to correctly predict. IOU is a measure of the quality of the predicted bounding box. The IOU is the ratio of intersection area between the predicted bounding box and the ground truth bounding box [38]. See figure 2.6 for illustration of IOU.

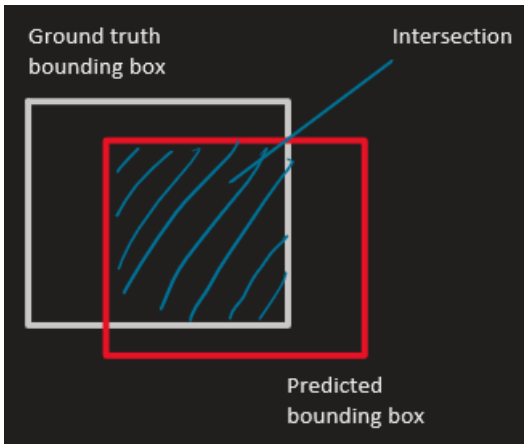


Figure 2.6: Intersection Over Union.

Since YOLO was released in 2015, many newer versions have been developed. The latest version is YOLOv8, and is developed by ultralytics. YOLOv8 is mostly an improvement of YOLOv5, and have been improved with higher performance, accuracy and bigger image sizes [39].

Chapter 3

Materials

3.1 Jetson Nano

Jetson Nano Developer Kit runs on Linux using the NVIDIA JetPack SDK and has built-in support for, among other things, USB, Ethernet, GPIO pins, and I2C. It is perfect for autonomous machines, robots, and embedded systems that need to do AI and high-performance computing in a small space [18]. The Jetson can be seen in figure 3.1.

Figure 3.1: Jetson nano Development Kit [19].

3.2 Thruster

The thrusters chosen are T200 thrusters with an operating voltage of 7–20 v and a maximum thrust of 6 kgf at 18 V. These thrusters are commonly used for underwater vehicles like ROVs and AUVs [5]. The thruster can be seen in figure 3.2.

Figure 3.2: T200 thruster [5].

3.3 Basic ESC

To control the thruster, an ESC is used as the thrusters are three phase brushless motors. The basic ESC is rated at 30 A [5]. The basic ESC can be seen in figure 3.3.

Figure 3.3: Basic ESC [5].

3.4 Autopilot

The Cube Orange flight controller produced by Ardupilot is an autopilot 3.4.

Figure 3.4: Ardupilots Cube Orange [2].

3.5 Batteries

The chosen batteries are four 18 V, 6 Ah Li-ion batteries to power the main current as seen in figure 3.6, and one 18 V, 5Ah Li-ion battery as seen in figure 3.5 to power the control current. There is a voltage converter used to power the Jetson.

Figure 3.5: M18™ 5.0 AH [15].

Figure 3.6: Battery 6,0 Ah LXT [12].

3.6 Camera

The ZED 2 camera as seen in figure 3.7 was provided. It is a stereo camera with an 8-element lens, a 120-degree field of view, optical correction, and integrated IMU, barometer, and magnetometer sensors [34].

Figure 3.7: ZED 2 camera [34].

3.7 WiFi-antenna

The TP-Link TL-WN722N USB WiFi adapter shown in figure 3.8, was used to use wireless internet connection on the Jetson Nano.

Figure 3.8: TP-Link TL-WN722N [11].

3.8 Phone with network sharing

A phone with network sharing capabilities was used to SSH into the Jetson Nano during field testing.

3.9 RC controller

The FrSky Taranis Q X7S shown in figure 3.9 is a 16 channnel RC controller suitable for use outdoors [7].

Figure 3.9: Taranis Q X7S[7].

3.10 Software and libraries

3.10.1 Software

Table 3.1: Table below contains software used.

Software	Version	Comment
Overleaf		Library for scientific computing that makes it easy to work with arrays and do math [22]
PCSchematic Automation	23	CAD software solution used for electrical design and documentation tasks [23].
Siemens NX	2206	3D CAD, manufacturing, and engineering software [30].
PrusaSlicer	2.5.1	Open-source slicing software that allows users to prepare 3D models for printing with a variety of settings and customizations [27].
OctoPrint	1.8.5	Free open source software used for controlling and monitoring 3D printers.It is compatible with most common 3D printers [21].
Mission planner	1.3.80	Open source ground station software for connecting to, and monitor flight controllers in real time using MavLink [3].
Visual Studio Code	1.77.3	Lightweight but powerful source code editor developed by Microsoft. It supports multiple programming languages and has a vast library of extensions available [14]
GitHub		Platform for hosting, storing and editing code using Git.
ZED-SDK	3.8	Enables the ZED2 to do onboard computing for various stereo camera applications, such as depth sensing and positional tracking [35].
ROS	Melodic	Software used to develop robot applications using libraries and tools [28].
NVIDIA JetPack	4.6	Used for building hardware-accelerateed AI applications on the NVIDIA Jetson modules. Contains Jetson Linux bootlader, Linux Kernel, Ubuntu dekstop and libraries for accelerating GPU computing [17].
Python	3.8.0	

3.10.2 Python libraries

Table 3.2: Python libraries used in the project.

Library	Version	Comment
Numpy		Library for scientific computing that makes it easy to work with arrays and do math [8].
OpenCV		Library of algorithms for computer vision and machine learning that can be used to process and analyze images and videos [6].
Matplotlib		Library for plotting and visualizing data that lets you make high quality graphs and figures in multiple different formats [9].
labelImg		Tool for labeling a image's bounding boxes [24].
ultralytics		Library for using ultralytics YOLOv8-models and functions [39].
datetime		Library for working with date and time [25].
rospy		Library for integrating ROS into Python [24].

Chapter 4

Method

4.1 Physical design

The vessel was designed in Simens NX prior to its physical construction. This was done in order to determine how much material was required and how each component would fit. Figure 4.1 displays a picture of the 3D model.

Figure 4.1: 3D model of vessel

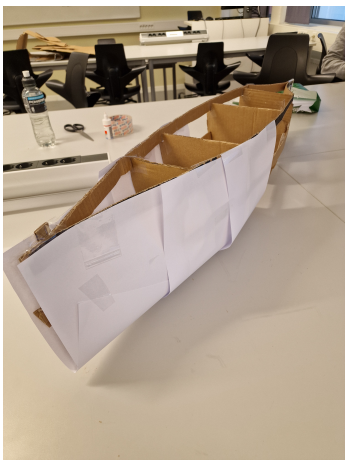
4.1.1 Hull

The solution of using a catamaran was chosen because it is most stable hull choice for the given dimension limitations. Additionally, it opened up the possibility of mounting a portion of the components within the hulls.

The hull's mold was constructed from strips of huntonit and plywood, as well as laser-cut plywood plate floors. Two 36-by-68-millimeter and two 11-by-36-millimeter pieces of wood were used to construct the keels. The hull was constructed from fiberglass and epoxy.

First, the mold was made in Siemens NX, see figure 4.2a so that the right measurements could be taken. Also, a cardboard model of one of the mold's earlier design iteration was made so that the exact size of the mold could be visualized. See figure 4.2 for a comparison between the cardboard model and the final design in Siemens NX. This allowed for any necessary adjustments to be made before production of the mold began.

(a) Last version of 3D modeled hull mold



(b) Cardboard prototype of hull mold

Figure 4.2: Prototyping of hull mold

Afterwards, the hull mold was created with the previously mentioned materials, the frame skeleton can be seen in 4.3a. Figure 4.3b depicts that the molds were additionally wrapped in plastic film. This was done in order to prevent epoxy from seeping through the molds and to make mold release easier.

(a) Mold frame

(b) Hull molds

Figure 4.3: Various stages of the hull mold.

4.1.2 Casting

Before casting was started, a risk assessment were done. The risk assessment in appendix A concluded that the group members were to use respiratory protection, safety goggles and disposable gloves as a minimum. Using proper work clothes was also recommended for own convenience.

First, a layer of mold release wax was applied to the hull. After a few minutes when the wax had dried, the casting was started. First epoxy was applied to the hull, then the fiberglass was laid over. Then another layer of epoxy was applied on the fiberglass. After the first layer of fiberglass was laid, the other layers were laid immediately using the same approach.

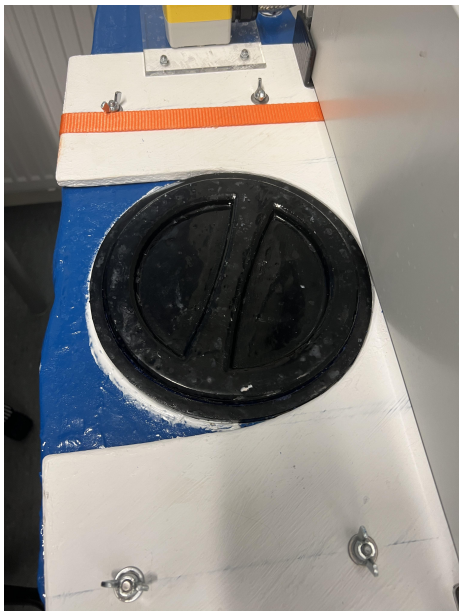
The hulls were cast in four layers of fiberglass. The inner layer was of $300g/m^2$ fiberglass mat, the next two layers was of $450g/m^2$ fiberglass mat, and the last layer was of $150g/m^2$ fiberglass cloth to get a smooth surface. To make the casting easier, the fiberglass was cut in pieces beforehand to fit the hull.

After the hulls were cast and dried they were cut flush at the top. A floorplate of plywood, roughly 350x200mm was then fastened to the inside using spackle to allow for mounting of components inside the hulls. Lids were made from plywood, and cast with two layers of fiberglass cloth. A circular hole with a diameter of 21cm was made for future access to the inside of the hulls.

When the hulls were properly hardened they were covered in 4 layers of epoxy primer, as a way to avoid osmosis of the outer layers and further ensure waterproofing.

4.1.3 Lids

To access the inside of the hulls there was printed circular threaded lids in PETG, as well as threaded bases. The parts were coated in two layers of resin, except the threads, and a layer of silicone was applied to the contact points to act as a gasket. The bases were fastened to the hulls using marine silicone. A lid can be seen in figure 4.4a and 4.4b.



(a) The lid when tightened

(b) The lid when open

Figure 4.4: The lids closed and opened

4.1.4 Thruster mounting

There was drilled holes in the bottom of the hull for cable glands. Wetlink Penetrator glands were used, which is rated waterproof to 950m depth. Thruster mounts were 3D printed in PETG and fastened to the bottom of the hulls using M4 bolts, the drilled holes were sealed using marine sealant specifically meant for sealing under the waterline. After assembling and testing the hulls to be waterproof they were painted using a blue two-component spray paint.

4.1.5 Connector plate

A plate was cut out in MDF to connect the hulls, the plate was covered in two layers of epoxy primer to avoid absorbing moisture when exposed to the elements.

4.2 Electrical design

The electrical build was designed in order to adhere to NEK 410:2021[13]. The ASD should be able to handle the full current draw of the thrusters over an extended time, minimize EMC, and parts being submerged for a short amount of time in case of accidental flips of the vessel.

Full electrilcal diagrams of the circuitry can be found in the wiki A.

4.2.1 Control cabinet

The vessel was fitted with a 500x400mm cabinet to house the control circuitry. The cabinet is made of poly-carbonate, and has an IP grade of 66 ensuring that it can handle the salty and wet environment that the vessel will be deployed in.

The cabinet which can be seen in figure 4.5 was fitted with 3 DIN rails, 4 cable gates, and terminals as well as a power supply, Jetson Nano, Orange Cube, and a battery mount.

The wiring for the control cabinet was done with multi-threaded wire, mainly $1.5mm^2$ in the colors red(+) and blue(-). This diameter fulfills the demands made in NEK410[13] table 11 and 12 for the expected current draws.

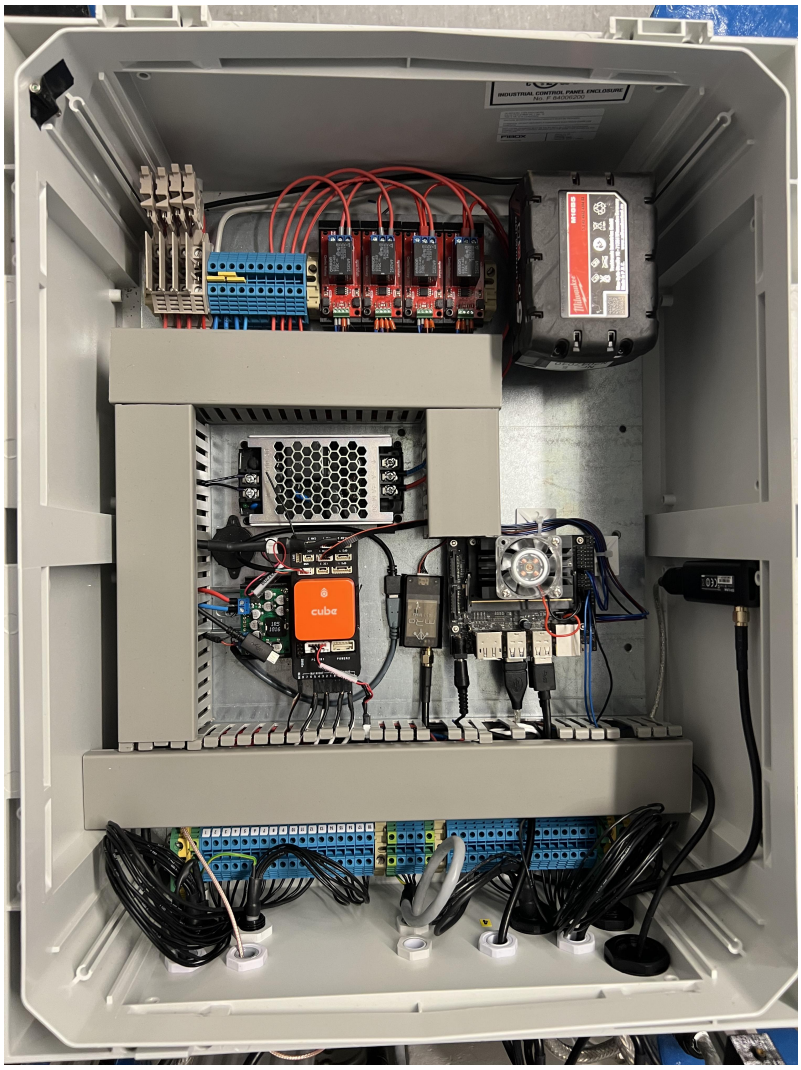


Figure 4.5: The control cabinet of the vessel

4.2.1.1 Control circuitry

The control circuitry for the vessel consists of a Orange Cube which handles everything related to steering, missions and GPS, and a Jetson Nano which does the image processing and also handles relays. The Cube and the Nano is connected through USB serial communication.

The Jetson is connected to four relays using I^2C communication from the J41 pins, one handles remote actuation of the emergency stop, and three handles the lights that indicates which mode the vessel is in. Through it's USB ports it is connected to a WIFI unit, and the camera.

The Orange cube is connected to three antennas mounted on an external rack, Here3+ GPS, telemetry and RC. It is also connected to four ESCs and two power modules which are located in the hulls.

4.2.2 Outside components

Some components were mounted on the outside of the control cabinet. All wiring going out of the cabinet, except to the lantern, is shielded to minimize EMC disturbance of the telemetry and RC.

On the port side there was mounted a lantern with three colors, red, yellow and green. On the starboard side there was mounted an emergency stop button with twist release. Both of these components are requirements given by the Autodrone competition.

The back of the vessel was fitted with an antenna rack as seen in figure 4.6 where three RP-SMA antennas and the RC antenna was placed. The top of the rack holds the Here3+ GPS antenna and the ZED2 camera. This rack has two purposes, to get a higher vantage point for the vision, and mounting antennas outside of the cabinet to maximize signal range.

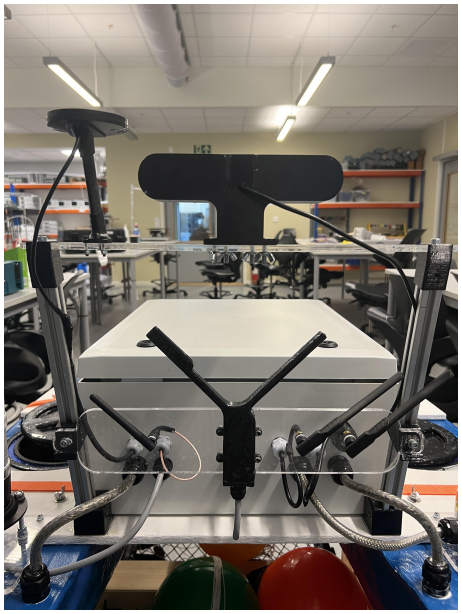


Figure 4.6: The antenna rack seen from behind

4.2.3 Main power cabinets

To prevent issues with EMC that was observed in the previous bachelor groups vessel [32], the main power to the thrusters were separated from the control circuitry. This had the added benefits of shortening the wires feeding the thrusters, as well as lowering the weight point of the vessel due to placing heavy components in the hulls.

In each hull there is mounted two Makita drill batteries coupled in parallel through a terminal, which then feeds into a junction box. Inside the box the power goes through a power module and is then divided between two 25A circuit breakers, solid state relays, and ESCs controlling the thrusters. The Junction box cna be seen in 4.7

Figure 4.7: The junction box inside the hull (before mounting relays)

the choice of dimensioning the circuit breakers to 25A were done after reviewing charts given by the manufacturer, in figure 4.8 and figure 4.9 it is shown that the maximum current draw at 18V is 27A which the circuit breakers with caracteristic C can maintain for minimum one hour, however using 100% thrust for such an extended time is not realistic for the planned use and is not power efficient in any way.

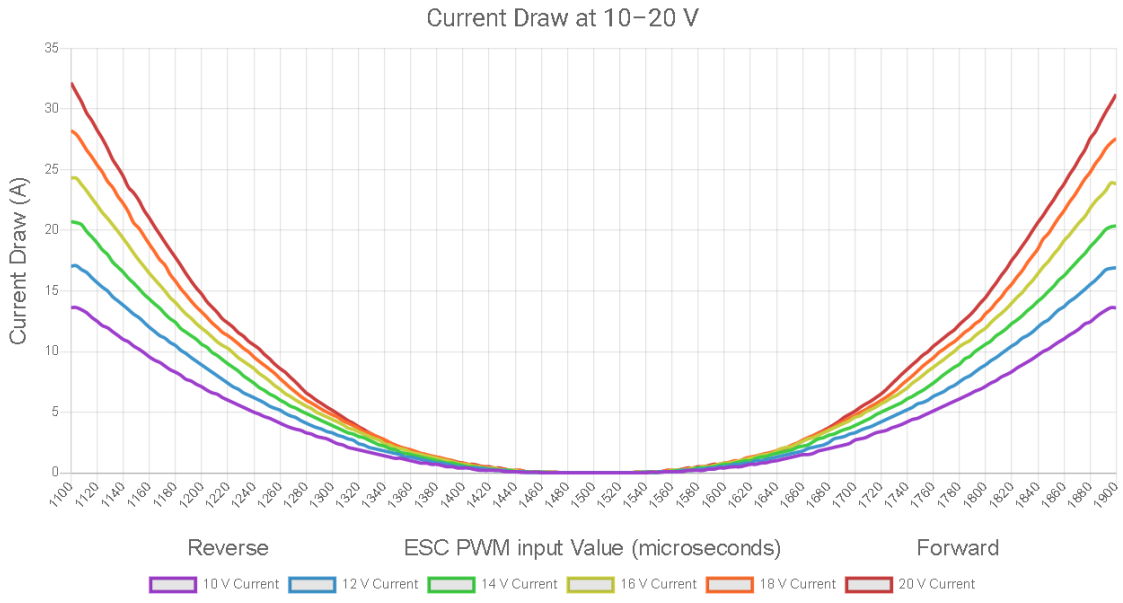


Figure 4.8: Current draw at given PWM [5]

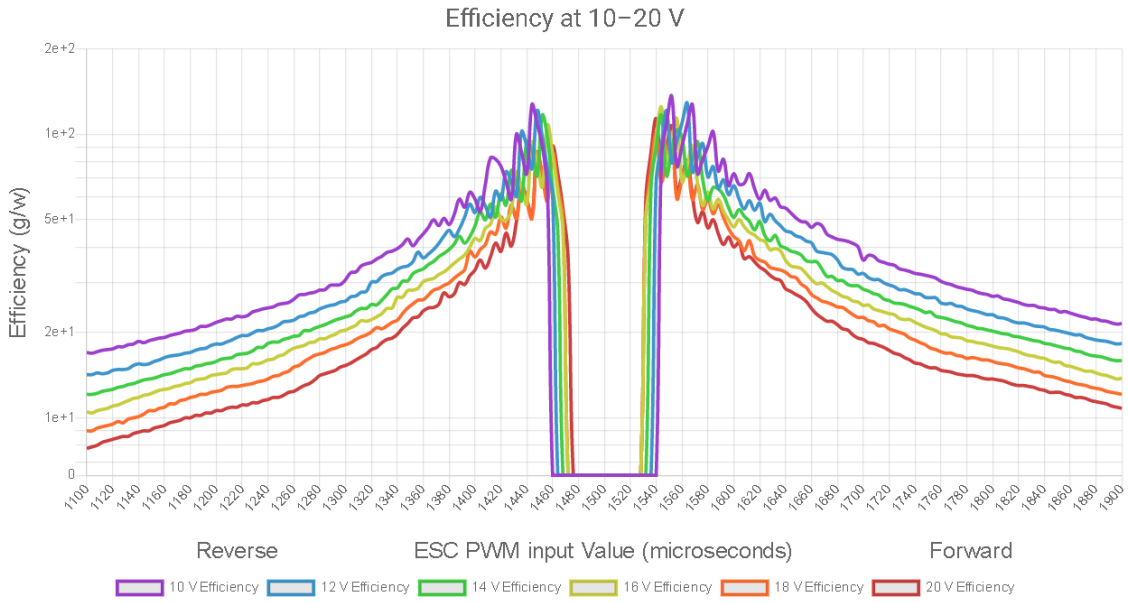


Figure 4.9: Efficiency at given PWM [5]

4.2.4 Ground station

The ground station setup consists of a laptop running Mission planner. Plugged in is the telemetry antenna for the Cube, and a Here+ RTK base mounted on a tall pole. Additionally we use the FRSKY X7S as RC control.

4.3 Software

All source codes can be found in the group's GitHub, see appendix B.

4.3.1 Ardupilot

The Orange Cube is responsible for handling and GPS localizing of the vessel, for this it uses the navigation firmware Ardupilot which we interface using the ground station software Mission Planner.

After installing the necessary firmware to the Cube, the FRAME_CLASS is set to "Boat" and the FRAME_TYPE is set to "OmniX" indicating what type of vessel and thruster configuration is mounted. Setting frametype as boat enables thruster steering, changes the icon on the map, and indicates that the vessel should maintain its position when in autonomous modes so it does not drift away.

4.3.1.1 Modes

Ardupilot includes multiple control modes, in the scope of this thesis mainly four were used. Manual, Auto and Loiter.

Manual: The vessel is controlled via RC. The mode allows for complete thruster control, meaning lateral movement is possible. Using an additional switch also allows for saving the current position as a waypoint, meaning that one can map a course in manual mode, then make the vessel drive the same course in auto. The manual mode was used for thruster and handling tests, as well as general placement relative to speed gates.

Guided: The vessel recieves a waypoint by telemetry from the groundstation, it drives to the given point.

Auto: The vessel drives a preprogrammed course using multiple waypoints, it can be set to return to launch position upon completion. The auto mode is used by the vision control. During preliminary testing it was used for testing approach angles, turning radius, waypoint radiuses and more.

Loiter: The vessel holds the current position indefinitely. This mode was used extensively during testing to keep the vessel in place when attention needed to be elsewhere.

4.3.2 ROS

ROS is run on the Jetson Nano and the two main packages used are the *zed_wrapper* and *mavros*. The zed_wrapper was used to retrieve the left camera image and depth map from the ZED2 camera. When the image and depth map is retrieved, it has to be processed to be able to use it in python. Using the numpy functions *frombuffer* and and *reshape*, the image and depth map is converted to a Numpy array. The image's alpha channel had to be removed to be able to pass it through the YOLOv8-model, which was done by using list slicing. The depth map is just a grayscale image so it had to have all it's channels sliced. Additionally the image was normalized using OpenCV's function *normalize*, to get pixel values between $0 - 255$. The image and depth map callback is shown in figure 4.10.

```
def image_callback(self, msg):
    img_left = np.frombuffer(msg.data, dtype=np.uint8).reshape(msg.height, msg.width, -1)
    img_left = img_left[:,:,:3]
    self.img_array = cv.normalize(img_left, None, 0, 255, cv.NORM_MINMAX, cv.CV_8U)

def depth_callback(self, msg):
    depth_map = np.frombuffer(msg.data, dtype=np.float32).reshape(msg.height, msg.width, -1)
    self.depth_img = depth_map[:,:, 0]
```

Figure 4.10: Image and depth map callbacks.

The mavros package is used to communicate with the autopilot over USB serial communication using the MAVLink protocol. Mavros is launched using the *apm* launch file, which had to be configured for USB serial communication. Editing the *fcu_url* default argument to /dev/ttyACM0:921600 configured mavros to communicate using a USB port with a baudrate of 921600.

Table 4.1 shows the subscribed/published mavros topics used in the project.

Table 4.1: ROS topics used in the project.

Topic	Type	Comment
/mavros/setpoint_velocity/cmd_vel_unstamped	Publisher	Used to publish linear or angular velocity, to drive without waypoints.
/mavros/setpoint_raw/global	Publisher	Used to publish waypoints used for GUIDED mode.
/mavros/state	Subscriber	Used for during initialization to display vessel status.
/mavros/global_position/global	Subscriber	Used to get vessel's GPS coordinates in degrees decimal form.
/mavros/global_position/compass_hdg	Subscriber	Used to get vessel's heading in degrees.
/mavros/mission/reached	Subscriber	Used to get information when waypoint is reached.
/mavros/mission/waypoints	Subscriber	Used to get vessel's current waypoint sequence.

Table 4.2 shows the mavros services used in the project.

Table 4.2: ROS services used in the project.

Topic	Type	Comment
/mavros/cmd/arming	Service	Used to arm or disarm the vessel's motors.
/mavros/set_mode	Service	Used to change the vessel's autopilot mode.
/mavros/mission/push	Service	Used to push the vessel's waypoints used in AUTO mode.
/mavros/mission/clear	Service	Used to clear the vessel's waypoints used in AUTO mode.

4.3.3 Machine vision

Machine vision was used as the main source of navigation by recognizing the buoys by their shape and color. The vessel is only concerned about the two closest buoys when setting waypoints. This is to filter out all other buoys which may be further away and not immediately important. The eight version of the YOLO-algorithm by Ultalytics was used to train an object detection model for the buoys.

Some changes were made to the ZED2-camera settings to make the camera more computational efficient. These setting files are found in the zed_wrapper ROS package. The depth quality was increased for better depth readings, but this increases the computational resources needed. To compensate for this the minimum depth was increased and the publish rate was reduced, as a measure to reduce the computational resources needed. Table 4.3 shows the changes done to the camera settings in the zed_wrapper package.

Table 4.3: Changed camera settings.

Setting	File	Comment
min_depth	zed2.yaml	Raised to $1.0m$ from $0.3m$, to reduce computational resources needed.
pub_frame_rate	common.yaml	Reduced to 5 from 15 to reduce computational resources needed.
depth quality	common.yaml	Changed to ULTRA from PERFORMANCE for better depth results.

4.3.3.1 Training a YOLOv8-model

To train a YOLOv8-model a large data set of bouy images was gathered. The buoy image set consists of a large variety of different angles, light conditions and backgrounds to make the model as robust as possible. Additionally the images were captured with different cameras to get a variety of image resolutions and lighting. The data set was captured by attaching a line to the buoys and throwing them in the ocean from land, as the group did not have access to a boat.

After the data set was acquired, the images had to be labeled before it could be used for training. The labelImg Python GUI was used to label each image with the buoy location and color. Before labeling the save format had to be changed to YOLO in the labelImg GUI. When the whole data set was labeled the image folder and coherent label folder was put in a *Training* folder. Some of the training images and their labels were transferred to a *Validation* folder. Then the YOLOv8-model was trained over 100 epochs using the command line, as described in the YOLOv8 documentation by Ultralytics [39]. After training is complete, the last model and the best model will be saved. The best model was extracted and used in the code.

4.3.3.2 Converting detections to GPS coordinates

The detection results were processed to extract the buoy type, detection depth and the detection bearing. Detection depth was retrieved from the ZED ROS package using the detection center. To find the detection bearing, equations 4.1 and 4.2 were used [16].

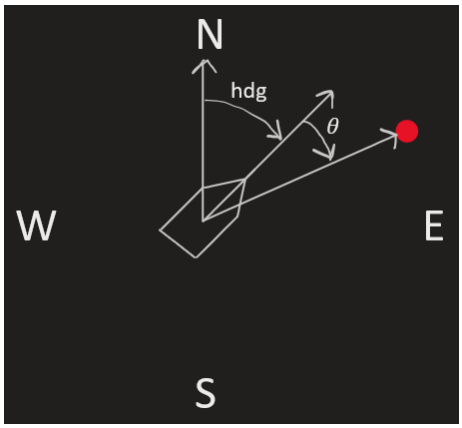
$$\lambda = \frac{hfov}{width} \tag{4.1}$$

$$\theta = T \cdot \lambda \tag{4.2}$$

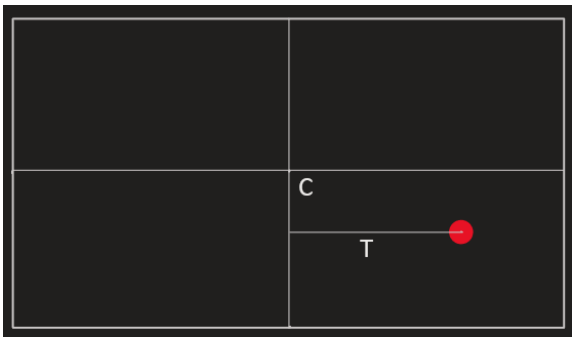
Where:

- θ - Bearing.
- **T** - Translation from image center.
- λ - Degrees per pixel.
- **hfov** - Horizontal field of view.
- **width** - Width of image.

Figure 4.11a shows the bearing angle θ, and figure 4.11b show the translation relative to the image center.



(a) Bearing.



(b) Translation in the image.

Figure 4.11: Bearing and translation illustrated.

When distance and bearing is found, the buoy's GPS location can be found using the vessel's GPS location and heading [10]. First the bearing with respect to north is found using equation 4.3.

$$\theta_N = hdg + \theta \tag{4.3}$$

Using equation 4.4 to find the buoy's latitude.

$$lat = \arcsin\left(\sin lat_v \cdot \cos\left(\frac{dist}{R}\right) + \cos lat_v \cdot \sin\left(\frac{dist}{R}\right) \cdot \cos\theta_N\right) \tag{4.4}$$

And equation 4.5 to find the buoy's latitude.

$$lon = lon_v + \arctan 2\left(\sin\theta_N \cdot \sin\left(\frac{dist}{R}\right) \cdot \cos lat_v,\ \cos\left(\frac{dist}{R}\right) - \sin lat_v \cdot lat\right) \tag{4.5}$$

Where:

- lat_v - Vessel latitude.
- lon_v - Vessel longitude.
- θ_N - Buoy bearing with respect to north.
- $dist$ - Distance to buoy.
- hdg - The vessel's heading.
- R - 6371e3, Earth's radius in meters.

These principles will also be used for converting distance and bearing to GPS coordinates for other uses later.

4.3.4 Missions

During the missions the vessel navigates by GPS locations, and by simple path planning as will be described in each mission section. Whenever the vessel sets one waypoint at a time, the autopilot will be in *Guided* mode. If the vessel sets two or more waypoints at a time, *Auto* mode will be utilized.

4.3.4.1 Navigation channel

Figure 4.12 shows the flow chart for the navigation channel mission.

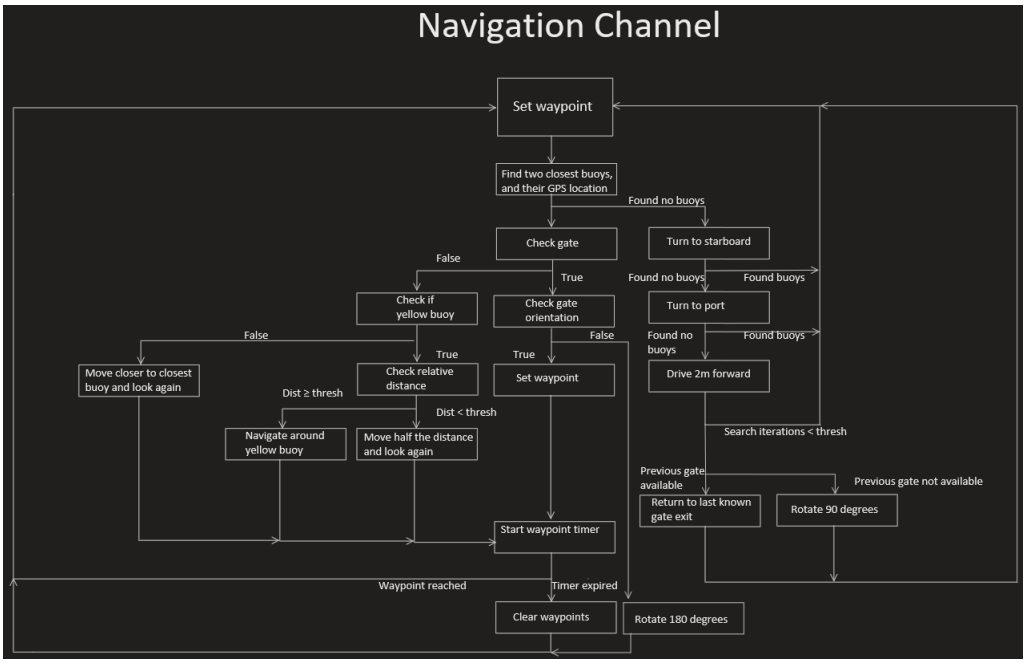


Figure 4.12: Navigation channel flow chart.

During the navigation channel mission, the vessel will look for a buoy gate which consists of a red and green buoy. If a gate is found it will also check the gate's orientation, as a measure to prevent

it from going backwards in the course. When a gate is found, the waypoint is set in the middle of the gate. Another waypoint is set one meter out of the gate and perpendicular with respect to the gate's bearing, to get the vessel clear of the buoys. Figure 4.13 illustrates how the waypoints are set.

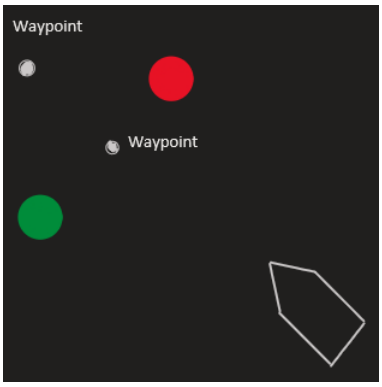


Figure 4.13: How the gate waypoints were set.

Equation 4.6 shows how the gate waypoint is calculated.

$$wp_{lat} = \frac{buoy1_{lat} + buoy2_{lat}}{2}, \; wp_{lon} = \frac{buoy1_{lon} + buoy2_{lon}}{2} \tag{4.6}$$

To set the waypoint one meter perpendicular out of the gate, the buoys bearing had to be calculated as shown in equations 4.7, 4.8 and 4.9, [10].

$$x = \sin\left(buoy2_{lon} - buoy1_{lon}\right) \cdot \cos buoy2_{lat} \tag{4.7}$$

$$y = \cos buoy1_{lat} \cdot \sin buoy2_{lat} - \sin buoy1_{lat} \cdot \cos buoy2_{lat} \cdot \cos\left(buoy2_{lon} - buoy1_{lon}\right) \tag{4.8}$$

$$\theta = \arctan 2(x, y) \tag{4.9}$$

When the buoy gate bearing is found, the vessel's heading out of the gate can be found. This was done by adding or subtracting 90° to the buoys bearing, depending on the buoys bearing and the vessel's heading. Figure 4.14 shows how this was implemented in code.

```
@staticmethod
def hdg_rel_to_bearing(_bearing, rel_angle, drone_hdg): # _bearing needs to be in range [-180, 180]
    out_heading = None
    if (_bearing >= 0 and _bearing < 90) or (_bearing < 0 and _bearing >= -90):
        if (drone_hdg >= 270 and drone_hdg < 360) or (drone_hdg >= 0 and drone_hdg < 90):
            out_heading = _bearing - rel_angle
        else:
            out_heading = _bearing + rel_angle
    elif (_bearing >= 90 and _bearing <= 180) or (_bearing < -90 and _bearing >= -180):
        if drone_hdg >= 90 and drone_hdg < 270:
            out_heading = _bearing + rel_angle
        else:
            out_heading = _bearing - rel_angle
    return out_heading
```

Figure 4.14: How the out of gate heading was determined.

Where in this case:

- **_bearing** - The buoy gate's bearing.

- **rel_angle** - The 90° relative angle to the buoy gate bearing.
- **out_heading** - The heading the vessel should have when leaving the gate.

The out of gate heading is also helpful for the vessel to look for the next buoys after exiting gate. If the vessel were to exit the gate at an angle it might not see the next buoys.

When the vessel's out of gate heading is determined, the waypoint were set using the same principles for determining the buoys GPS location as shown earlier in equations 4.3, 4.4, 4.5. However the buoys distance and bearing were substituted for distance- and heading- out of gate.

If a gate is not found, the vessel will see if the closest buoy is yellow. However, the camera have been struggling differentiating yellow buoy from green and red buoys in sunny conditions. As a counter measure to avoid the vessel navigating around a false positive detection of yellow buoy, the relative distance between the closest and second closest buoy is calculated using the law of cosines as shown in equation 4.10.

$$Relative\ distance = \sqrt{a^2 + c^2 - 2ac \cdot \cos\beta} \tag{4.10}$$

Where:

- a - Distance between vessel and closest buoy.
- c - Distance between vessel and second closest buoy.
- β - Angle between buoys as seen from the vessel.

If the relative distance is lower than the set threshold, the vessel will move half the distance to get a closer look. If the relative distance is greater than the threshold, or if only a yellow buoy is detected, the vessel will navigate around the buoy. Setting the waypoint around the yellow buoy was done using Pythagoras sentence as shown in figure 4.15.

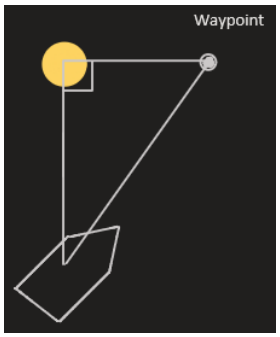


Figure 4.15: How the waypoint around a yellow buoy was determined.

Equations 4.11 and 4.12 shows how the waypoint's distance and bearing is calculated.

$$\text{Distance to waypoint} = \sqrt{a^2 + b^2} \tag{4.11}$$

$$\theta = x \cdot \arctan 2(b, a), \quad x = \begin{cases} -1, & \text{if buoy bearing} < 0 \\ 1, & \text{if buoy bearing} \geq 0 \end{cases} \tag{4.12}$$

Where:

- a - Distance between vessel and buoy.

- b - Distance between buoy and waypoint.
- θ - Angle between buoy and waypoint as seen from vessel.

The vessel will determine if it should go around the yellow buoy on the left or right side by looking at the buoy's bearing. If the buoy's bearing is less then zero, the vessel will go around on the left side and vice versa. Then the equations 4.3, 4.4, 4.5 is used to convert the waypoints distance and bearing to GPS coordinates.

If only one green or red buoy is detected, the vessel will set the waypoint directly on the buoy and move half the distance to get a closer look.

If no buoys is detected, the vessel will start searching for buoys. First it will rotate a set amount of degrees to starboard and look, then rotate to port and look, and then drive a few meters forward and look. The vessel will repeat this process a set amount of times or until it finds a buoy. If no buoys is found after the vessel have searched for the set amount of times, it will return to the last gate exit. However if the vessel have not passed a gate, it will consider it self lost and stay in one place while rotating around itself to look for buoys.

4.3.4.2 Speed gate

Figure 4.12 shows the flow chart for the speed gate mission.

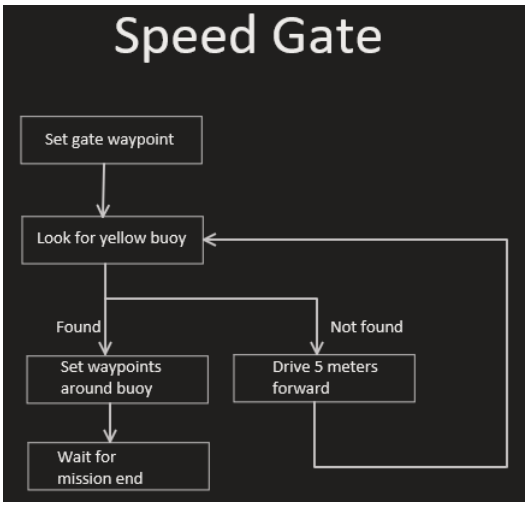


Figure 4.16: Flow chart for the speed gate mission.

The vessel will start the mission by setting the gate waypoint, and a waypoint one meter forward as explained in the navigation channel section. Additionally a waypoint is set one meter backwards which will be used when returning to the gate to make sure the whole vessel passes the finish line. The backwards waypoint is calculated as shown in equations 4.13 and 4.14.

$$\text{backwards } wp_{lat} = 2 \cdot \text{gate } wp_{lat} - \text{forward } wp_{lat} \tag{4.13}$$

$$\text{backwards } wp_{lon} = 2 \cdot \text{gate } wp_{lon} - \text{forward } wp_{lon} \tag{4.14}$$

After the vessel has passed the gate, it will start looking for the yellow buoy. When the yellow buoy is detected it will set three waypoints in a triangle around it as shown in figure 4.17.

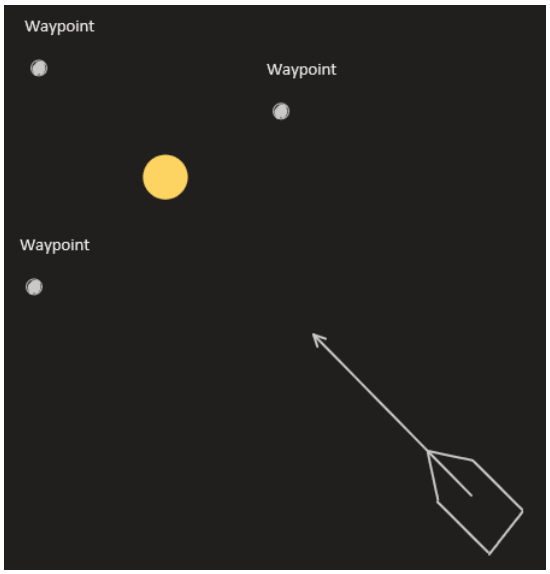


Figure 4.17: How the waypoints is set around the yellow buoy.

The waypoints is calculated using the same principles as in the navigation channel mission, by using equations 4.11 and 4.12.

After the yellow buoy is rounded, the vessel will return to the starting gate and pass the gate by one meter to make sure the whole vessel passes the finish line.

4.3.4.3 Docking

Figure 4.18 shows the preliminary flow chart for the docking mission.

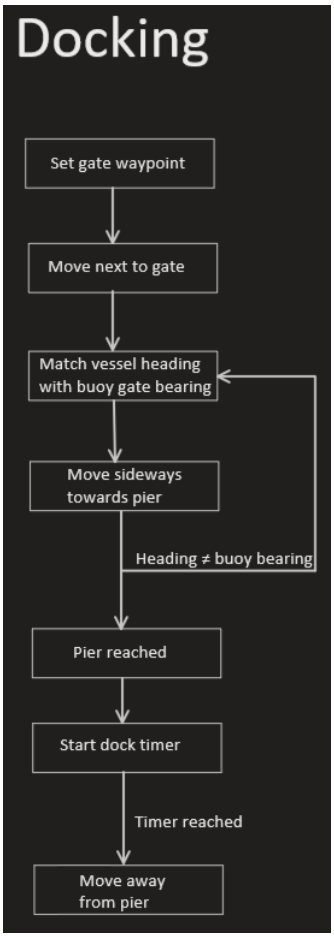


Figure 4.18: Flow chart for the speed gate mission.

The vessel will start by finding the buoy gate, and calculate it's bearing using equations 4.7, 4.8 and 4.9. The buoy gate bearing will be used to match the vessel's heading, so it can line up with the pier. When the vessel has reached the pier a timer of 30 seconds will be started, according to the competition specification. After the timer is reached, the vessel will leave the pier and the mission is complete.

4.4 Testing

4.4.1 Pull force

The pull force tests were conducted in a controlled environment at NTNU in Ålesund's indoor water tank. The pull force was measured in newtons using a digital hanging scale accurate to within 0.1 kilograms. More information about the weight can be found in the wiki in appendix A. The test was conducted with the vessel stationary, and a rope attached to a wire on the stern of the vessel was used to secure the digital scale as seen in figure 4.19.

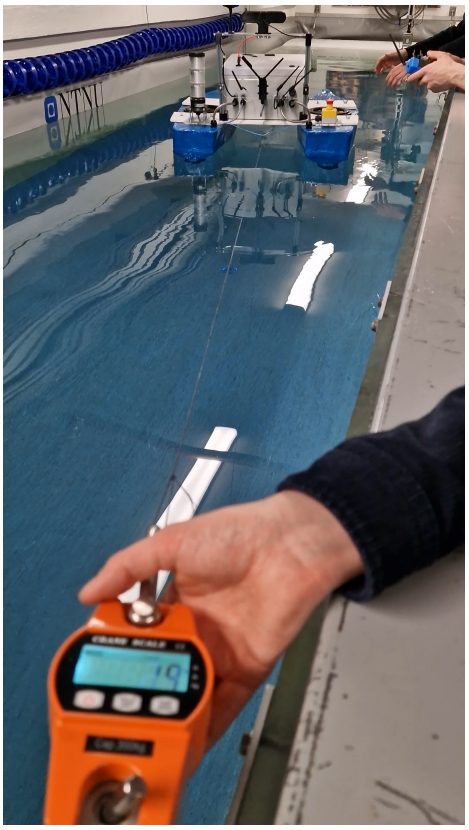


Figure 4.19: Test of pull force in indoor water tank.

4.4.2 Speed test

The speed was measured by the floating pier by Aalesunds Roklub on the 27th of April 2023 from 18:15 to 19:10. With a temperature ranging from 6.2°C to 4°C and a northwest wind of 3m/s [33]. The top speed was measured using the mission planer.

4.4.3 Mission testing

The missions were tested at the pier by Den Norske Turistforeningen Naustet. As the group did not have a boat at disposal, the buoys were tied to a rope and thrown out in the sea from the pier. Some weight was tied to the ropes to prevent the buoys from drifting too far away. Mobile network was shared with the Jetson Nano and the laptop to SSH into the Jetson from the pier. However the Jetson struggled to automatically reconnect to the network whenever it lost network connection. Therefore the phone used for sharing the network was placed inside the control cabinet, as the laptop did not experience the same reconnecting issues.

4.4.4 GPS accuracy test

One of the problems encountered by the last bachelor group was a lack of accuracy in their GPS waypoints [32], to remedy this problem an RTK GPS was used, and a comparison test was done.

The GPS tests were done by the DNT Naust on the 14th of May 2023, between 14:00-15:15 the weather was roughly 18°C and overcast with a 2m/s SW wind [33]. To test the accuracy of the GPS the vessel was deployed and a ”Home” position was set approximately one meter from the pier, since the pier is stationary it allows for using a meter stick to measure drift from the start

position. The tests can roughly be divided into two categories, attempting to hold a position, and attempting to return to a position from another.

1. The vessel was set to loiter and the natural drift due to stream and wind was measured after one minute.
2. The vessel was softly pushed until it attempted to thrust against the force.
3. The vessel was sent to a waypoint 4 meters away and asked to return to a waypoint at home position.

The tests were done while varying the WP_RADIUS parameter which controls when the GPS considers a waypoint reached while driving. If this values are too high it leads to inaccuracy since the vessel considers the waypoint as a large area, if it is too small it might not be able to consider the point reached leading to backtracking, or unnecessary turns while driving. As missing a waypoint would impede performance in the competition test 3 was considered failed if the vessel missed the waypoint on return.

Chapter 5

Results

5.1 Physical dimensions

The boat is 890mm wide, 1110mm long and 880mm tall

5.2 Weight

The total weight of the boat is 36.6kg with an accuracy of 0.5gram.

5.3 Pull force

The expected result was derived from equation 2.10 where $K_1 = 6.02, \alpha = \frac{\pi}{4}$ and gave the expected result to be 17.03 kgf or 167.01N.

The result for the test can be seen in table 5.1

Table 5.1: Measured with an accuracy of 0.98N.

Thruster force	
Trial	Maximum force [N]
1	118.6
2	143.1
3	158.8

5.4 Speed test

The top speed was measured at 1.65m/s.

5.5 Depth to keel

The depth to keel is about 15cm on the SB side and about 15.5cm on the port side.

5.6 ROS

At first the group attempted to establish communication between the Jetson Nano and Cube Orange using the UART protocol with the mavros package. However the Jetson was only able to receive messages from the Cube, but could not send any messages to the Cube. Communication was successfully established between the Jetson and Cube using USB serial connection. Utilizing the mavros package, messages was sent between the Jetson and Cube using the MAVLink protocol.

5.7 YOLO training

YOLOv8 models are categorized depending on the models size. The smallest model is a nano-model, then second smallest is the small-model and then medium-model, large-model and XL-model [40]. For this project a nano-model, small-model and a medium-model was trained.

The training data set had a total of 259 images, while the validation data set had a total of 53 images.

5.7.1 Nano-model

Figure 5.1 and 5.2 shows the confusion matrix and precision-recall curve for the nano-model.

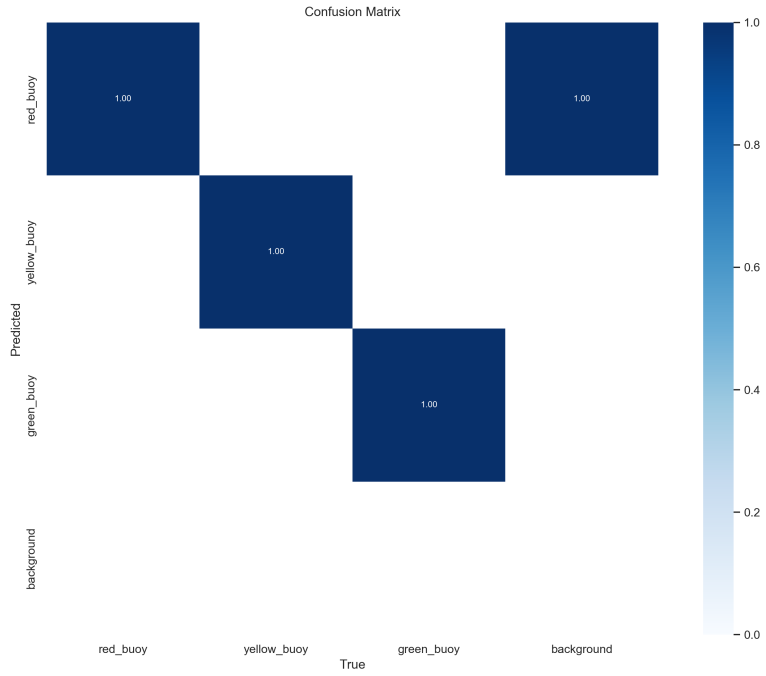


Figure 5.1: Confusion matrix nano-model.

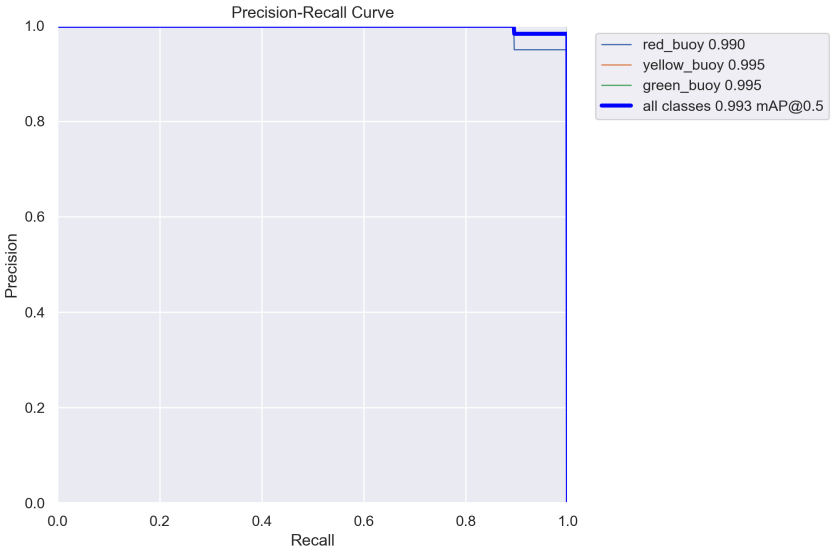


Figure 5.2: Precision-recall curve for nano-model.

The nano-model's mAP@0.5 is 0.993 as seen in figure 5.2.

5.7.2 Small-model

Figure 5.3 and 5.4 shows the confusion matrix and precision-recall curve for the small-model.

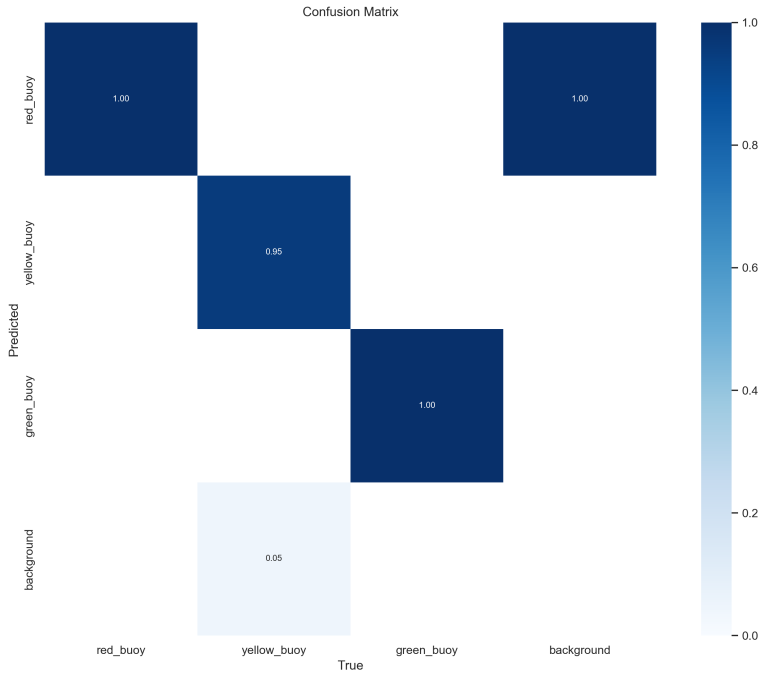


Figure 5.3: Confusion matrix small-model.

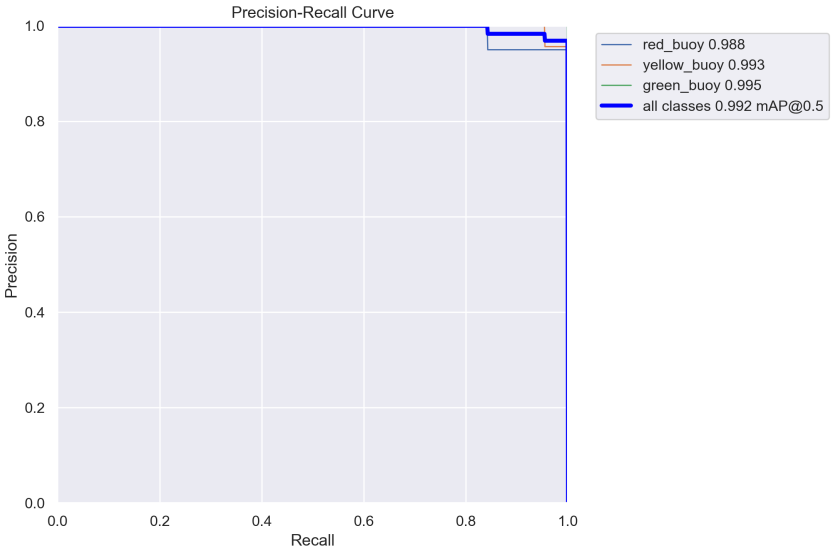


Figure 5.4: Precision-recall curve for small-model.

The small-model's mAP@0.5 is 0.992 as seen in figure 5.4.

5.7.3 Medium-model

Figure 5.5 and 5.6 shows the confusion matrix and precision-recall curve for the medium-model.

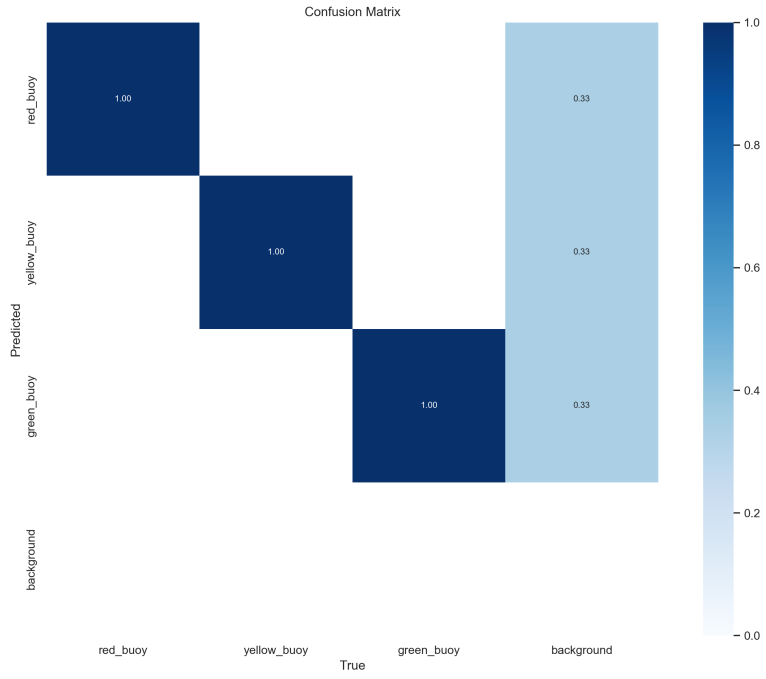


Figure 5.5: Confusion matrix medium-model.

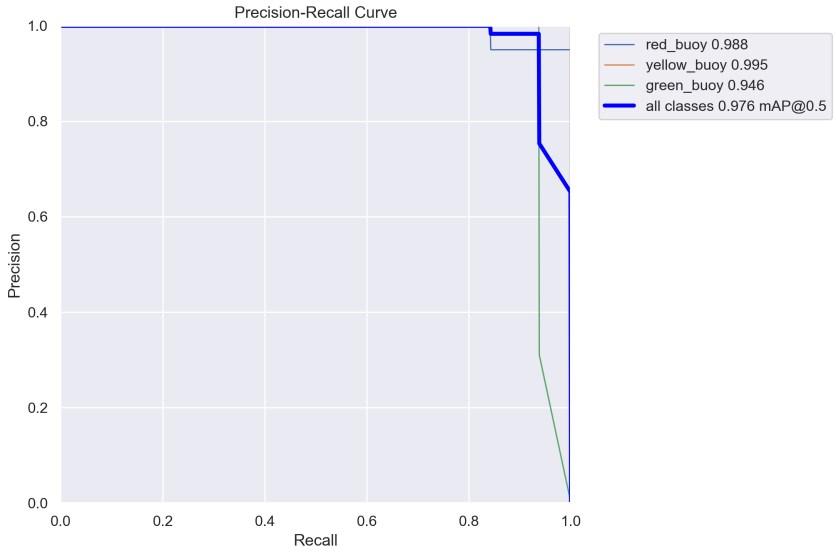


Figure 5.6: Precision-recall curve for medium-model.

The medium-model's mAP@0.5 is 0.976 as seen in figure 5.6.

5.7.4 Inference time

Table 5.2 shows the inference time of the different YOLO-models. The models inference times were tested with different image resolutions on the Jetson Nano.

Table 5.2: Inference times on the Jetson Nano.

Model	Resolution	Inference time
Nano	HD2K	$200 - 300$ ms
Small	HD2K	$450 - 550$ ms
Medium	HD2K	$1000 - 1200$ ms
Nano	HD1080	$200 - 300$ ms
Small	HD1080	$450 - 550$ ms
Medium	HD1080	$1000 - 1200$ ms

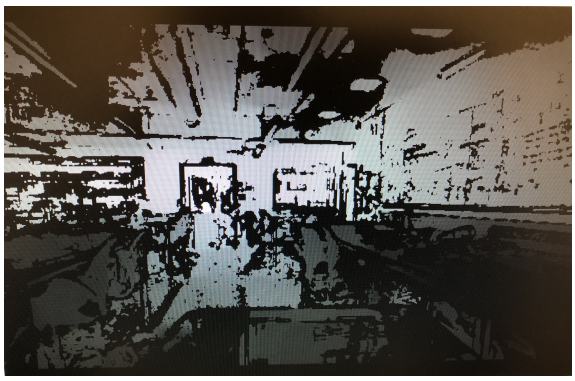
5.8 Machine vision

The vessel was able to successfully detect the buoys. From field testing the nano-model was able to detect buoys up to approximately 5-7 meters, the small up to approximately 8-10 meters and the medium-model up to approximately 10-12 meters. The accuracy decreased with range, and all models had trouble distinguishing red and green from yellow on long range detection. When the camera got closer, the models managed to detect correctly. This was also dependent on the lighting conditions, as in sunny conditions this issue was more pronounced than in cloudy weather.

The ZED-camera often returned the depth as nan. According to the documentation [35] this means the depth measure is an occlusion value, and that the depth cannot be determined. To debug this issue the ROS visualization program rviz was used. At first the depth mode setting was set to *performance*, which is shown in figure 5.7a. All the black spots in the image are the nan values. Setting the depth mode to *quality* as shown in figure 5.7b did not reduce the nan values either.

Setting the depth mode to *ultra* improved this issue substantially as shown in figure 5.7c. Field testing proved that the camera was still struggling determining the buoy's depth value. Setting the depth mode to *neural* as shown in figure 5.7d removed almost all nan values.

(a) Depth mode performance.



(b) Depth mode quality.

(c) Depth mode ultra.

(d) Depth mode neural.

Figure 5.7: The different depth modes available for the ZED2 camera.

Whenever the camera managed to determine the buoy's depth, then the rest of the machine vision part worked as expected. The detections were successfully converted to GPS locations, and waypoints were set correctly. The buoy gate's were detected and their orientation was checked.

5.9 Navigation channel

Field testing the vessel in the navigation channel mission proved it was able to navigate accordingly to the detected buoys. However the searching for buoys function was a bit clumsy, as the vessel would rotate further than the set 45°.

5.10 Speed gate

Field testing the speed gate mission proved that the vessel is able to set the waypoints around the yellow buoy as expected, and return to the starting gate.

5.11 GPS accuracy

Table 5.3: Total displacement from start position during GPS testing (m)

WP_Radius	Standing still	External force	Return to point
1	0.2, 0.5, 0.4	0.3, 0.6, 0.7	1.5, 2, 1.9
0.5	0.4, 0.4, 0.3	0.6, 0.5, 0.2	0.5, 1.9, 2.1
0.2	0.3, 0.3, 0.5	0.2, 0.3, 0.5	1.1, 2.3, 0.9
0	0.3, 0.3, 0.4	0.4, 0.5 0.2	(Missed waypoint), 0.5, (Missed waypoint)

5.11.1 RTK GPS accuracy

Table 5.4: Total displacement from start position during RTK testing (m)

WP_Radius	Standing still	External force	Return to point
1	0.4, 0.2, 0.3	0., 0.1, 0.1	1, 0.9, 1.2
0.5	0.5, 0.4, 0.3	0.4, 0.6, 0.2	0.7, 0.5, 1.
0.2	0.2, 0.1, 0.2	0.2, 0.3, 0.2.	0.3, 0.2, 0.5
0	0.2, 0.1, 0.1	0.1, 0.1 0.2	0.4, 0, (Missed waypoint)

Chapter 6

Discussion

6.1 Hull design

The hulls were cast using a total of four layers fiberglass, which might have been excessive. Four layers of fiberglass and even more layers with epoxy, resulted in two solid, sturdy and heavy hulls. In hindsight it might have been enough using only one layer of $450g/m^2$ and one layer of $150g/m^2$. However, this was the first time the any of the group members worked with fiberglass and epoxy, and mistakes were inevitable.

6.2 Hardware

6.2.1 Jetson Nano

The Jetson Nano is a bit computational underpowered for this project, as the inference times using the YOLO-models is quite high. Being able to run larger models would mean better detection accuracy and range. The group thinks that upgrading the Jetson Nano to the newest version, the Jetson Orin Nano, would greatly increase performance. Looking at the specifications for the Jetson Orin Nano developer kit, we can see it has a 6-core arm CPU and a 1024-core GPU with 32 tensor cores [20]. Meanwhile the Jetson Nano has a 4-core arm CPU and a 128-core GPU with no tensor cores [18]. The increase in computational power the Jetson Orin Nano brings should allow for the use of a larger YOLO-model, and faster script times.

6.2.2 ZED2

The ZED2 camera have a wide FOV which is not ideal when trying to get long range detections. In sunny weather conditions the image colors are more prone to reflection and glare. Therefore the newer ZED2i camera might a better fit for this project. The ZED2i can be bought with a 4mm lens, which would mean a narrower FOV and longer range. Additionally the camera can be bought with a polarizing filter which could help reduce reflection and glare and increase detection accuracy [36]. However the narrower FOV might be an issue in the docking mission and collision avoidance mission.

6.2.3 GPS and RTK GPS comparison

When comparing the results of the accuracy tests in table 5.11 and 5.11.1 and observations made during testing an increase in accuracy can be observed.

An observation made during testing is that the loiter mode relies on two factors, both the internal IMU and the GPS, leading to the vessel being able to react quicker when moved by strong wind or being pushed than by gradual natural drift. This results in the loitering tests not showing a large improvement between regular GPS and RTK.

When in auto mode and using waypoints as reference the performance of the boat is noticeably improved. When considering that the obstacle course is only three meters wide, the vessels ability to reliably set waypoints is crucial.

The RTK base was only able to set waypoints down to 10-20cm, but can theoretically get down to as little as 2cm given enough setup time and good weather conditions. Further testing should be done in an area that is not as shielded by trees as the DNT naust. When using RTK in mission planner the screen displays the amount of GPS connections as well as the signal strength, this should be a consideration when choosing where to deploy and do tests.

It was observed that setting the WP_RADIUS below 0.2m sometimes lead to the vessel missing its waypoint, the manufacturer recommends not putting it below 0.2, but it should work below this with RTK.

6.2.4 Thrusters

When starting out the project the OmniX thruster configuration was chosen. This configuration is one of the standard configurations given in the firmware and makes the vessel highly maneuverable, as well as makes it able to move sideways. This maneuverability also has negative downside, as the angle removes an entire 20N of theoretical thrust force per thruster.

Another concideration when choosing thrusters was the size. The vessel is currently equipped with T200 thrusters, but could be fitted with the larger model T500. Fitting larger engines would however increase the current pull, which would require the wiring and possibly batteries to be changed.

6.3 Software

6.3.1 ROS

At the start of the project ROS was not used for retrieving the images from the ZED2 camera. Instead the ZED Python API was used. We were not having any issues using the Python API, until it suddenly stopped working. The following error code was displayed: "Cannot start camera stream." The group had made no changes to the camera settings or code, which made this error even stranger. Running the code using Python directly worked fine but running the code through ROS using the *roslaunch* function caused the error message. The ZED-SDK version used is 3.8, but version 4.0 was released around the time we started experiencing the API issues. The group can only speculate if there might have been some conflict between ROS and the ZED-SDK version 3.8 due to the transition to version 4.0. It can also be speculated that this might have triggered some bug in the ZED-SDK version 3.8 when using ROS.

To further debug the issue, the group tried the camera on a different Jetson Nano and then a different ZED2 camera on both Jetsons, but the error code persisted. After a day of debugging the decision was made to use the ROS package zed_wrapper instead. This was also the point we started having issues with depth readings returning nan. The main difference moving from the Python API to ROS is that the ROS images need to be converted using the Numpy function *frombuffer*. Perhaps there is some losses during the conversion process which causes the issues with the depth values. The ROS wiki uses a class called *CvBridge* and a function called *imgmsg_to_cv2* to convert from ROS image to OpenCv images [41]. However this function is based on Python2 and the group were not able to make it work with Python3.

Perhaps if the issues with the Python API is solved it would be more beneficial using the Python API over ROS. Another solution may be to migrate from ZED-SDK version 3.8 to version 4.0. This was not tried as at the time of working with this project, version 4.0 was still in early access.

6.3.2 YOLO

The confusion matrices from the YOLO-training are strange. In both the nano-model and small-model, a red buoy has been predicted 100% of the times when in reality it was just the background. The medium-model predicted red, yellow or green buoy 1/3 of the times. It's hard to know why this has happened, but it can be speculated that the data set is too small, and/or that the images are too alike.

The inference times were deemed a bit long. The models were tested with different image resolutions as an attempt to reduce the inference time. However as table 5.2 shows, the image resolution had no effect on the inference time. Therefore the camera was set to HD2K to get better detection results.

6.3.3 Machine vision

The most limiting factor for the machine vision is the depth readings. Whenever the depth value is nan, the code can't calculate the buoys GPS location. This forces the code to skip the rest of the code and try again. As the depth is read at the detection center, the code must run through the YOLO-model to find the detection center. The YOLO-model is also the most time consuming process to run through as seen in table 5.2. This causes the vessel to stop for up to several seconds as the code must run through the YOLO-model several times until the camera manages to determine the depth. During the time the vessel is stationary, it may drift and lose its heading which may cause it to get lost.

The neural depth setting removed almost all nan values, but caused the Jetson to crash. As the depth is computed onboard the camera itself, the crashes is likely due to the camera drawing too much power. In the end depth mode ultra was chosen.

Some reasons to why the depth is returned as nan may be because both cameras can't see the same point. However, the depth has been tested in areas where we are certain both cameras can see the buoy. Another reason may be that the cameras can't find the same point, due to the image not having enough unique points to match with each camera.

6.4 Mission testing

The mission testing was a bit challenging as we had to throw the buoys out from the pier. This limited the course setup, as it was hard getting the ideal distance between buoys. The buoys would constantly drift out of position, which caused the course to be deformed. The group tied some weight to the buoy lines to anchor the buoys, which helped a bit. If the group had access to a boat, we could have used the line to measure the depth. This way we could have more accurately found the best point to tie the weight to. Then the weight could have lied on the ocean floor without to much slack in the line. This would allow the group to set up a better testing course for the vessel.

None of the missions are using a sophisticated path planning algorithm. It seems the simple path planning used this far is adequate for these missions, but the path planning algorithm should be improved by future groups.

6.4.1 Navigation channel

The navigation channel mission proved it's capability, but further testing on a larger course is recommended. The rotating function was tested onshore by lifting the vessel and rotate it the set amount of degrees. This way it worked so further debugging the function in the ocean is needed.

6.4.2 Speed gate

The speed gate mission seem to be complete, but further testing on a larger course should be done.

6.4.3 Docking

The issue with the docking mission was that it's dependent on the vessel being able to move sideways. However, we were not able to make the vessel move sideways through mavros. Publishing to the */mavros/setpoint_velocity/cmd_vel_unstamped* topic, enabled forward movement and angular movement, but not sideways. When setting the *linear.x* or the *linear.y* value, the vessel would move forward for both values. Then we tried driving the motors individually by overriding the rc controls, using the */mavros/rc/overide* topic. However we were unable to setup permission to override the rc signals from the Jetson Nano. Then we looked at the mavros page in the ROS wiki, and saw an actuator control topic [42]. However this topic was not available for us, and after some debugging we were unable to figure out why.

Further development of this mission would contain some distance regulation from the buoy in front of the vessel. This would act as a measure to prevent to vessel touching the buoys by moving too far forwards or backwards. Also a smarter method for identifying when the vessel reaches the pier would need to be made. For the moment the vessel only "guesses" that it will reach the pier after a certain time.

6.4.4 Collision avoidance

This task was not prioritized during this project as the focus was on making the other missions work first. However the group still had some thoughts on how to solve this task. The main problem would be to identify the vessels used by the competition organizers. If the vessel is detected then they could be tracked, to keep track of their relative bearing. If their relative bearing does not change over time, that would mean the vessels are on a collision course.

One idea was to detect the vessels was to find a method for using the YOLO-model to detect objects that's not buoys or background and marking them as potential danger. Another idea was to use an IR-camera, to differentiate the vessels from background.

6.5 Experiences

6.5.1 Distribution of work

The group decided to divide the work based on the technical expertise of each individual, resulting in improved outcomes and more opportunities to learn. During the planning phase, group members pooled their knowledge and experiences to determine the optimal strategy. The optimal distribution of work was then determined based on each member's strengths and weaknesses. This allowed everyone to contribute based on their respective strengths, resulting in a sense of accomplishment and fulfillment for all. In addition, the research conducted in uncharted territories helped the team learn new things and acquire skills applicable to future endeavors.

6.5.2 Time management

The preliminary report included a Gantt chart and task-hour estimates, and the group set times for group and individual work to work efficiently and finish the project on time. Communication, regular meetings, and clear lines of communication kept everyone up to date on how tasks were going and any problems that came up. This transparency helped the group adjust in order to complete the project.

6.5.3 Risk assessment

The group prioritized safety, and multiple risk assessments were conducted. Due to the initial lack of an appropriate workspace, this was especially important when working with epoxy and fiberglass. Due to a comprehensive risk analysis, the group was able to complete this task without incident. Moreover, risk assessment helped the group identify potential dangers and implement mitigation strategies. By continuously evaluating risks, the team was able to ensure the safety of all project participants.

6.5.4 3D printing

There was no big issue with the 3D printing process, and minimal materials were wasted during the process. The only problem that occurred was that the 3D printer could not communicate with OctoPrint after being put on pause. This was in the process of placing the nut in the part while printing. Due to this, some grams of fillament were wasted less than 15 grams. To avoid this issue in the future, it is recommended to manually pause the print instead of using OctoPrint. Additionally, it could also be beneficial to print through the local SD card instead.

Chapter 7

Conclusion

The group decided that it was important for the boat to meet the requirements of NEK410:2021 and be able to withstand the rough conditions that can occur at sea. In addition to having a battery that has sufficient capacity to allow it to function for a period of at least one hour. The majority of the group's objectives have been accomplished.

The vessel's waterproofing is adequate, and the theoretical pull force value was calculated to be 167.01N, while the measured pull force was 158.8N which is sufficient.

The vessel that has been constructed for this project is intended to participate in the Autodrone competition, which requires it to navigate autonomously around a buoy course utilizing machine vision. In order to locate the buoys, the YOLOv8 algorithm was used to train a small YOLO-model. Field testing has shown that the ZED2-camera's issues with depth perception is the primary issue with the vessel. However, these tests have also demonstrated that the vessel is capable of navigating fully autonomously if this issue can be resolved. Use of the Python Zed API rather than ROS or upgrading to version 4.0 of the ZED-SDK could both be viable options for finding a workable solution to this issue.

It is possible that upgrading the vessel's hardware will improve its performance. Either the ZED2 camera or the Jetson Nano should be upgraded, or if possible, both should be upgraded. This could help get more accurate detections, as well as a longer detection range, while simultaneously improving the program's overall performance.

Bibliography

[1] European Space Agency. *RTK Fundamentals*. 2023. URL: https://gssc.esa.int/navipedia/index.php/RTK_Fundamentals (visited on 11th May 2023).

[2] Ardupilot. *Cube overview*. 2023. URL: https://ardupilot.org/copter/docs/common-thecubeorange-overview.html (visited on 21st May 2023).

[3] Ardupilot. *rospy*. 2023. URL: https://ardupilot.org/planner/ (visited on 21st May 2023).

[4] Autodrone. *Autonomous Sea Drones*. 2023. URL: https://www.autodrone.no/home#h.b8jdi94p46p9 (visited on 3rd Mar. 2023).

[5] BlueRobotics. *T200 Thruster*. 2023. URL: https://bluerobotics.com/store/thrusters/t100-t200-thrusters/t200-thruster-r2-rp/ (visited on 6th Mar. 2023).

[6] G. Bradski. 'The OpenCV Library'. In: *Dr. Dobb's Journal of Software Tools* (2000).

[7] FrSky. *RC documentation*. 2023. URL: https://www.frsky-rc.com/product/taranis-q-x7s/ (visited on 21st May 2023).

[8] Charles R. Harris et al. 'Array programming with NumPy'. In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2. URL: https://doi.org/10.1038/s41586-020-2649-2.

[9] J. D. Hunter. 'Matplotlib: A 2D graphics environment'. In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[10] igismap. *Formula to Find Bearing or Heading angle between two points: Latitude Longitude*. 2023. URL: https://www.igismap.com/formula-to-find-bearing-or-heading-angle-between-two-points-latitude-longitude/ (visited on 8th May 2023).

[11] TP-Link. *Documentation*. 2023. URL: https://www.tp-link.com/us/home-networking/usb-adapter/tl-wn722n/ (visited on 21st May 2023).

[12] Makita. *Batteri 6,0 Ah LXT*. 2023. URL: https://www.makita.no/product/bl1860b.html (visited on 7th Mar. 2023).

[13] *Maritime elektriske anlegg, NEK 410 : installasjoner og utstyr om bord i skip*. nob. Oslo.

[14] Microsoft. *It's how you make software*. 2023. URL: https://visualstudio.microsoft.com (visited on 3rd Mar. 2023).

[15] Milwaukee. *M18™ 5.0 AH BATTERI*. 2023. URL: https://no.milwaukeetool.eu/nn-no/m18-50-ah-batteri/m18-b5/ (visited on 7th Mar. 2023).

[16] Elnaz Namazi et al. *Geolocation estimation of target vehicles using image processing and geometric computation*. 2022. URL: https://www.sciencedirect.com/science/article/pii/S0925231222005732 (visited on 12th Apr. 2023).

[17] NVIDIA. *JetPack SDK*. 2023. URL: https://developer.nvidia.com/embedded/jetpack (visited on 21st May 2023).

[18] NVIDIA. *Jetson Nano*. 2023. URL: https://developer.nvidia.com/embedded/jetson-nano (visited on 11th May 2023).

[19] NVIDIA. *Jetson Nano Developer Kit*. 2023. URL: https://developer.nvidia.com/embedded/jetson-nano-developer-kit (visited on 21st May 2023).

[20] NVIDIA. *NVIDIA Jetson Orin Nano Developer Kit*. 2023. URL: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-orin/ (visited on 11th May 2023).

[21] OctoPrint. *OctoPrint The snappy web interface for your 3D printer*. 2023. URL: https://octoprint.org/ (visited on 9th Mar. 2023).

[22] Overleaf. *About us*. 2023. URL: https://www.overleaf.com/about (visited on 3rd Mar. 2023).

[23] PCschematic. *PC—Automation Software for automasjon og elektro*. 2023. URL: https://www.pcschematic.no/produkter/automation (visited on 3rd Mar. 2023).

[24] pypi. *labelImg 1.4.0*. 2023. URL: https://pypi.org/project/labelImg/1.4.0/ (visited on 16th May 2023).

[25] python. *datetime — Basic date and time types*. 2023. URL: https://docs.python.org/3/library/datetime.html (visited on 16th May 2023).

[26] Joseph Redmon and Ali Farhadi. 'YOLOv3: An Incremental Improvement'. In: *arXiv* (2018).

[27] Prusa Research. *PrusaSlicer introduction and download*. 2023. URL: https://www.prusa3d.com/page/prusaslicer_424/ (visited on 3rd Mar. 2023).

[28] ROS. *Documentation*. 2022. URL: http://wiki.ros.org/ (visited on 21st May 2023).

[29] *Rules and Task Descriptions*. Rev. 2022-1.0. AutoDrone, 2022. URL: https://drive.google.com/file/d/1JWbxQqRgRemu-KvenRzdBvyx52UTFXDs/view.

[30] Siemens. *NX*. 2023. URL: https://www.plm.automation.siemens.com/global/en/products/nx/ (visited on 3rd Mar. 2023).

[31] European Union Agency for the Space Programme. *Ehat is GNSS?* 2023. URL: https://www.euspa.europa.eu/european-space/eu-space-programme/what-gnss (visited on 11th May 2023).

[32] Magnus Stava, Markus Grorud Gaasholt and Kaung Htet San. *Developing waypoint navigation and buoy detection using YOLO for an autonomous surface vessel*. eng. 2021. URL: https://hdl.handle.net/11250/2782134.

[33] Stereolabs. *Høgskolen i Ålesund*. 2023. URL: https://www.yr.no (visited on 27th Apr. 2023).

[34] Stereolabs. *Meet ZED 2*. 2023. URL: https://www.stereolabs.com/zed-2i/ (visited on 10th Mar. 2023).

[35] Stereolabs. *Stereolabs Docs: API Reference, Tutorials, and Integration*. 2023. URL: https://www.stereolabs.com/docs/ (visited on 10th May 2023).

[36] Stereolabs. *ZED 2i*. 2023. URL: https://www.stereolabs.com/zed-2/ (visited on 11th May 2023).

[37] Richard Szeliski. *Computer Vision: Algorithms and Applications 2nd Edition*. Springer, 2021.

[38] Juan Terven and Diana Cordova-Esparza. *A Comprehensive Review of YOLO: From YOLOv1 to YOLOv8 and Beyond*. 2023. arXiv: 2304.00501 [cs.CV].

[39] Ultralytics. *ultralytics*. 2023. URL: https://docs.ultralytics.com/ (visited on 12th Apr. 2023).

[40] Ultralytics. *YOLOv8 Docs*. 2023. URL: https://github.com/ultralytics/ultralytics (visited on 12th Apr. 2023).

[41] ROS wiki. *Converting between ROS images and OpenCV images (Python)*. 2023. URL: http://wiki.ros.org/cv_bridge/Tutorials/ConvertingBetweenROSImagesAndOpenCVImagesPython (visited on 11th May 2023).

[42] ROS wiki. *Package summary*. 2023. URL: http://wiki.ros.org/mavros (visited on 13th May 2023).

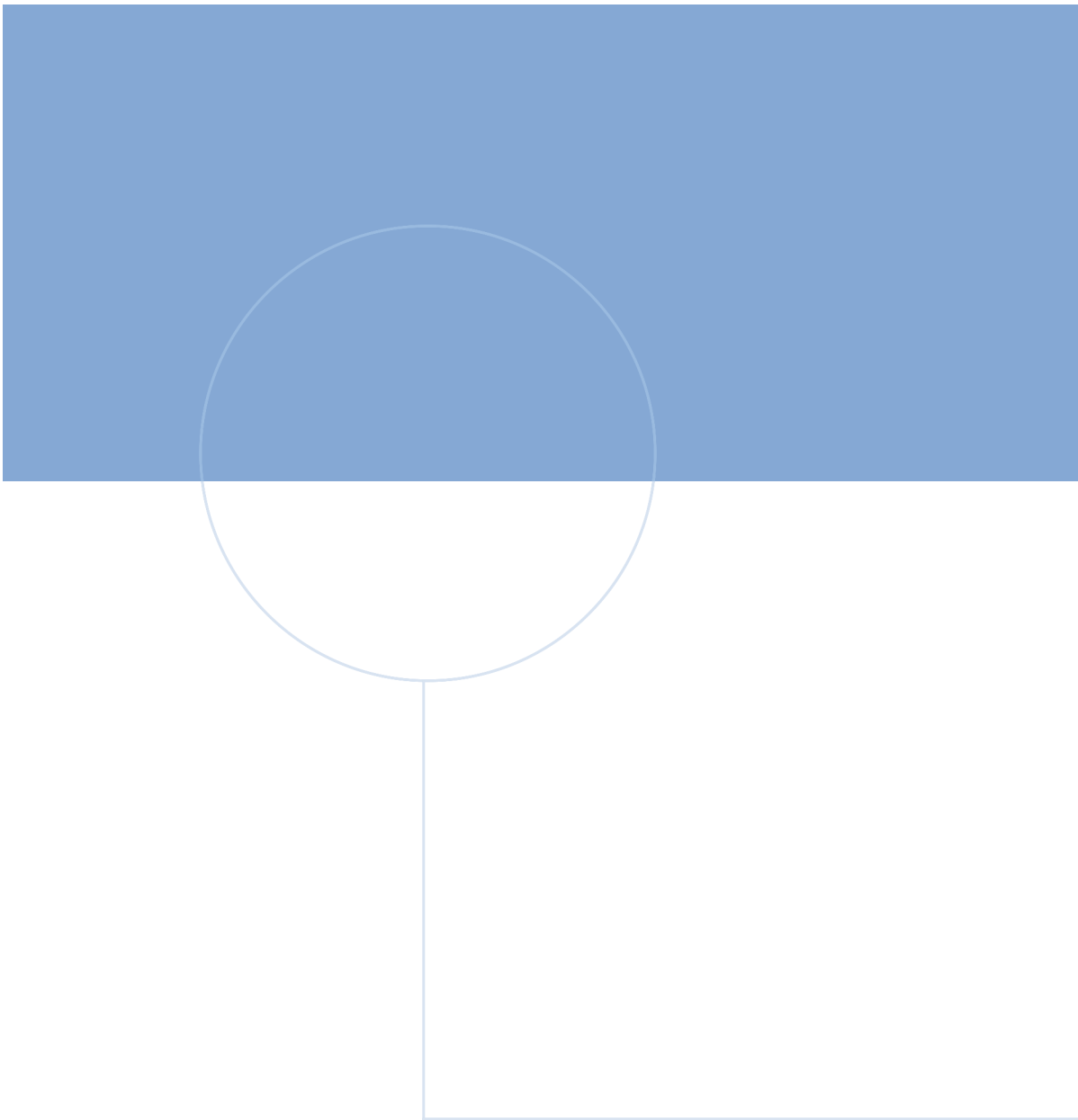
Appendix

A Wiki

Parts, CAD files and other relevant documents can be found in the wiki.
https://confluence.iir.ntnu.no/x/BIXzB

B GitHub

The workspace, mission codes and YOLO files can be found in the GitHub repository.
https://github.com/Jorgen14/SS_Martha



NTNU

Norwegian University of
Science and Technology