Bachelor's thesis

Yrian Hovde Øksne
Sigurd Odden
Mikal Aanning
Markus Holt

Graph-based slam and navigation in a simulated environment and buoy classification using YOLO for an Autonomous Surface Vessel

May 2020

NTNU
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

Bachelor's thesis

2020

NTNU

Norwegian University of
Science and Technology

Yrian Hovde Øksne
Sigurd Odden
Mikal Aanning
Markus Holt

Graph-based slam and navigation in a simulated environment and buoy classification using YOLO for an Autonomous Surface Vessel

Bachelor's thesis
May 2020

NTNU
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
Department of ICT and
Natural Sciences

Graph-based slam and navigation in a simulated environment and buoy classification using YOLO for an Autonomous Surface Vessel

Yrian Hovde Øksne (10010), Mikal Aanning (10011),
Sigurd Odden(10044) & Markus Holt (10038)

May 20, 2020

BACHELOR THESIS

Faculty of Information Technology and Electrical Engineering
Norwegian University of Science and Technology

Supervisor 1: Ottar L. Osen - Associate Professor at NTNU
Supervisor 2: Robin T. Bye – Associate Professor at NTNU
Supervisor 3: Girts Strazdins– Associate Professor at NTNU

Pages/Appendix
86/145

Mandatory self declaration / group declaration

Every group member is responsible for familiarizing themselves with what is legal aids, guidelines for their use and rules about the use of sources. This statement should make the group members aware of their responsibilities and what consequences that can occur if cheating takes place. Failure to accept this declaration does not exempt students from their responsibilities.

You fill out the declaration by clicking in the box to the right of each section 1-6:		
1.	I/We hereby declare that my/our answer is my/our own work, and that I/we have not used any other sources or received any help other than that mentioned in the answer	☒
2.	I/We declare that this reply: • has not been used for another exam at another department/university/university college or abroad • does not refer to the work of others without being stated • does not refer to own previous work without it being stated • has all the references listed in the literature list • is not a copy, duplicate or copy of the work or response of others	☒
3.	I/We are aware that violations of the above are regarded as cheating and may result in the cancellation of examinations and exclusion from universities and colleges in Norway, jf. University and college Act §§4-7 and 4-8 and Examination regulations §§14 and 15	☒
4.	I/We are aware that all submitted assignments can be plagiarized in Ephorus, see Guidelines for electronic submission and publication of credits for studio assignments	☒
5.	I/we are aware that ntnu university will handle all cases where there is suspicion of cheating according to the university's study regulations §31	☒
6.	I/we have familiarized ourselves with the rules and guidelines in the use of source and references on the library's website	☒

Preface

Unmanned surface vehicles (USVs) are water-borne vessels that are capable of operating without any onboard operators. USVs can both be controlled remotely by a operator or it can have autonomous functionalities. An unmanned surface vehicle that have full autonomous functionality is often referred to as an autonomous surface vehicle (ASV). [1]

Autonomous surface vehicles have a broad list of applications within different sectors. These applications can for instance be shipping, surveillance, inspection of coastal structures, environmental and climate monitoring.

This thesis is written by three automation engineering students and one computer science student at NTNU in Aalesund, and conclude three years of engineering studies. The task at hand was to develop an autonomous surface vehicle that would participate in a national competition between universities in Norway held in June of 2020. However, halfway through the project a pandemic led to a national lockdown. Causing the competition to be cancelled, and campus premises to be closed. Thus, the physical vessel never passed the prototype testing stage. This causing to a new problem to be addressed.

What intrigued us to choose this project were that we saw the possibilities to use our knowledge from three years of studies to create something physical. With the increased interest in the drone industry and autonomous vehicles, this project caught our interest due to it being relevant knowledge for the future. The fact that the project also involved a national competition between universities and colleges made the task very attractive.

Abstract

The AutoDrone competition is a national maritime robotics event hosted by the University of South-Eastern Norway (USN) and is held in Horten, Norway. The competition is open to fully unmanned and autonomous boats within dimensional bounds set by the host. NTNU wishes to have a representative in this new yearly competition. The group members of this project wishes that students who participate in similar projects can start development at a faster rate in order to produce a better fleet for the NTNU representative.

This thesis aims to develop parts of a robot that can fit the criteria of entrance and compete at this event. In addition, the thesis aims to help the next years students to more easily test and begin development of software related to this competition.

The results showcased that parts of an ASV can be developed remotely and that a simulated environment can be used for testing and development of software regarding ASVs for the AutoDrone competition. This project shows a solid foundation for further development of ASV/USV solutions able to compete amongst the other universities attending AutoDrone. In addition, this project shows some of the different possibilities gained from testing and developing in a simulated environment, what technologies and solutions could be used to create a fully autonomous boat.

Abstrakt (Norwegian)

AutoDrone konkurransen er et nasjonalt maritimt robotikk event holdt av Universitetet i Sør-Øst Norge (USN) og er holdt i Horten, Norge. Konkurransen er åpen for fullstendig autonome og ubenammede båter innen gitte dimensjonelle krav satt av avholder. NTNU ønsker å ha en representant i den nye konkurranen. Gruppemedlemmene på dette prosjektet øsnker at studenter som deltar i lignende prosjekter kan starte utvikling av programvare i en raskere rate slik at de får et bedre produkt å representere NTNU med.

Denne avhandlingen sikter til å utvikle deler av en robot som møter ingangskravene til konkuransen. I tillegg sikter denne avhandlignen til å hjelpe studenter av neste årskull til å komme lettere og raskere i gang med utvikling av programvare relatert til konkurransen.

Resultatene viser at deler av en ASV kan bli utviklet fra fjernt og at simulerte miljøer kan bli brukt for testing og utvikling av programvare for ASVer som skal delta i konkurransen. Dette prosjektet viser et solid grunnlag for videreutvikling av ASV/USV løsninger som gjør det mulig å delta og konkurere blant de andre universitetene som er møtende på AutoDrone. I tillegg viser dette prosjektet mulighetene man tjenerer ved å teste og utvikle i et simulert miljø, hvilke teknologier og løsninger som kan bli brukt til å konkstruere en fullstendig autonom båt.

Acknowledgement

We would like to thank all the contributors who have given us support through the project, and in particular we would like to thank:

- Our supervisors Ottar L. Osen, Girts Strazdins & Robin T. Bye for all support, motivational conversations and great guidance throughout the whole project.
- PhD candidates Anete Vagale, Elias Hasle and Aleksander Larsen Skrede for guidance with path-planning, navigation and the ROS framework.
- Laboratory engineers Anders Sætersmoen, Øyvind Andre Hanken & Markus Gutvik Lyngstad for assistance in purchasing and lending equipment.
- Aalesund Sports Diving Club for lending the floating dock used for data collection.
- The Roboboat team including students from Naval Architecture, faculty of engineering for the assistance on hull design and creation.

Contents

Preface ii

Acknowledgement v

Acronyms 2

1 Introductions **8**

1.1 Background 8

1.2 Problem Formulation 9

1.3 Objectives 13

1.4 Structure of the Report 14

2 Theoretical basis **15**

2.1 Autonomous Surface Vehicle 15

2.2 Motion Variables 15

2.3 Reference Frames 16

2.3.1 Earth Centered Reference Frames 16

2.3.2 Geographic Reference Frames 17

2.3.3 Transformation from geodetic- to NED reference frame 18

2.3.4 Body-Fixed Reference Points 19

2.3.5 Transformation between BODY and NED 19

2.4 Maneuvering 20

2.5 GNSS 21

2.6 Hull configuration 22

2.7 Thruster configuration for Catamaran 24

2.8 Computer Vision 25
2.8.1 Graph-based SLAM 25
2.8.2 Stereo Camera 26
2.9 Data-collection and -annotation 28
2.10 Deep Learning 29
2.10.1 YOLO - You Only Look Once 30
2.10.2 Mean Average Precision 33
2.11 Extended Kalman Filter 34

3 Materials and methods **37**
3.1 Project Organisation 37
3.2 Deviation management 38
3.3 Gantt-diagram work description 40
3.3.1 Pre-project 40
3.3.2 Research 40
3.3.3 Planning Design 41
3.3.4 Software 41
3.3.5 Hardware 41
3.3.6 Testing 41
3.4 Materials 42
3.4.1 Catamaran Hull 42
3.4.2 "X" Holonomic thruster calibration 42
3.4.3 T200 Thruster 42
3.4.4 Nvidia Jetson TX2 43
3.4.5 Lithium-ion Battery 44
3.4.6 Zed 2 Stereo Camera 45
3.4.7 Pixhawk mRo controller 46
3.5 Software 47
3.5.1 List of software used 47
3.5.2 ROS with Linux Ubuntu 47

3.5.3 Computer vision using OpenCV 48
3.5.4 Data-collection and -annotation 49
3.5.5 Deep learning with YOLO 49
3.6 Simulation and visualization 50
3.6.1 Gazebo with ROS 50
3.6.2 Rviz with ROS 50
3.6.3 USV model and sensor package 51
3.7 ROS Development Studio 52
3.8 Navigation 52
3.8.1 Localization 53
3.9 Transformation to a common reference frame 54
3.10 Sensor Data Preprocessing 55

4 Result **56**
4.1 Simulation 56
4.1.1 Testsim in ROS Development Studio 56
4.1.2 WAM-V USV 57
4.2 Sensor fusion 57
4.3 Sensor Data Preprocessing 57
4.4 Mapping 61
4.5 Materials 63
4.5.1 Jetson TX2 63
4.5.2 Pixhawk MRo with GNSS reciever 63
4.5.3 Zed 2 Stereo Camera 64
4.5.4 Hull and Thruster 64
4.6 Data collection and annotation 65
4.7 Buoy detection 67
4.7.1 Training result 1: 4000 iterations 67
4.7.2 Training result 2: 6000 iterations 68
4.7.3 Training result 3: 7000 iterations 69

4.7.4 Training result 4: 8000 iterations 70
4.7.5 Object Detection 71

5 Discussion **73**
5.1 Physical Vessel 73
5.2 Mapping 73
5.3 Navigation 74
5.4 Data-collection and annotation 74
5.5 Object detection 75
5.6 ROS Development Studio 76
5.7 Scrum to Gantt transition 76
5.8 Phase improvement potential 77
5.8.1 Pre-project 77
5.8.2 Research 78
5.8.3 Planning Design 78
5.8.4 Software 78

6 Conclusions **80**

Bibliography **82**

Appendices **86**
A Preproject report 86
B Gantt diagram 86
C Hour list 86
D MVP 86
E BOM 86
F AutoDrone 2020 Rules and Task Description 86
G Buoy data collection 86
H RGB-D Mapping launch file 86
I RGB-D Localization file 86
J YOLO Training file 86

K YOLO mAP validation file 86
L YOLO Processing Video file 86
M YOLO Processing Image file 86
N YOLO Labels file 86
O YOLO CFG file 86
P Patrol file 86
Q WAM-V simulation model 86
R Setup guide simulation 86
S Meeting Minutes 86
T Progress reports 86

A Preproject report **87**

B Gantt diagram **103**

C Hour List **105**

D MVP **107**

E BOM **111**

F Autodrone 2020 Rules and Task Description **113**

G Buoy data collection **127**

H RGB-D Mapping launch file **130**

I RGB-D Localization launch file **136**

J YOLO Training file **142**

K YOLO mAP validation file **144**

L YOLO Processing Video file **169**

M YOLO labels file **171**

N YOLO CFG configuration file **173**

O Patrol file **180**

P WAM-V simulation model **183**

Q Setup guide simulation **203**

R Meeting minutes **207**

S Progress Reports **212**

Terminology

IMO International Maritime Organization, is the united nations specialised agency with responsibility for the safety and security of shipping and and the prevention of marine and atmospheric pollution by ships

API Application Programming Interface, activates functions from a remote software

MavLink Lightweight, efficient and reliable communication protocol mainly used for drones

Stereo Image rectification The process of finding similarities in two images and transforming them to a common image plane such a way that the corresponding points have the same row coordinate

Abbreviations

ASV Autonomous surface vehicles

USV Unmanned surface vehicle

GDP Gross Domestic Product

MASS Maritime Autonomous Surface Ships

DOF Degrees of Freedom, number of configurations for a object

RGB-D Red, Green, Blue - Depth

SLAM Simultaneous localization and mapping

RGB Red, Green, Blue

HSV Hue, Saturation and Value

YOLO You Only Look Once

IoU Intersection over Union

CNN Convolutional Neural Network

R-CNN Region based Convolutional Neural Network

AP Average Precision

mAP mean Average Precision

EKF Extended Kalman Filter

ROS Robot Operating System

OpenCV Open Source Computer Vision Library

MVP Minimum Viable Product

FOV Field of view

IMU Inertial Measurement Unit

RC Radio Control

VRX Virtual RobotX

OSRF Open Source Robotics Foundation

ROSDS Robot Operating System Development Studio

RTABMap Real-Time Appearance Based Mapping

XML Extensible Markup Language

ROI Region Of Interest

PWM Pulse Width Modulation

ESC Electronic Speed Controller

URDF Unified Robot Description Format

List of Figures

1.1 Mission Layout 10
1.2 Obstacle Channel layout 11
1.3 Collision avoidance layout 11
1.4 Visual Docking layout 12
1.5 Speed gate layout 12
1.6 Course buoys 13

2.1 ECEF-, ECI-, NED- and BODY reference frames [2] 17
2.2 Body fixed reference points [2] 19
2.3 Location determination based on intersecting spheres 22
2.4 Using the earth as the fourth space 22
2.5 Flat-Bottomed Hull 23
2.6 Round-bottomed Hull 23
2.7 SWATH Hull 24
2.8 Catamaran Hull 24
2.9 Motion of thrusters in X-Configuration 25
2.10 Stereo camera system 27
2.11 Stereo camera system 27
2.12 Stereo camera system included the point of interest 27
2.13 Disparity of the point 28
2.14 YOLO network Architecture 31
2.15 IoU Formula 33

2.16 Precision-Recall curve 34
2.17 Extended Kalman Filter loop [3] 36

3.1 Catamaran 42
3.2 2D Drawing of the T200 Thruster 43
3.3 Jetson TX2 developer kit 44
3.4 Dimensions of the ZED 2 Camera 45
3.5 Pixhawk mRo controller 46
3.6 Propulsion options 51
3.7 RGB-D mapping with lidar 53
3.8 Stereo mapping 54

4.1 Room mapping using Turtlebot with RTAB script 56
4.2 Simulation model of the WAM-V 14 USV 57
4.3 Unfiltered odometry 58
4.4 Filtered odometry, unfiltered point detection 58
4.5 Unfiltered point detection 59
4.6 Filtered point detection 60
4.7 Inliers detected using stereo images 61
4.8 Depth image calculated from rectified stereo images 62
4.9 Successfully mapping with RGB-D and 3D lidar 63
4.10 Prototype of the vessel 64
4.11 Collected image of an yellow buoy 66
4.12 Bounding box of the yellow buoy 66
4.13 Detection result 1 67
4.14 mAP 1 67
4.15 Detection result 2 68
4.16 mAP 2 68
4.17 Detection result 3 69
4.18 mAP 3 69
4.19 Detection result 4 70

4.20 mAP 4 70
4.21 Buoy detection of all three buoys 71
4.22 Green buoy detection 71
4.23 Red buoy detection 71
4.24 Yellow buoy detection 72

List of Tables

1.1 Obstacle channel score-sheet 10
1.2 Collision avoidance score-sheet 11
1.3 Visual Docking score-sheet 12
1.4 Speed Gate score-sheet 12

2.1 The notation of SNAME (1950) for marine vessels [2] 16
2.2 The advantages and disadvantages of different hull configurations 23

Chapter 1

Introduction

1.1 Background

Norway counts for only 0.07% of the world population, yet is a maritime superpower and a frontrunner when it comes to maritime technology. The maritime industry is one of Norway's oldest business activities and employs around 90,000 people today, maritime companies are often a cornerstone industry in local communities. The industry receives a strong national attention, for good reason since it covers 10% of Norway's GDP. [4]

The International Maritime Organization has shown an enormous interest in the possibility of applying ships with varying degrees of automation, through a regulatory scoping exercise on Maritime Autonomous Surface Ships (MASS).[5]

Unmanned vessels can be used in various environments where it is a high risk for humans to operate, removing the risk of human operators getting harmed and removing the human element, in which can be a cause of accidents have been the direct cause of many catastrophic events in marine environments.

Small seadrones may act like testbeds for mimicking real-world challenges in the maritime industry. Some of these challenges includes object detection, path-planning, error-correction, environment mapping and target tracking. RoboNation, an open community of students, industry leaders and mentors, have a few years created an international competition called RoboBoat,

where students will be solving some of these challenges. MARKOM2020, an coalition of different Norwegian universities, have taken inspiration from RoboBoat to create their own national competition here in Norway, called Autodrone.

1.2 Problem Formulation

NTNU in Aalesund wants a group of students represented in the competition Autodrone. This is the bachelor project is based upon. Thus, the problem specification will be based upon the content- and rules of the competition which can be found in appendix F.

The task at hand will be to create a sea-drone within the requirements and solve the courses in the competition. The project consists of both software- and hardware development.

Problems to be addressed - Hardware

- **Energy source:** The drone must be battery powered.
- **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.
- **Propulsion:** Any propulsion system may be used (thruster, paddle, etc.). However, all moving parts must have protection. For instance, a propeller must be shrouded.
- **Weight:** The entire maritime system shall weigh less than 70 kg.
- **Size:** The vessel should not be wider nor higher than 0.9 meter, and not longer than 1.8 meter.

Problems to be addressed - Software

- **Autonomy:** The drone shall be fully autonomous and shall have all autonomy decisions made on-board the ASV.
- **Communication:** The drone cannot send or receive any control information to and from Operators Control Station while in autonomous mode.

- **Remote-controllable:** The drone must be remote-controllable from an operator control station.
- **Computer vision:** The drone should be able to have an open-camera stream for object detection.
- **Object recognition:** The drone should be able to recognize different objects.

Mission Layout at Autodrone

The following will describe the four different courses that the vessel are required to manoeuvre through at Autodrone. Figure 1.1 shows the mission layout of the course, this includes an obstacle channel, collision avoidance, visual docking and a speed gate.

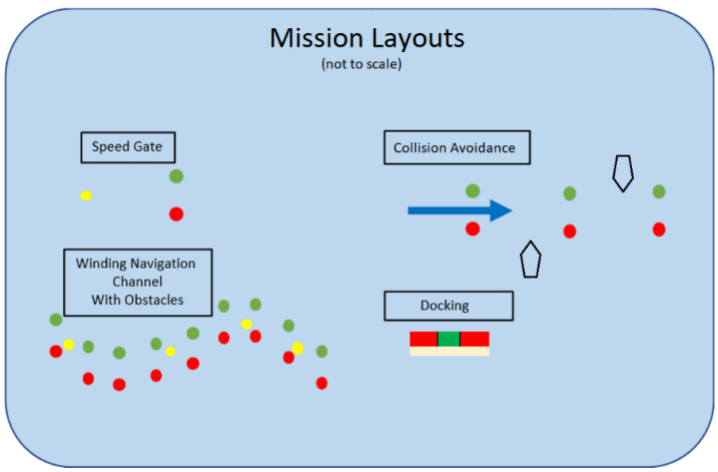


Figure 1.1: Mission Layout

- **Obstacle channel:** An successful completion of the obstacle course without touching the objects and staying in the path will demonstrate the vessel's ability to sense and manoeuvre through an difficult path. The path will be defined by red buoys on the right side, green buoys on the left side and yellow buoys scattered in the path.

Table 1.1: Obstacle channel score-sheet

Points		
Gate	**Obstacle**	**Time**
+5	-10	100 -10/min

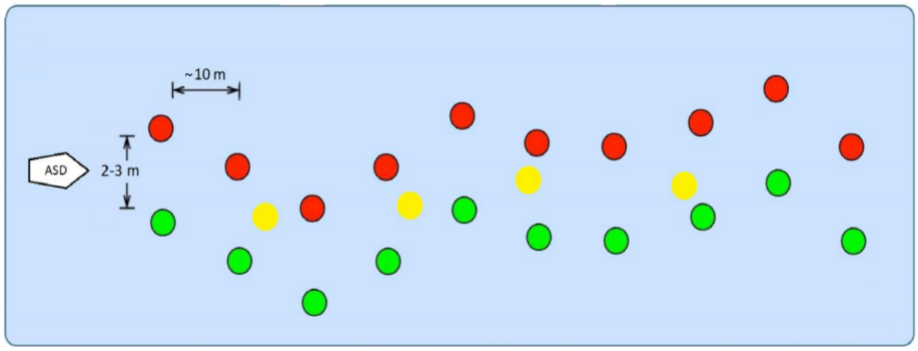


Figure 1.2: Obstacle Channel layout

- **Collision avoidance:** An successful completion of the collision avoidance without hitting the dynamic obstacles and staying in the path will demonstrate the vessel's ability to detect objects coming from both port and starboard.

Table 1.2: Collision avoidance score-sheet

Points		
Gate	**Collision**	**COLREGs**
+10	-30	+30/rule followed

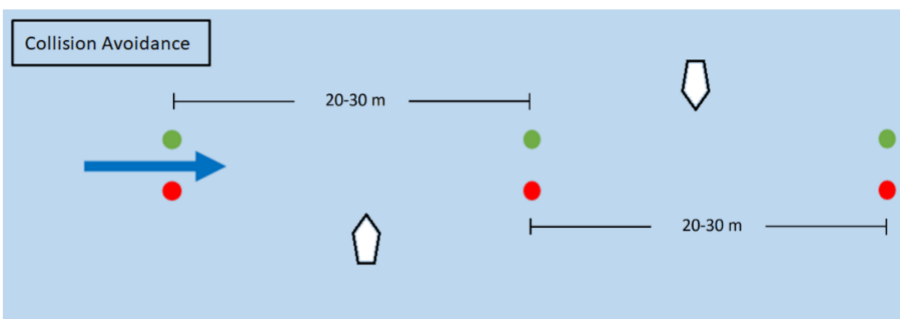


Figure 1.3: Collision avoidance layout

- **Visual Docking:** A successful completion of the visual docking without hitting the two buoys nor the dock will demonstrate the vessels ability to detect distance to objects based on it's orientation. It will also shows that the thrusters can work at a stable rate to keep the boat stationary.

Table 1.3: Visual Docking score-sheet

Points		
Dock Reached	**Docking Correct**	**Docking time**
+10	+30	+2 /sec docked

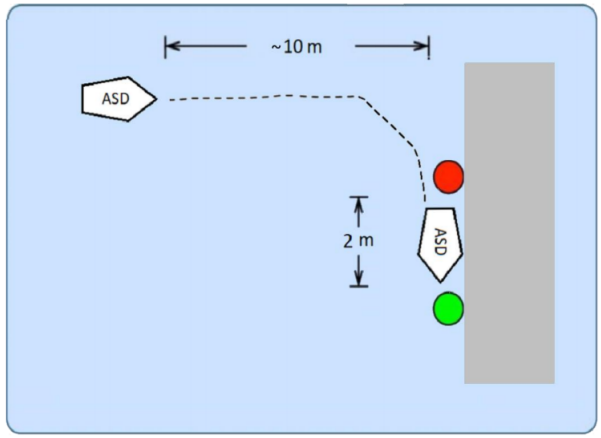


Figure 1.4: Visual Docking layout

- **Speed Gate:** A successful completion of the speed gate course will demonstrate the vessel's hull form efficiency coupled with its propulsion system, and the resulting maneuverability.

Table 1.4: Speed Gate score-sheet

Scoring	
Time	**Points**
Fastest	80
Second	50
Third	20

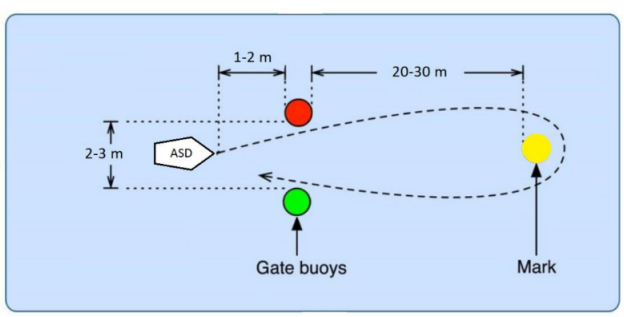


Figure 1.5: Speed gate layout

The different courses are created by using three types of buoys; red, green and yellow as illustrated in figure 1.6. The red and green buoy can be bought from Biltema, they both have the Art.nr. 25-017 and are originally both red. The green buoy has therefore been spray-painted green using paint from Biltema with the Art.nr. 36-577. The yellow buoy has been bought from Sommerbutikken with the product name "Badebøye/Markeringsbøye".

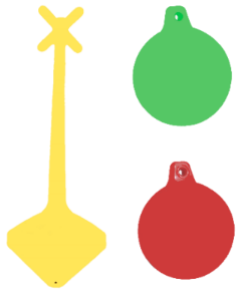


Figure 1.6: Course buoys

1.3 Objectives

The Objectives for this report thesis are:

1. Building the Software of the vessel
2. Building the vessel using different components
3. Implementing the software to the vessel
4. Test the vessel in different environments with different courses
5. Compete in Autodrone

1.4 Structure of the Report

The rest of the report is structured as follows.

Chapter 2 - Theoretical basis: Chapter two gives an introduction to the theoretical background on the subjects used to make decisions and solve problems later on in the report

Chapter 3 - Method: Contains a description of the methodology and materials that were considered throughout the project to solve the problem at hand

Chapter 4 - Result: Contains a description of the finished results of the software and hardware revolving the project

Chapter 5 - Discussion: A summary of the results achieved and problems encountered throughout the project and potential improvements for further development

Chapter 6 - Conclusions: This chapter present an overall conclusion from the project implementation and answers the problem statement

Chapter 2

Theoretical basis

This chapter contains theory that has been used in this thesis. This includes applications such as materials, software and equations.

2.1 Autonomous Surface Vehicle

An Unmanned Surface Vehicle (USV) is a maritime floating- or semi-submerged vessel used eg. for transportation and scientific research, without any onboard operator. Whereas an USV is manually controlled or partly autonomous, an Autonomous Surface Vehicle (ASV) is completely autonomous without any interactions from a operator. The vehicle is able to make decisions and determine action by itself. [1]

2.2 Motion Variables

For a vessel moving in six degrees of freedom (DOF), it is necessary to define six independent coordinates to describe position and orientation. The first three coordinates, and their time derivatives, correspond to the position and translational displacement along the x, y and z axes, while the last three is used to describe orientation and rotational motion [2]. The notation defined by the Society of Naval Architects and Marine Engineers (SNAME) can be seen in table 2.1.

Table 2.1: The notation of SNAME (1950) for marine vessels [2]

DOF		Forces and moments	Linear and angular velocities	Positions and Euler angles
1	motions in the x direction (surge)	X	u	x
2	motions in the y direction (sway)	Y	v	y
3	motions in the z direction (heave)	Z	w	z
4	rotation about the x axis (roll, heel)	K	p	ϕ
5	rotation about the y axis (pitch, trim)	M	q	θ
6	rotation about the z axis (yaw)	N	r	ψ

2.3 Reference Frames

In order to describe motion of a marine craft in 6 DOF, it is necessary to to define earth centered- and geographical reference frames [2].

2.3.1 Earth Centered Reference Frames

ECI: The Earth-Centered inertial (ECI) frame $\{i\} = (x_i, y_i, z_i)$ is an inertial reference frame in which Newtons laws of motion can be applied. The origo of {i} is located in the earths center o_i.
ECEF: The Earth-centered Earth-fixed (ECEF) reference frame $\{e\} = (x_e, y_e, z_e)$ with the earths center as origo o_e. The axis rotate relative to the inertial frame (ECI), which is fixed in space. For marine crafts moving in relative low speed, the earths rotation can be neglected and {e} can be considered a fictitious (inertial) force. {e} is used for global guidance, navigation and control for ships in transit between continents.

2.3.2 Geographic Reference Frames

Geographical reference frames are usually chosen as tangent planes on the surface of the earth.

Terrestrial Navigation: The tangent plane on the surface of the earth moves with the vessel, specified by longitude-latitude values (l, μ).

Local navigation: The tangent plane is fixed at constant values (l_0, μ_0) and the position is computed with respect to a local coordinate origin.

NED: North-East-Down coordinate system $\{n\} = (x_n, y_n, z_n)$, with origo o_n defined by the Worlds Geodetic System's (WGS84) reference ellipsoid. Flat-Earth navigation is accurate for smaller geographical areas ($10km \times 10km$). For flat Earth navigation one can assume that {n} is inertial such that Newtons law's of motion can be applied.

BODY: The Body-fixed reference frame $\{b\} = (x_b, y_b, z_b)$ with origo o_b fixed on the vessel, usually a point midships in the waterline referred to as CO, such that the coordinate frame is moving with the vessel. For marine craft, the body axes x_b, y_b and z_b is often chosen to coincide with the principal axes of inertia (see Figure 2.2).

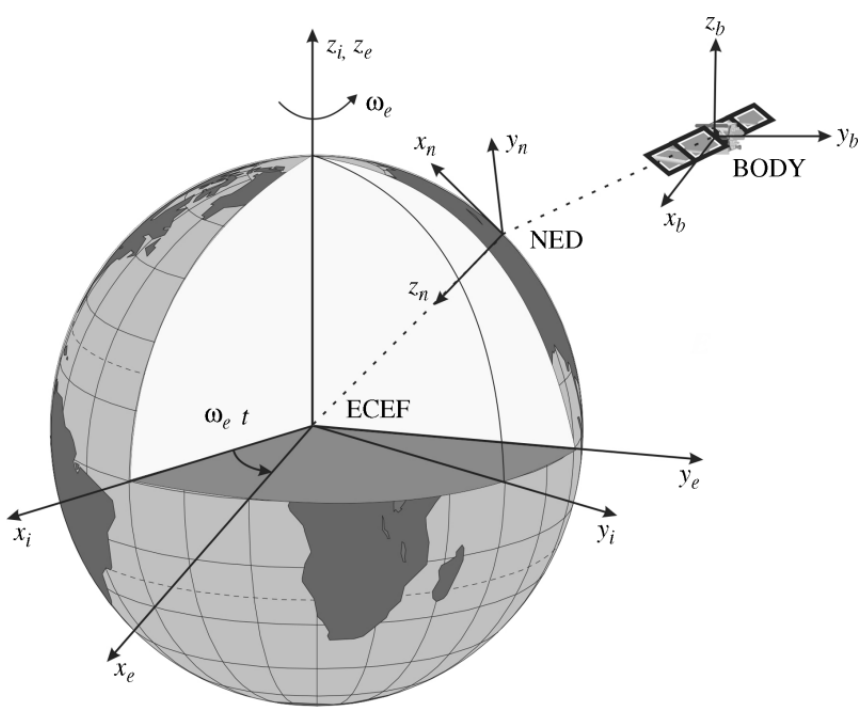


Figure 2.1: ECEF-, ECI-, NED- and BODY reference frames [2]

2.3.3 Transformation from geodetic- to NED reference frame

Transformation from geodetic- to NED reference frame is necessary e.g. for using GNSS in navigation. The transformation begins by estimating small changes in latitude $d\mu$ and longitude dl.

$$d\mu = \mu - \mu_0 \tag{2.1}$$

$$dl = l - l_0 \tag{2.2}$$

To convert geodetic latitude and longitude to the North- and East- coordinates, the estimation uses the radius of curvature in the prime vertical (R_N) and the radius of curvature in the meridian (R_M).

$$R_N = \frac{R}{\sqrt{1-(2f-f^2)\sin^2\mu_0}} \tag{2.3}$$

$$R_M = R_N \frac{1-(2f-f^2)}{1-(2f-f^2)\sin^2\mu_0} \tag{2.4}$$

where R is the equatorial radius of the planet and f is the flattening of the planet.

Small changes in the North- dN and East dE positions are approximated by

$$dN = \frac{d\mu}{\arctan(\frac{1}{R_M})} \tag{2.5}$$

$$dE = \frac{dl}{\arctan(\frac{1}{R_N \cos\mu_0})} \tag{2.6}$$

With the conversion of the North and East coordinates to the flat Earth x and y coordinates, the transformation has the form of

$$\begin{bmatrix} p_x \\ p_y \end{bmatrix} = \begin{bmatrix} \cos\psi & \sin\psi \\ -\sin\psi & \cos\psi \end{bmatrix} \begin{bmatrix} N \\ E \end{bmatrix} \tag{2.7}$$

where ψ is the degrees clockwise between the x-axis and north [6].

2.3.4 Body-Fixed Reference Points

The following reference points are defined with respect to CO:

CG - Center of Gravity

CB - Center of Buoyancy

CF - Center of Flotation (Located at a distance LCF from CO in the x-direction)

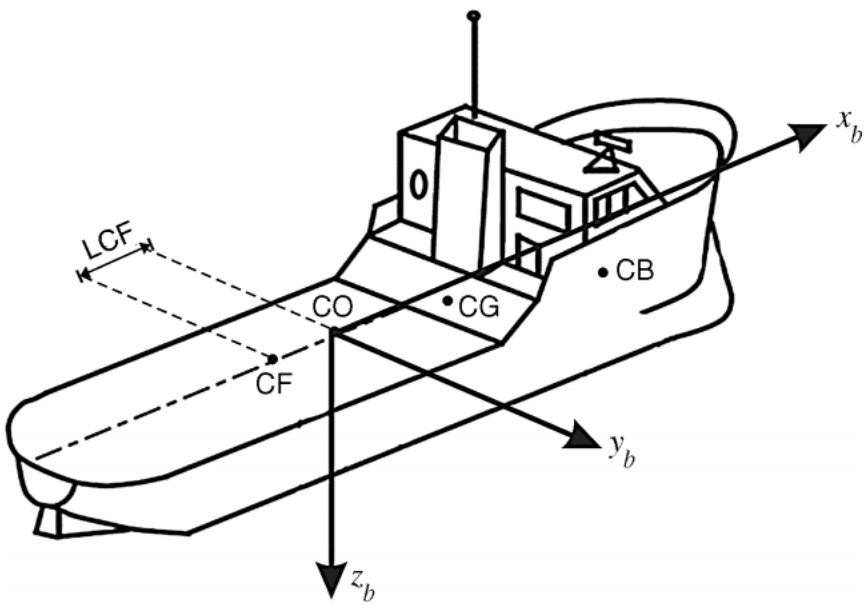


Figure 2.2: Body fixed reference points [2]

2.3.5 Transformation between BODY and NED

In guidance, navigation and control applications it is common to use the zyx-conversion from {n} to {b} specified in the terms of each Euler angles for each coordinate ψ (z), Θ (y) and ϕ (x). The conversion is done by performing three principal rotations about the z-, y-, and x-axes. The matrix can be transposed in order to obtain the opposite transformation from b to n

$$\mathbf{R}_b^n(\Theta_{nb}) = \mathbf{R}_b^n(\Theta_{nb})^T \tag{2.8}$$

The rotation sequence is described as

$$\mathbf{R}_b^n(\Theta_{nb}) := \mathbf{R}_{z,\psi}\mathbf{R}_{y,\Theta}\mathbf{R}_{x,\phi} \tag{2.9}$$

and the inverse transformation

$$\mathbf{R}_b^n(\Theta_{nb})^{-1} = \mathbf{R}_n^b(\Theta_{nb}) = \mathbf{R}_{z,\psi}^T\mathbf{R}_{y,\Theta}^T\mathbf{R}_{x,\phi}^T \tag{2.10}$$

As proved in [2], using the principal rotation matrices equation (2.9) can be expanded

$$\mathbf{R}_b^n(\Theta_{nb}) = \begin{bmatrix} \cos\psi\cos\Theta & -\sin\psi\cos\phi + \cos\psi\sin\Theta\sin\phi & \sin\psi\sin\phi + \cos\psi\cos\phi\sin\Theta \\ \sin\psi\cos\Theta & \cos\psi\cos\phi + \sin\phi\sin\Theta\sin\psi & -\cos\psi\sin\phi + \sin\Theta\sin\psi\cos\phi \\ -\sin\Theta & \cos\Theta\sin\phi & \cos\Theta\cos\phi \end{bmatrix} \tag{2.11}$$

The body-fixed velocity vector $\mathbf{v}_{b/n}^b$ can then be represented in {n} as

$$\mathbf{v}_{b/n}^n = \mathbf{R}_b^n(\Theta_{nb})\mathbf{v}_{b/n}^b \tag{2.12}$$

2.4 Maneuvering

In order to simplify operation, dynamics of a marine vessel can be explained using two different theories: maneuvering and seakeeping. Maneuvering refers to the motion of a marine vessel in calm water with absence of wave excitation. Seakeeping refers to the motion of a marine vessel in constant course and speed when there is wave excitation. A 6DOF maneuvering model of a marine vessel can be explained using a non-linear mass-damper-spring system with constant coefficients:

$$\dot{\eta} = J_\Theta(\eta)\nu \tag{2.13}$$

$$\underbrace{M_{RB}\dot{\nu} + C_{RB}(\nu)\nu}_{\text{Rigid-Body forces}} + \underbrace{M_A\dot{\nu}_r + C_A(\nu_r)\nu_r + D(\nu_r)\nu_r}_{\text{Hydrodynamic forces}} + \underbrace{g(\eta) + g_o}_{\text{Hydrostatic forces}} = \tau + \tau_{wind} + \tau_{wave} \tag{2.14}$$

where

$\boldsymbol{M_{RB}}$ - the system inertia matrix $\boldsymbol{M_A}$ - the added mass matrix and

$\boldsymbol{M} = \boldsymbol{M_{RB}} + \boldsymbol{M_A}$ - system inertia matrix (including added mass)

$\boldsymbol{C}(\boldsymbol{\nu_r}) = \boldsymbol{C_{RB}}(\boldsymbol{\nu_r}) + \boldsymbol{C_A}(\boldsymbol{\nu_r})$ - Coriolis-centripetal matrix (including added mass

$\boldsymbol{D}(\boldsymbol{\nu_r})$ - damping matrix

$\boldsymbol{g}(\boldsymbol{\eta})$ - vector for gravitational/buoyancy forces and moments

$\boldsymbol{g_0}$ - vector used for pretrimming (ballas control)

$\boldsymbol{\tau}$ - vector for control inputs

$\boldsymbol{\tau_{wind}}$ - vector for wind forces

$\boldsymbol{\tau_{wave}}$ - vector for wave-inducted forces

A complete description of the maneuvering model can be found in [2]. Because of its complexity, the 6DOF model is mainly used for controller-observer design, prediction and computer simulations.

2.5 GNSS

Global Navigation Satellite System is a constellation of satellites providing signals from space that transmit and timing data to GNSS receivers.

GNSS provides global coverage, GNNS includes GPS, GLONASS, Galileo or BeiDou systems. Each system consists of a set of satellites which send continuous signal towards the earth. All the satellites are synchronized thanks to the reference time scale defined by the system, so that the signals coming from different satellites of the same constellation share the same reference time scale. Uses GNSS to receive the position of the antenna from signals that are coming from the satellites. The receiver translates these signals into coordinates.

Knowing the distance to one satellite, the receiver antenna location surely lies on a sphere around the satellite with radius equal to the measured distance between the satellite and the receiver location. Knowing the distance of two satellites, the receiver antenna must lie on the intersection of the spheres with known radii around both satellites. With the third satellite, a third sphere can be determined that will intersect the circle in exactly two points.

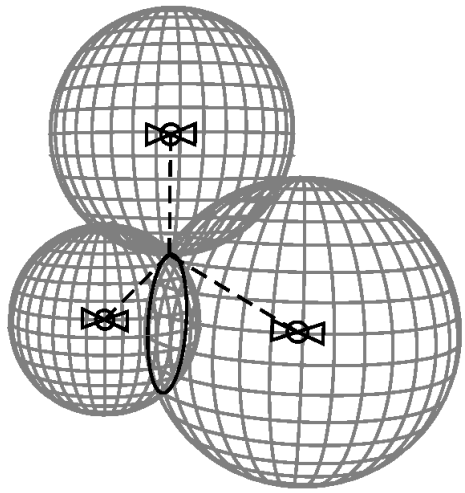


Figure 2.3: Location determination based on intersecting spheres

The fourth satellite will make it possible to select the correct point out of the two given points found earlier, and this point is the location of the receiver. [7]

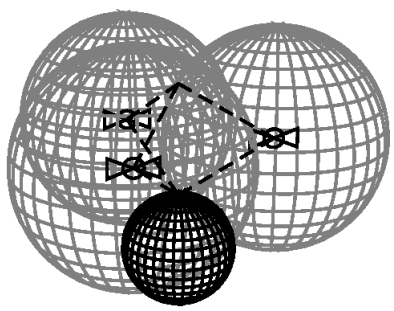


Figure 2.4: Using the earth as the fourth space

2.6 Hull configuration

Boats are designed with a broad range of different hull configurations, every configuration have different advantages and disadvantages. There are two main types of hull configuration, either planning- or displacement hulls. Planning hulls will result in the boat riding on top of the water, while displacement hulls will displace the water.

Larger boats like freighters or cruise-ships are moving lower in the water due to their size and weight, therefore are displacement hulls frequently used. Smaller and faster boats such as power-boats will float on-top of the water as a result of their size, weight and high speed. Thus, they

frequently have planning hulls.

Four common hull configurations with their corresponding advantages and disadvantages are listed in table 2.2. [8]

Table 2.2: The advantages and disadvantages of different hull configurations

Hull Shapes	Advantages	Disadvantages
Flat-Bottomed	Very stable on calm bodies of water.	Rides roughly in choppy waters.
Round-Bottomed	Gives a smoother ride than a flat-bottomed hull in rough water.	Takes more power to move at the same speed as flat-bottomed hulls. May roll or bank in sharp turns.
Catamaran	Has greater stability because of the wide hull. Good fuel efficiency at high speeds, and excess stability allows equipment to be mounted high above the water line.	Are more susceptible to pitch and heave responses. Sensitive to changes in weight and position of the cargo. Needs a large area when turning.
SWATH	Similar to the catamaran, but has a better ability to withstand rough conditions.	Similar to the catamaran.

Figure 2.5: Flat-Bottomed Hull

Figure 2.6: Round-bottomed Hull

Figure 2.7: SWATH Hull

Figure 2.8: Catamaran Hull

2.7 Thruster configuration for Catamaran

This chapter will cover the theory behind the "X" holonomic thruster configuration. This thruster configuration involves having four thrusters mounted in an fixed angle of **45°** on the centerline of the hull as illustrated in figure 2.9. Figure 2.9 also displays the necessary thruster motion to move the vessel in any direction.

Holonomic refers to the relationship between controllable and total degrees of freedom of a entity. If the controllable degree of freedom is equal to total degrees of freedom, then the entity is said to be Holonomic. Thus, by having the four thrusters at a **45°** the vessel becomes holonomic. [9]

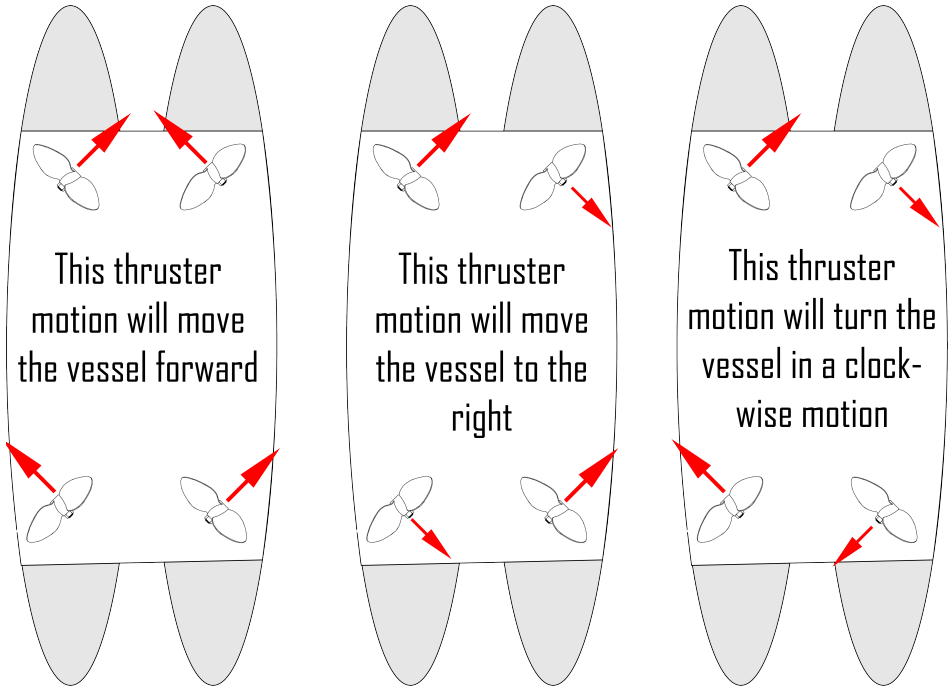


Figure 2.9: Motion of thrusters in X-Configuration

2.8 Computer Vision

2.8.1 Graph-based SLAM

Real-Time Appearance Based mapping is a RGB-D, Stereo and Lidar Graph-based SLAM approach based on an incremental appearance-based loop closure detector. The loop closure detector uses a bag-of-words approach to determinate how likely a new image comes from a previous detected location or a new location. Whenever a loop closure is accepted, a new constraint is added to the map's graph, then a graph optimizer minimizes the errors in the map. A memory management approach is used to limit the number of locations used for loop closure detection and graph optimization.

The underlying structure of the map is a graph with nodes and links. The nodes save odometry poses for each location in the map. The nodes also contain visualization information like laser scans, RGB images, depth images and visual words which is used for loop closure detection. There are two types of links: neighbour and loop closure. Neighbour links are added between the current and the previous nodes with their odometry transformation. Loop closure links are added when a loop closure detection is found between the current node and one from the same

or previous maps.

The Loop closure detection is the process of finding a match between the current and a previously visited location in SLAM. The underlying idea behind these approaches is that loop closure detection is done by comparing all previous images with the new one. When loop closures are found between the maps, a global graph can be created by combining the graphs from each session.

2.8.2 Stereo Camera

This chapter will cover the basics of stereoscopic vision, which in this case can be used for measuring depth. Regular webcams offer streams of RGB or HSV data, which can be used for object recognition and tracking, but this alone is not enough for identifying the dimension and depth of an object.[10]

A stereo camera consists of two monocular camera lenses. The two lenses on the stereo camera is separated approximate the same distance as human eyes. The depth and motion can be calculated by comparing the displacement of pixels between the two images.

Depth estimation

This chapter will explain the math behind depth estimation, the source of this is an lecture held by Robert Collins at the CSE Department, Penn State University, Pennsylvania. [11]
Figure 2.10 shows two frames that represents each frame of the two lenses of an stereo camera. The right lens is shifted by $\boldsymbol{T_x}$ units along the X axis. Otherwise, the two lenses are identical with the same orientation and focal lengths. The distance between the two lenses, $\boldsymbol{T_x}$, is also known as the baseline.

Figure 2.11 shows an top down view of the system. Here we have a few variables; $\boldsymbol{f}$ as the focal length, $\boldsymbol{Z}$ as the distance to the object and $\boldsymbol{P}$ as the point of the object.

The image coordinate of the point $\boldsymbol{P}$ in the left frame is calculated by:

$$\boldsymbol{x_l} = \boldsymbol{f}\frac{\boldsymbol{X}}{\boldsymbol{Z}} \text{ and } \boldsymbol{y_l} = \boldsymbol{f}\frac{\boldsymbol{Y}}{\boldsymbol{Z}} \tag{2.15}$$

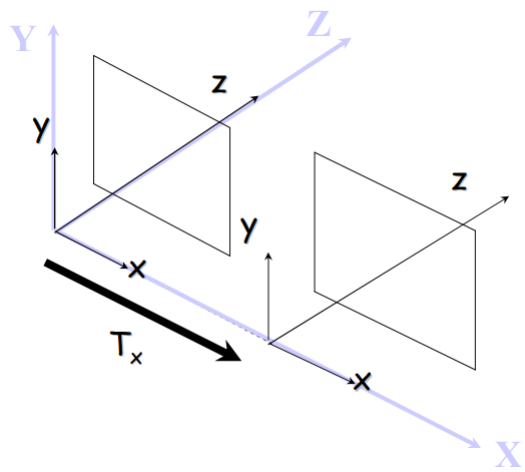


Figure 2.10: Stereo camera system

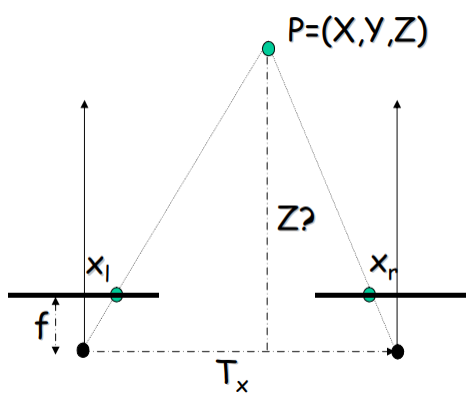


Figure 2.11: Stereo camera system

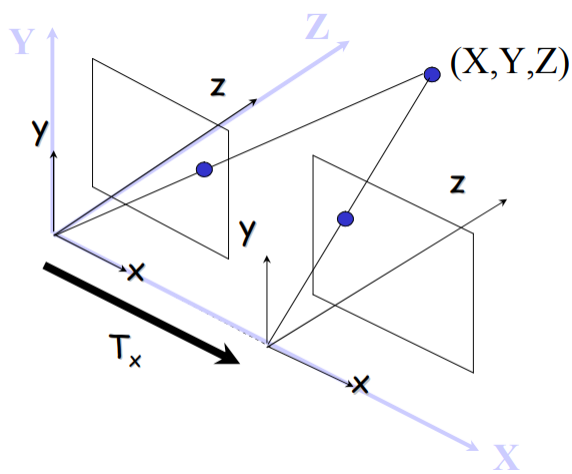


Figure 2.12: Stereo camera system included the point of interest

And the image coordinate of the point $\boldsymbol{P}$ in the left frame is calculated by:

$$\boldsymbol{x_r} = \boldsymbol{f}\frac{\boldsymbol{X} - \boldsymbol{T_x}}{\boldsymbol{Z}} \text{ and } \boldsymbol{y_r} = \boldsymbol{f}\frac{\boldsymbol{Y}}{\boldsymbol{Z}} \tag{2.16}$$

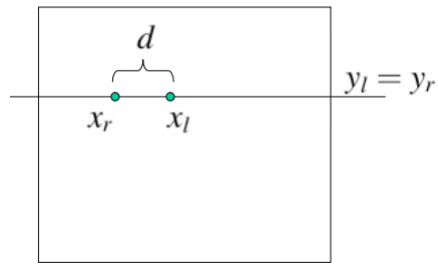


Figure 2.13: Disparity of the point

Thus, the stereo disparity, as seen in figure 2.13 is given by

$$\boldsymbol{d} = \boldsymbol{x_l} - \boldsymbol{x_r} = \boldsymbol{f}\frac{\boldsymbol{X}}{\boldsymbol{Z}} - (\boldsymbol{f}\frac{\boldsymbol{X}}{\boldsymbol{Z}} - \boldsymbol{f}\frac{\boldsymbol{T_x}}{\boldsymbol{Z}}) = \boldsymbol{f}\frac{\boldsymbol{T_x}}{\boldsymbol{Z}} \tag{2.17}$$

and the depth of a point is given by

$$\boldsymbol{Z} = \boldsymbol{f}\frac{\boldsymbol{T_x}}{\boldsymbol{d}} \rightarrow \text{Depth} = \text{Focal lenght} \times \frac{\text{Baseline}}{\text{Disparity}} \tag{2.18}$$

2.9 Data-collection and -annotation

Data collection is the process of gathering information from different sources. The different types of data can be either numeric, categorical, or even pictures. Because the use of data collection in this project are images, this will be the main topic further in this chapter.

Image annotation is the human-powered task of associating an entire image, or a section of an image, with an identifier label. These labels are predetermined and are chosen to give the computer trained model information about what is included in the image.

Five common image annotation services are as follows: [12]

- **Bounding Boxes:** With bounding boxes the annotators must draw a box that includes the object they want to annotate within the image. This can be done in both 2D and 3D.
- **Image Classification:** Image classification is the process of associating an entire image with just one label. This is used when the picture includes only the object, with minimal background or other objects involved.
- **Lines and splines** Lines and splines annotation is the labeling of straight or curved lines on images. This is used for annotating lanes, sidewalks, power lines amongst others.
- **Polygons:** Polygon annotation allows annotators to plot points on each vertex of the target object. This annotation method allows all of the object's exact edges to be annotated, regardless of its shape.
- **Semantic Segmentation:** Semantic segmentation is the annotation of every pixel within an image. Instead of marking the object using bounding lines, each pixel of the given object will be annotated.

2.10 Deep Learning

Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans. In deep learning, a computer models can achieve state-of-the-art accuracy, sometimes exceeding human-level performance. Models are trained by using a large set of labeled data and neural network architectures that contain many layers.

Deep learning is a specialized form of machine learning. A machine learning workflow starts with relevant features being manually extracted from images. The features are then used to create a model that categorizes the objects in the images. Deep learning performs "end-to-end learning", where the network is given raw data and a task to perform, such as classification, and it learns how to do this automatically.

Deep learning algorithms scale with data, where shallow learning converges. The advantage of deep learning networks is that they often continue to improve as the size of the data increases. There are three common ways to create and train deep learning models:

- **Training from scratch:** To be able to train a deep network from scratch, the network needs a a lot labeled data and the network architecture needs to be designed to learn the features. This is a good approach for network that will have large number of outputs. Because the dataset has be very large, this approach will typically take days or weeks to train.
- **Transfer Learning:** Transfer Learning is a process that involves fine-tuning a pre-trained network.The benefits of using Transfer Learning is that the model needs less amount of data. This reduces the training time to hours instead of days.
- **Feature Extraction:** Uses the network as a feature extractor. Since all the layers are tasked with learning certain features from images, the features can be pulled out of the network at any time during the training process. These features can be used as input to a machine learning model such as support vector machines. [13]

2.10.1 YOLO - You Only Look Once

You only look once is an object detection system targeted for real-time processing. It takes the entire image in a single instance and predicts the bounding box coordinates and class probabilities for these boxes.

YOLO divides the input image into a SxS grid, where each grid cell predicts B bounding boxes together with their confidence score. Each confidence score reflects the probability of the predicted box containing an object pr(Object), as well as how accurate the predicted box is by evaluating the overlap with the ground truth bounding box measured by intersection over union. Hence the confidence is defined as:

$$\text{Pr(Object)} \cdot \text{IoU}_{\text{pred}}^{\text{truth}} \tag{2.19}$$

If no object exists in that cell, the confidence score should be zero. Otherwise the score equal the intersection over union (IoU) between the predicted box and the ground truth.

Each bounding box consists of five predictions: x ,y ,w ,h , *confidence*. The (x,y) coordinates represent the center of the box relative to the bounds of the grid cell. The width and height are predicted relative to the whole image. The confidence represents the IoU between the predicted

box and any ground truth box. Each grid cell also predicts C conditional class probabilities, $\Pr(\text{Class}_i|\text{Object})$. These probabilities are conditioned on the grid cell containing an object. The network is implemented as a convolutional neural network (CNN) and evaluate it on the dataset of the different buoys. The initial convolutional layers of the network extract features from the image while the fully connected layers predict the output probabilities and coordinates. The network has 24 convolutional layers followed by two fully connected layers.[14]

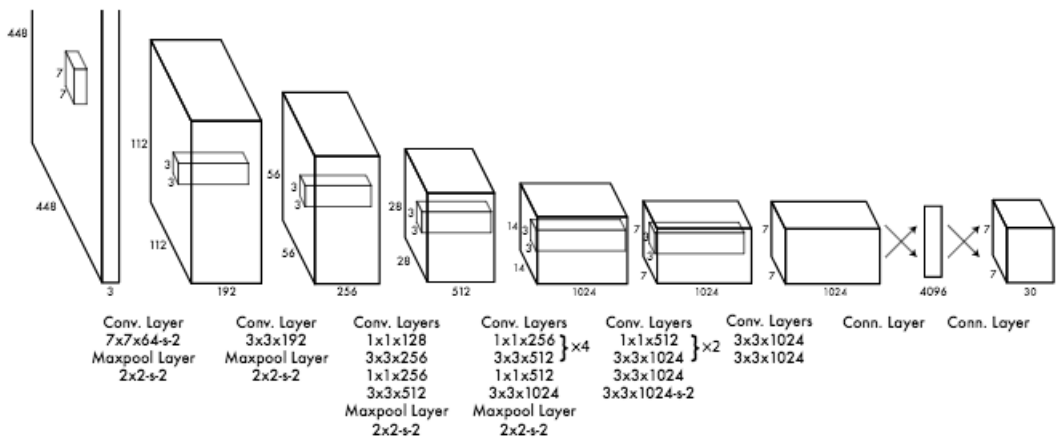


Figure 2.14: YOLO network Architecture

YOLO predicts multiple bounding boxes per grid cell. To compute the loss for the true positive, the model only wants one of them to be responsible for the object. For this purpose, the one with the highest IoU with the ground truth is selected. This strategy leads to specialization among the bounding box predictions.

YOLO uses sum-squared error between the predictions and the ground truth to calculate loss. The loss function composes of the classification loss, the localization loss and the confidence loss.

If an object is detected, the classification loss at each cell is the squared error of the class conditional probabilities for each class:

$$\sum_{i=0}^{S^2} \mathbb{1}_i^{obj} \sum_{c \in class} (p_i(c) - \hat{p}(c))^2 \tag{2.20}$$

where $\mathbb{1}_{i}^{obj} = \mathbf{1}$, if an object appears in cell i, otherwise 0.

$\hat{p}(c)$ denotes the conditional class probability for class c in cell i

The localization loss measures the errors in the predicted boundary box locations and sizes. The localization loss only count the box responsible for detecting the object.

$$\begin{aligned} &\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{i}^{obj} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\ &+ \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \end{aligned} \tag{2.21}$$

where $\mathbb{1}_{i}^{obj} = \mathbf{1}$, if an object appears in cell i, otherwise 0

λ_{coord} increase the weight for the loss in the boundary box coordinates

If an object is detected in the box, the confidence loss is:

$$\sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left(C_i - \hat{C}_i \right)^2 \tag{2.22}$$

if an object is not detected in the box, the confidence loss is:

$$\lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{noobj} \left(C_i - \hat{C}_i \right)^2 \tag{2.23}$$

where $\mathbb{1}_{ij}^{noobj}$ is the compliment of $\mathbb{1}_{ij}^{obj}$

$\hat{C}_i$ is the box confidence score of the box j in cell i

λ_{noobj} weights down the loss when detecting background

The final loss adds localization, confidence and classification losses together: [15]

$$
\begin{aligned}
&\lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_i^{obj} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{noobj} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_i^{obj} \sum_{c \in class} (p_i(c) - \hat{p}(c))^2
\end{aligned}
\tag{2.24}
$$

2.10.2 Mean Average Precision

The performance of the buoy detection algorithm will be judged by mAP criterium defined in the PASCAL VOC 2012 competition. In this validation of the object detection, the IoU has to be over 60% instead of 50% as in the PASCAL VOC 2012 competition.

For each class the AP will be calculated. The neural net detection-results are sorted by decreasing confidence and are assigned to ground-truth objects. Its a "match" when they share same label and the intersection over Union ≥ 60%. The "match" is considered a true positive if the ground-truth object has not been already used, which is to avoid multiple detection of the same object. [16]

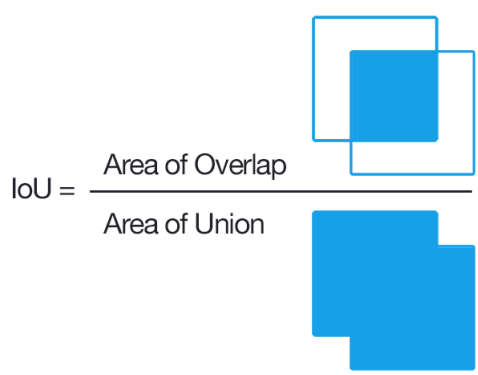


Figure 2.15: IoU Formula

Using this criterium, the precision/recall curve can be calculated:

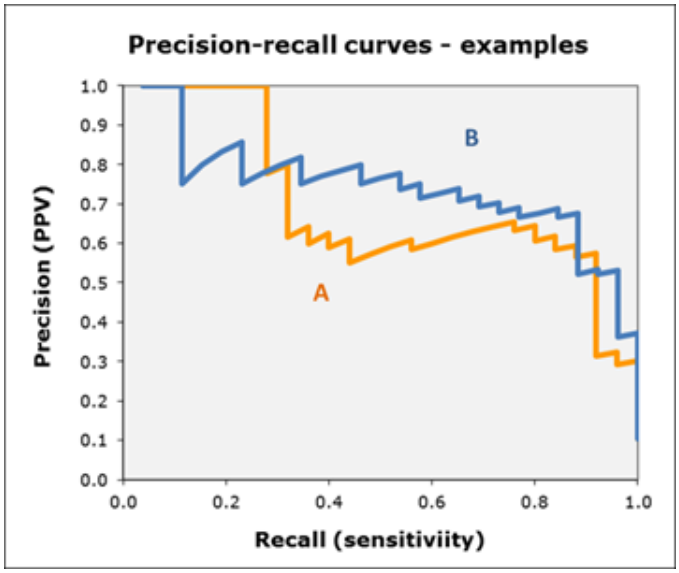


Figure 2.16: Precision-Recall curve

Then a new version of the measured precision/recall curve is computed with **precision monotonically decreasing** by setting the precision for recall r to the maximum precision obtained for any recall $r' > r$.

Finally, the AP is computed as the **area under this curve** by numerical integration. No approximation is involved since the curve is piece-wise constant.

2.11 Extended Kalman Filter

Extended Kalman Filter was introduced to solve the problem of non-linearity in Kalman filter. In real life there may be a lot of scenarios where the system may look in one direction and may take the measurement from another direction. Resulting in non-linear function which when fed to a Gaussian resulted in a non-Gaussian distribution. [17]

An Extended Kalman Filter is a filter that linearizes about the current mean and covariance of a non-linear stochastic difference equation. With EKF its possible to linearize the estimation around the current estimate using the partial derivatives of the process and measurement functions to compute estimates even in the face on non-linear relationships. [18]

Assume there is a closed-form expression for the predicted state as a function of the previous state, controls, noise and time.

$$\boldsymbol{x_{k+1}} = \boldsymbol{f}(\boldsymbol{x_k}, \boldsymbol{u_k}, \boldsymbol{w_k}, \boldsymbol{t}) \tag{2.25}$$

The Jacobian of the predicted state with respect to the previous state is

$$\boldsymbol{f}^{(x)} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} \tag{2.26}$$

The Jacobian of the predicted state with respect to the noise is

$$\boldsymbol{F}^{(w)} = \frac{\partial \boldsymbol{f}}{\partial \boldsymbol{w_i}} \tag{2.27}$$

These functions take simpler forms when the noise eenters linearly into the state update equation:

$$\boldsymbol{x_{k+1}} = \boldsymbol{f}(\boldsymbol{x_k}, \boldsymbol{u_k}, \boldsymbol{t}) + \boldsymbol{w_k} \tag{2.28}$$

In the EKF, the measurement can be a nonlinear function of the state and the measurement noise.

$$\boldsymbol{z_k} = \boldsymbol{h}(\boldsymbol{x_k}, \boldsymbol{v_k}, \boldsymbol{t}) \tag{2.29}$$

The Jacobian of the measurement with respect to the state is

$$\boldsymbol{H}^{(x)} = \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{x}} \tag{2.30}$$

The Jacobian of the measurement with respect to the measurement noise is

$$\boldsymbol{H}^{(v)} = \frac{\partial \boldsymbol{h}}{\partial \boldsymbol{v}} \tag{2.31}$$

These functions take simpler forms when the noise enters linearly into the measurement equation:

$$\boldsymbol{z_k} = \boldsymbol{h}(\boldsymbol{x_k}, \boldsymbol{t}) + \boldsymbol{v_k} \tag{2.32}$$

The EKF loop is almost identical to the linear Kalman filter except that:

- the exact nonlinear state update and measurement functions are used whenever possible and the state transistion matrix is replaced by the state Jacobian.
- The measurement matrices are replaced by the appropriate Jacobians. [19]

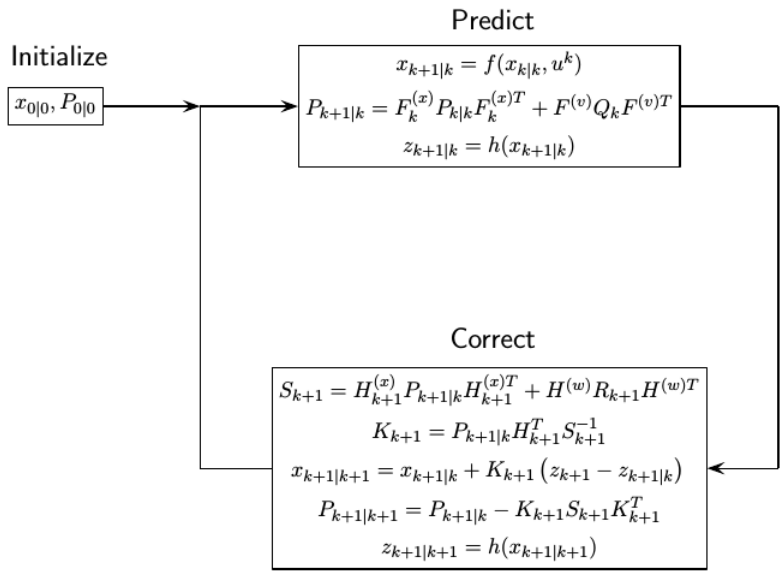


Figure 2.17: Extended Kalman Filter loop [3]

[3]

Chapter 3

Materials and methods

This chapter contains materials and methods and how they are presented in the development of this thesis.

3.1 Project Organisation

The group consists of four members where one is elected project manager and another is elected secretary. These roles have extra responsibilities in addition to being a project developer. The group agreed that even though there is a project manager, the decision making is collective and the power balance between project developer and manager is equal. Even though it's called "project manager" it would be more accurate to call it project structure facilitator and relate it to a scrum master in this case.

The project manager has the responsibility to ensure progression by facilitating a consistent project structure. The project manager is also responsible for creating such a structure and making sure that the group members utilizes it. In the case of this project, the project manager decided in conjunction with the group members on a structure where stand-up meetings are conducted both at the start and end of each workday to get the group members in the habit of reporting their current workload, obstacles and future tasks with each other. This provides a uniform overview of the day to day projects development status to group members. In addition, it gives the possibility of discussing and solving problems related to obstacles from individual

workloads.

To get a greater view of the overall progression of the project development, the project manager is responsible for conducting progress meetings every Friday. Progress meetings are similar to stand-up meetings in the way each member presents their current workload, obstacles and future tasks, however, in the progress meeting, these points are described with work from the whole week and not just the current or past day in particular. All information about current progress will be written down by the secretary into a progress report and sent to all supervisors.

Early on in in the project, a partnership was established with a group of students from Naval Architecture, faculty of engineering to help build the fleets hull. Their team wanted to promote the idea of NTNU students participating in the international competition RoboBoat in Florida, USA. In addition, they had ambitions of competing in the Norwegian competition AutoDrone. Both parts found it helpful to trade knowledge and experience in order to create the best possible fleet. Therefore, it was agreed upon conducting progress meetings each Friday. These meetings were meant for both parts to present and discuss weekly progress. In addition, the partners should agree on the way to move forward in order to achieve the highest cohesiveness and synergy as possible. Unfortunately, due to the COVID-19 crisis, the plans of creating a physical fleet changed and the group had to liquidate these meetings around 11.04.2020. The team from Naval Architecture derived their focus onto upcoming exams and the group did not see the need for more meetings due to the change of scope in the new problem description mentioned in chapter 3.2 revolving the deviation management.

3.2 Deviation management

"An ASV may be developed to manoeuvre through each of the four courses mentioned in chapter 1.2 without problems. In addition, the ASV will satisfy the given parameters of each course and satisfy the four score-sheets from chapter 1.2 in order to harvest the most points available."

This was the original problem description defined in our pre-project report. However, the COVID-

19 crisis escalated in the time-frame of the development phase in the bachelor project. This resulted in the school shutting down and students like lost the opportunity to use the school's premises for development. In other words, group members cannot develop a physical construction. The original plan was to develop a hull in cooperation with a group of partners from the ship design masters program at our university. The bachelor group would then provide software and electronic hardware for the fleet in order to representing NTNU in the international competition RoboBoat and the Norwegian competition AutoDrone. As the crisis broke out and hull development staggered, development of software using electronic hardware became constrained. The bachelors group scattered across the country to get back with family during this period and are therefor working remotely from home. This constrains development of software using hardware like motor control, optical object detection etc. because the group only have one member in Aalesund where most of the hardware is located.

However, this does not mean that the project gets discarded, there must be a change in the scope of the project and redefine the problem description in order to fulfil the requirements of a bachelor's project.

As mentioned in the deviation management plan in the pre-project appendixA attachment the group members gathered in an extraordinary meeting facilitated by the group leader. In this meeting the group concluded that in order to achieve the closest results to the original problem description the group would create a simulation of the competition and develop the fleet as a digital-twin instead of a physical construct. Then the group would showcase the twins runs in the simulated competition alongside the programs the fleet uses to manoeuvre these runs. In addition, the group would develop as much as possible of the intended components for the fleet like for example object recognition and room mapping. The original components would have plan descriptions regarding how to set them up and use for a later project. the bachelor group also concluded that it was wanted to deliberately develop for later bachelor or student projects to have a platform for further development of such project. As the bachelor groups intentions was to represent the university in both the national and international competition, the group would like to help later groups as much as possible because the competition most likely will be cancelled this year.

3.3 Gantt-diagram work description

This section will describe more generally what each phase in the gantt diagram attached B consists of.

3.3.1 Pre-project

In the pre-project phase, the group made several decisions on how the group will function for the period of the bachelor's project. Responsibilities and roles such as project manager, secretary, and developer were declared. In addition, it was agreed upon what premises would be used alongside defining what resources the project was provided with. Lastly for the work form, group work norms, attitude, and cooperation rules were decided.

The project problem description was defined in conjunction with what problems the group could assume to be dealing with, how the group would plan out and execute development, gather information, assess risk, and predict desired progression. Other declarations like technical documentation, meeting form, frequency of meetings and how progress will be tracked was also decided on. Lastly, the group decided how project deviation will be managed and what equipment and requirements the implementation will need.

3.3.2 Research

As shown in the Gantt-diagram in attachment B the research phase unfolded somewhat parallel to the pre-project and planning design phase. The research phase was used to establish a basis for designs and plans. This phase was started by researching existing solutions such as the Kongsberg Groups USV Sounder and GeoSwath to get a basic overview of the current state of USV/ASVs. Simultaneously, different work methodologies like Scrum, Gantt and Kanban were researched. After gaining a greater understanding of the components needed to establish a competitive fleet, it looked promising to look further into software and hardware solutions such as OpenCV, ROS and Pixhawk. It was decided that it's possible to create a fleet gathering these solutions, therefore, research responsibilities were divided in a way where two people would look into the same topic if it had a major role in the development phase such as ROS and OpenCV.

Whereas, if the topic had a smaller role like Pixhawk, only one person had the responsibility to research and develop.

3.3.3 Planning Design

In this phase an MVP-description was made using confluence, this MVP-description is attached at E. After defining what the minimal value product was, the group started designing software and hardware diagrams that would build upon it. By using the MVP-description in conjunction with designs as a basis, the group started filling up a Scrum backlog. In addition, test diagrams that would describe what parameters the product will be measured on was created.

3.3.4 Software

The software phase was meant to develop all the software needed to complete the fleet. This phase was started by establishing a remote connection from the fleet to a laptop. However, this did not get finished because of the change in priorities that happened around the COVID-19 crisis. Development on computer vision and object detection continued as planned because it did not need any hardware except a camera. Therefor, this phase includes setting up a test simulation in both RDS and using VRX.

3.3.5 Hardware

This phase was meant to cover all aspects of the project regarding hardware such as weight budget, ordering of parts etc.

3.3.6 Testing

This phase was oriented around having a physical fleet, therefore, alot of the sub tasks is marked as unfinished in the gantt-diagram attachment. Mainly, this phase was meant for testing the ASV in different environments. In addition, it would cover all of the software testing aspects.

3.4 Materials

3.4.1 Catamaran Hull

The choice of hull for the ASV ended up being the catamaran, an multi-hulled design that have two instead of just one hull. This will help by making the vessel stable and thus, the vessel will behave in a smoother way at sea. This will be on the positive spectre for the vessels sensors and camera, and will amongst other prevent too much motion-blur.

Figure 3.1: Catamaran

3.4.2 "X" Holonomic thruster calibration

The choice of thruster calibration ended up being the "X" Holonomic. This will make the controll of the catamaran much easier and will allow it to move in any direction. Thus, the process of turning fast after the speed gate task, or going sideways into the dock will be much easier to solve. The theory behind this can be found in chapter 2.7

3.4.3 T200 Thruster

The T200 Thruster is one of the world's most popular underwater thruster for ROVs, AUVs and surface vessels. The T200 Thruster uses an underwater thruster design consisting of a fully-flooded brushless motor with encapsulated motor windings and stator as well as coated mag-

nets and rotor. The design of the thruster is compact and has the minimal number of parts, which helps reduce both the price and weight of the thruster.

The core motor design is a three-phase brushless outrunner motor that has been optimized for underwater use. A sensorless brushless electronic speed controller (ESC) is required to run the thruster in all situations. It's optimized to run at a voltage of 16v (such as a 4s lithium-ion battery pack), but can run at a range of voltages.

In addition to the thruster, it is required to have an speed controller like an ESC, a power source, and a signal source, an RC radio receiver or a microcontroller for as an controll center. "BlueRobotics", the provider of the T200 thruster, also provides a wealth of available technical information, charts, code examples, and tutorials on their websites and forums.

The T200 thruster will provide a maximum thrust of 5.25N forward direction and 4.1N reverse direction. [20]

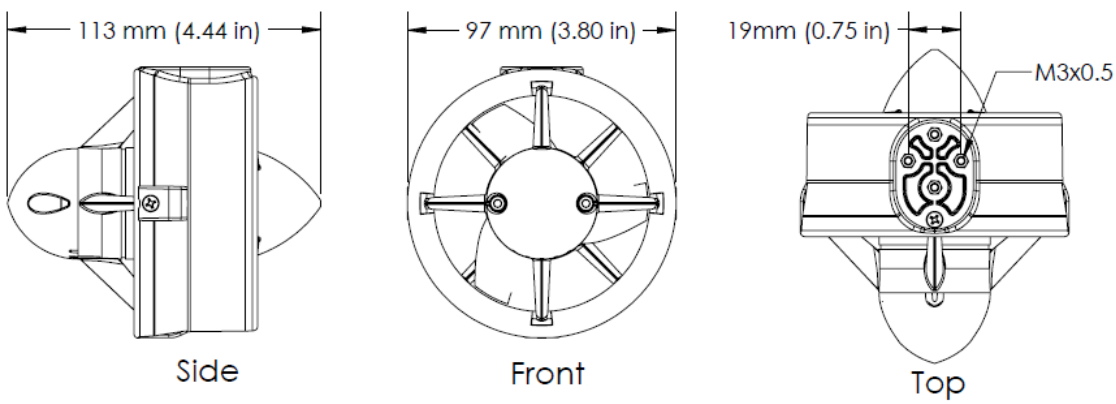


Figure 3.2: 2D Drawing of the T200 Thruster

3.4.4 Nvidia Jetson TX2

Nvidia Jetson TX2 is a high-performance single board computer and is mostly used for deploying vision and neural net processing onboard mobile platforms like drones, autonomous robots and surface vehicles. Jetson TX2 is good for application that require a lot of processing power on a significant power budget. Because many developers are using Jetson, it will simplify troubleshooting since a lot of information is available online.

Since the ASV needs to process data in real time with an AI model, it's important that the computing device is mobile and quickly render high-resolution images and videos concurrently. Because of the Pascal GPU the TX2 is delivering, the ASV can handle real time object detection with good results.

The TX2 Developer Kit has a price of 399,00 USD which is more expensive than other controllers like Raspberry Pi 4, but since the ASV requires good system specifications, which the raspberry Pi 4 can't deliver, the price is justified.

The Jetson is compatible with ROS, which is the operating system the ASV will operate with. ROS is also easily implemented to the Nvidia Jetson TX2. The developner kit can be seen in figure 3.3

Figure 3.3: Jetson TX2 developer kit

3.4.5 Lithium-ion Battery

For the power supply the ASV will use a Li-4S-18Ah (14.8V, 18Ah) battery from Blue Robotics. It is built with 4 Samsung 18650 30Q battery in series and 6 in parallel. The battery has a nominal capacity of 18.0Ah which gives the battery a lifetime of approximated 4 hours of continuous

moderate use, Which gives plenty of time to test the ASV, and competiting in the competition. The battery can tolerate a 90A continuous discharge and a 132A burst discharge. The lithium-ion cells in the battery has a high tolerance for accidental mishandling and good performance characteristics. The weight of the battery is 1152g, which is light weighted and fits great in the weight requirements of the competition. The battery is CE approved and certified.

3.4.6 Zed 2 Stereo Camera

ZED 2 is a stereo camera that has, amongst others, the following specifications:

- Neural Depth sensing
- Built-in IMU, Barometer & Magnetometer
- 120° Wide-Angle Field of View

The ZED 2 camera uses neural networks to reproduce human vision. This is done by having two lenses separated at approximate the same distance as human eyes, which allow it to capture the depth and motion of an object, as talked about in chapter 2.8.2. The lenses have an depth-FOV of 110° x 70° and a depth-range of 0.2 to 20 meters.

The camera comes with a built-in IMU, barometer and magnetometer which saves both space and weight of the vessel. The camera lens have an wide angle field of view of 120° that gives the vessel the possibility to detect objects further to the sides than with the use of a normal camera. The integrated software of the camera also includes possibilities for object detection, positional tracking and cloud connection.

Figure 3.4: Dimensions of the ZED 2 Camera

3.4.7 Pixhawk mRo controller

Pixhawk mRo is one of many open hardware options available for open-source autopilot. The benefits of using this controller is that it provides support to determine the vehicle state. The Pixhawk minimally requires a gyroscope, accelerometer, magnetometer and a barometer to determine the state. GPS or another type of position system is needed to controll the ASV autonomous.

Pixhawk has two type of flight modes: autonoumous and manual, which define how to autopilot responds to remote control input. Where a RC-controller or a joystick is needed to control the ASV manually, and GPS for autonomous navigation.

The Pixhawk mRo controller supports several different sensors, ESC and motors and power supplies. The benefits of the Pixhawk is that it's easy to implement sensors and motors. The Pixhawk is also compatitible with ROS using MAVLink through a MavROS topic.

Figure 3.5: Pixhawk mRo controller

3.5 Software

3.5.1 List of software used

- Overleaf
- Instagantt
- Pycharm
- Draw.io
- Microsoft Excel
- Microsoft Word
- ROS Development Studio
- UIALS Confluence

3.5.2 ROS with Linux Ubuntu

The following section will consist of justifications to why the group decided on using ROS in conjunction with Linux Ubuntu.

ROS has a large community of developers, if a problem rises, it is likely that someone else have had the same or a similar issue. Therefore, the group might find a solution or guidance online. This will help to keep the projects progression up to pace by reduce time spent on problem solving. Also, it might help get a better-quality solution than the project developers would come up with.

As a result of a large community there is of course a lot of packages/libraries that is highly relevant for this project. It seems that the group would miss out on a lot of pre-written code if the group did not use ROS and therefore waste time. For example, the vessel need to map the water surface in order to plan a path. As there is already a package for surface mapping called Gmapping.

It has good support for troubleshooting and system visualization. This should reduce time spent

on error correction and calibration. For example, using tools like rqt_graph and rqt_plot will help getting a mental picture of the system and how different components interact. There are several other tools like rvis, rqt_bag etc. that aids the same purpose.

For a project like this, it is helpful to be able to simulate solutions/features continuously, having to set up a vessel, get down to the docks and start testing for every newly added feature would be tedious. ROS provides a simulator called Gazebo which can help with this issue. It's possible to model the vessel with physical and non-physical components like sensors and software in order to test new features without being worried about time- or money squandering. During the research phase the group noticed that almost all earlier contestants have used ROS as well.

In addition, it has good support for Python which is the language our group has most experience with.

However, ROS in conjunction with Linux is very complex and can be overwhelming without Linux experience. The group has a mix between some experience in Linux and none at all, this results in a steep learning curve and demanding workloads. It may take some time before the group can take advantage of the benefits that ROS provides.

3.5.3 Computer vision using OpenCV

The following section will consist of the reasoning behind the use of OpenCV. OpenCV is an open source computer vision and machine learning software library. OpenCV was built to provide a common infrastructure for computer vision applications and to accelerate the use of machine perception in the commercial products. This library has more than 2500 optimized algorithms, which includes a comprehensive set of both classic and state-of-the-art computer vision and machine learning algorithms. OpenCV also has more than 47 thousand people of user community and estimated number of downloads exceeding 18 million. [21]

Thus, by taking this into consideration, both troubleshooting and search for existing solutions becomes less tedious. This will help in increasing the efficiency of the project progress by reducing time spent on problem solving.

3.5.4 Data-collection and -annotation

The following section will consist of the reasoning behind using the ZED 2 camera with OpenCV in pyCharm for data-collection and bounding boxes for annotation. The ZED 2 camera have two lenses instead of just one, thus by using this the data-collection time got halved. By making a script in pyCharm that used the input feed from the camera, images including the object of interest were saved with an corresponding file name with a keypress event. Thus, image files with the name red1, green1 and yellow 1 were saved in a data collection folder.

The next step of this process were to annotate each object in a given image, this would have to be done manually. Since the images included the three different types of the bouye while being buoyant, the most efficient annotation method would be by using 2D bounding boxes. Thus, by using the annotation software LabelImg [22] an corresponding XML file including the image filename, and four points identifying the box revolving the object were given. This XML file would later be used alongside the image file in the trained model using YOLO in chapter 3.5.5.

3.5.5 Deep learning with YOLO

To be able to detect buoys, YOLO will be implemented. The benefits of using YOLO is that it achieves high accuracy while also being able to run in real-time. YOLO is fast compared to other object detection algorithms like R-CNN. Darkflow has been implemented to run YOLO for the buoy detection. Darkflow is a network builder adapter from Darknet, which allows TensorFlow network from cfg files and pre-trained weights to be used. Darkflow is also compatible with Windows 10 and Linux. YOLO will need a lot of data for training to increase accuracy of the buoy detection. Images will be labeled using a annotation tool. YOLO uses a Transfer Learning approach to learn detection of new objects. YOLO saves a checkpoint for every 100th iteration, so it's possible to choose what checkpoint of the training that will be used for the buoy detection.

3.6 Simulation and visualization

A simulation for the environment has been set up using ROS Melodic and Gazebo 9. The environment and models in the simulation is based on the Virtual RobotX (VRX) simulation created by Open Source Robotics Foundation (OSRF). The simulation have implemented physics for wind, waves and water current. The physical USV maneuvering model in the simulation are based on the same theory presented in 2.4. The waves is created using three independent superimposed Gerstner wave functions [23], where the amplitude, period and direction can be user-specified. The theory behind the Gazebo USV plugins and its sources in the simulation can be found in [24].

3.6.1 Gazebo with ROS

Gazebo is a 3D multi-robot simulation program which are used by hundreds of thousands of users and developers. With Gazebo the user can create a 3D world with robots, obstacles and a lot of other objects. The program makes it simple to make virtual scenarios without having a realistic robot. Creating the course in Gazebo, it allows testing the software in a safe environment, without risking damaging the ASV. Rviz are able to display the mapped area with both lasers and stereo cameras, and able to locate the position of the robot in the area.

3.6.2 Rviz with ROS

Rviz is perfect for figuring out what went wrong with the ASV's vision system, the ROS package provides a lot of different visual information which the user can disable and enable at any time.

It also provides 3D visualization with interactive markers. That means while the program is running, its acceptable for environment changes, like ignoring or displaying regions.

The package can run multiple vision processes at the same time. Which makes it easier to display everything that the ASV sees. the package can be used to display data from Gazebo or the real world.

3.6.3 USV model and sensor package

A USV simulation model has been created based on Marine Advanced Robotics WAM-V 14 USV [25]. The model comes with configurable propulsion configurations:

- H - differential thrust with fixed angles
- Differential thrust with adjustable angles (Azimuth)
- T - differential thrust with Lateral/Bow
- X - Holonomic 2.7

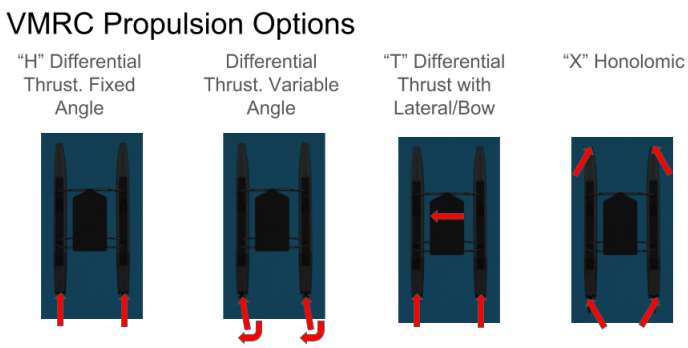


Figure 3.6: Propulsion options

Different available sensors from VRX source:

- Camera
- Stereo Camera
- GPS
- Lidar (16- and 32 beam)
- IMU
- P3D ground truth sensor
- Acoustic Pinger

additional sensors can be manually added using a URDF file.

3.7 ROS Development Studio

ROS Development Studio (ROSDS) is a online development Studio for ROS applications. This website allows programming any ROS-based robots in real-time using an online simulator. This application is already pre-configured with all typical ROS packages and other ROS packages can easily be implemented. It also provides pre-configured tools like Gazebo simulator, Linux Shell, Jupyter notebooks and graphical tools like Rviz.

ROSDS also provides support for testing robots in the simulator, for then connecting the environment to the real robot. Since the service is web based, it's possible to access the environment from any type of computer. Different types of mapping, localization and navigation packages were tested before the decision were made. Guides and tutorials for ROSDS is also provided and are regularly updated with new guides and videos. [26]

3.8 Navigation

ROS navigation stack is using information from odometry, sensor streams and goal pose to output safe velocity commands. The navigation stack require a map of the environment containing free maneuvering space and obstacles to calculate a safe path. RTABMap will be used to publish a map. RTABMap mapping at sea entail several issues to be solved. It is a limited amount of objects to verify location, distance and speed. The water surface is reflective, it can be the cause of noise. Marine vessels is constantly affected by waves, wind, water current and the Coriolis force. The vessels motion is in 6DOF. Floating object such as buoys will not be completely static, this leading to issues with finding similarities between two arbitrary points.

RGB-D

The first alternative is to use image data in RGB-D format and fuse the information with a 3D lidar pointcloud. The image data is required to be in RGB-D format in order to fuse with the 3D lidar for loop-closure detection in RTABMap. A disparity image can be computed using the `stereo_image_proc` node from `image_pipeline` package [27], which is implementing OpenCVs block matching algorithm [28]. Then a depth image can be created using the

`rtabmap_ros/disparity_to_depth` nodelet. The raw stereo image is then synchronised into a single file with depth image using the `rtabmap_ros/rgbd_sync` nodelet. RTABMap uses the synchronised RGB-D image together with lidar point-cloud and odometry information from the `/robot_localization/odometry/filtered` topic to fill nodes.

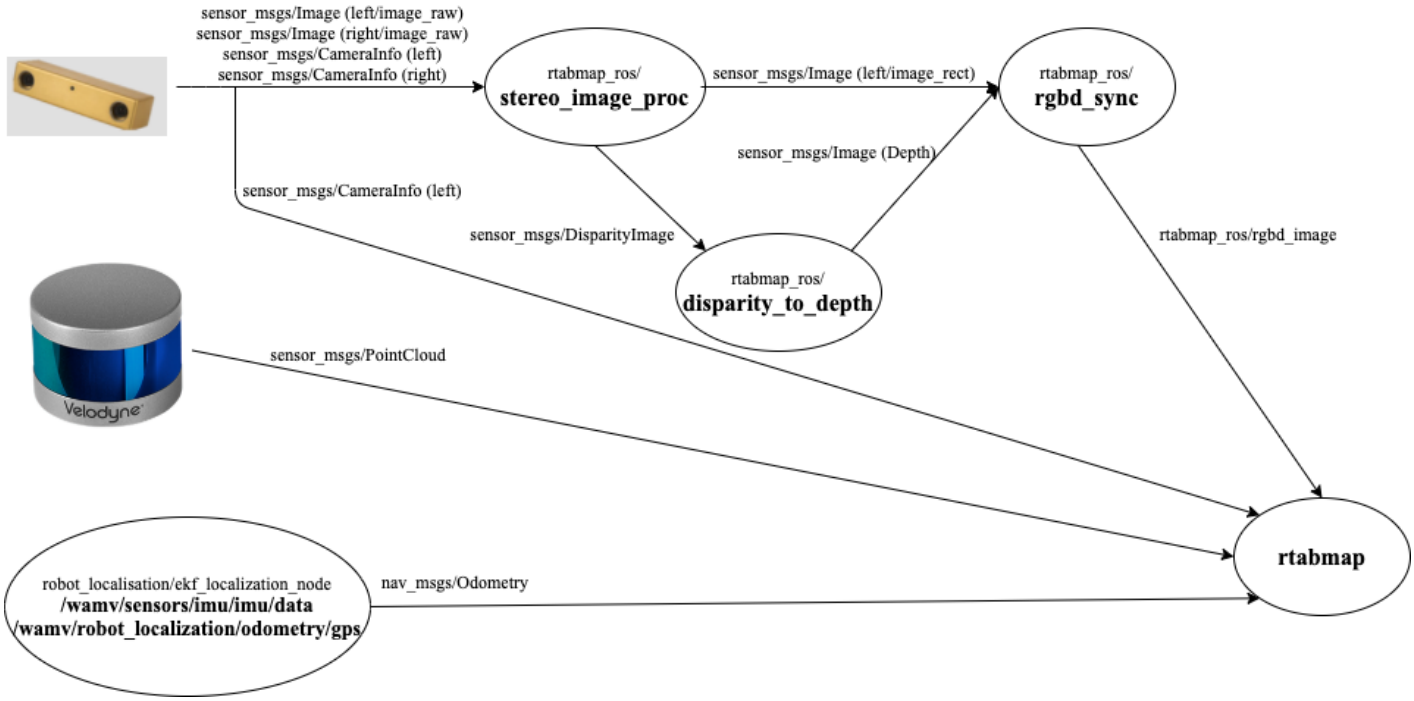


Figure 3.7: RGB-D mapping with lidar

Stereo

Alternatively the stereo images can be published to RTABMap with each respected monocular image. The raw stereo images has to be preprocessed by the `stereo_image_proc` node from `image_pipeline` package to rectify the images. RTABMap will use the rectified stereo images together with odometry information from either the `/robot_localization/odometry/filtered` topic or from a visual odometry estimation node `rtabmap_ros/stereo_odometry`.

3.8.1 Localization

The `move_base` planner use a global costmap generated by `rtabmap_ros/map` topic and a local costmap created by the `/wamv/sensors/lidar/pointcloud` topic. The `move_base` takes in a

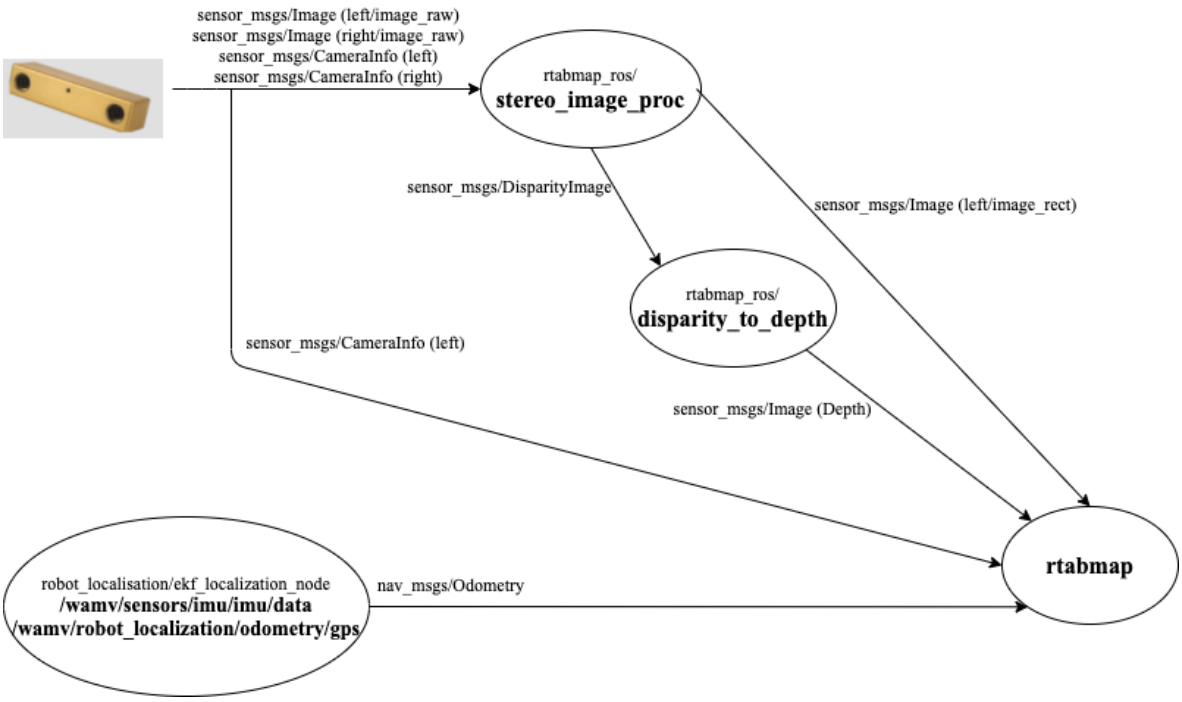


Figure 3.8: Stereo mapping

action goal massage and sends a `cmd_vel` twist message containing linear and angular velocity commands for each respected x, y and z coordinate, with the `wamv/base_link` body frame as reference. The vessel can be controlled with the twist messages using the `Twist2Thrust` node.

3.9 Transformation to a common reference frame

The sensor data is required to be transformed to a common reference frame. Each sensor has its own defined base link, which has to be transformed relative each other and the Body-fixed frame which moves with the vessel 2.3.2. In order to obtain the transformation between each sensors frame and the vessel CO, the lidar and camera need to be calibrated with a known reference. This can be done using ArUco markers with known dimensions. The ROS package `lidar_camera_calibration` will then use the known reference point to crate a rigid-body transformation matrix between the sensors. Which then has to be transformed to the body frame.

Because this project is based in a simulation it will not be necessary to manually calibrate the

sensors, as the base of each sensor is already known. Thus, the origin and distance between each sensor and the vessel base link is known and stored in a tree structure using the `/tf` package. The method described in 2.3.5 are then used to get lidar- and camera data to the common North-East-Down (NED) frame with the z-axis pointing toward the earths center, x-axis toward north and y-axis toward east.

3.10 Sensor Data Preprocessing

The RTABMap ros wrapper package contains implementation of several filtering algorithms and other preprocessing methods. The loop-closure detection require all the input data to be synchronized with the same timestamp. A queue with costomizable size of data can be stored and compared in order to approximate synchronization.

The `robot_localization` package can be used to fuse an arbitrary number of pose related sensors and estimate a continuous non-linear state. The Extended Kalman Filter `ekf_localization_node` will track the state of the vessel in 15 dimensions $\boldsymbol{X, Y, Z, roll, pitch, yaw, \dot{X}, \dot{Y}, \dot{Z}, \dot{roll}, \dot{pitch}, \dot{yaw}, \ddot{X}, \ddot{Y}, \ddot{Z}}$, then make a estimation, correct the estimation based on sensor data and use the correction to make a new estimated state. The mapping process can either be used in 3- or 6- Degrees Of Freedom (DOF), when using 3DOF, the registration will be forced in 2D using 3DOF x, y and yaw with z, roll and pitch set to zero. Both the vessel and obstacles will be affected by waves, causing them to move in three dimensions. Thus, the registration will be using 6DOF.

Images received from the stereo camera contains areas with little value for mapping, in fact, a lot of the information in a image frame can be received as noise and affect the end result. The vessels beams is in the lower part of the image frame and the skies on the top, all relevant information is concentrated at a small part in the middle of a frame. To lower the number of required computations and remove noise, a Region Of Interest (ROI) will be defined such that noise and irrelevant information is ignored. To remove the vessel beams from depth image a minimum registered depth can be set. Likewise, a maximum registered depth can be defined in such a way that background noise and objects far away will be ignored.

Chapter 4

Result

This chapter contain a objective presentation of results obtained in the duration of this project.

4.1 Simulation

4.1.1 Testsim in ROS Development Studio

Mapping, localization and navigation was successfully implemented in RDS. The turtlebot can manoeuvre to all navigation goals by using the navigation stacks move base package. The robot can map the environment and localize itself in it by using the rtab_map package. In addition, the turtlebot can autonomously navigate itself through the list of navigation goals given in the patrolling script that the group made in python.

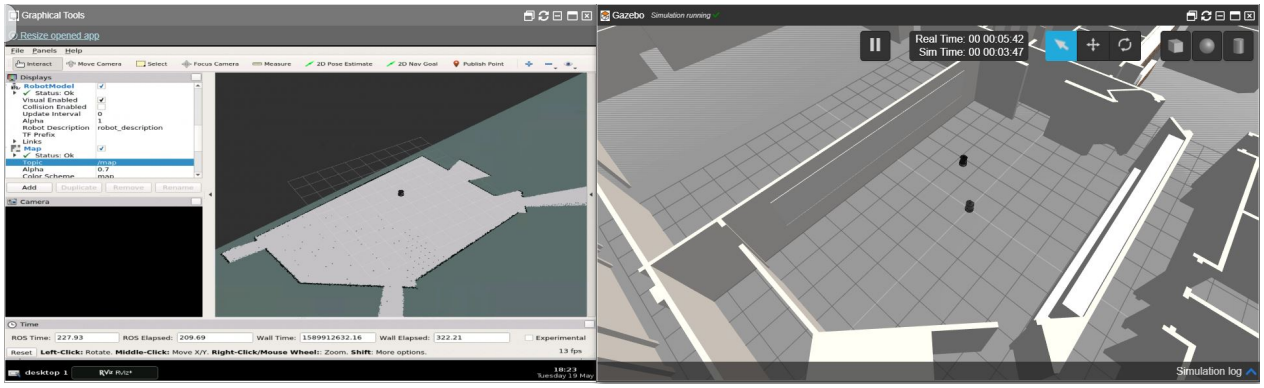


Figure 4.1: Room mapping using Turtlebot with RTAB script

4.1.2 WAM-V USV

A forked version of the simulation model WAM-V 14 USV was created and customised for this project. The model have integrated sensors as: Lidar, Stereo Camera, GPS and IMU.

Figure 4.2: Simulation model of the WAM-V 14 USV

4.2 Sensor fusion

Sensor data from IMU and GNSS was fused using `robot_localization/ekf_localization_node` implementing a Extended Kalman Filter and published to RTABMap with a `nav_msgs/Odometry` message, in addition, lidar point cloud was used with RTABMap to make corrections to odometry. Camera and lidar was fused together and stored in RTABMap-nodes for loop closure detection.

4.3 Sensor Data Preprocessing

Before filtering, the odometry experienced a lot of spikes when the vessel was affected by waves and during strafing.

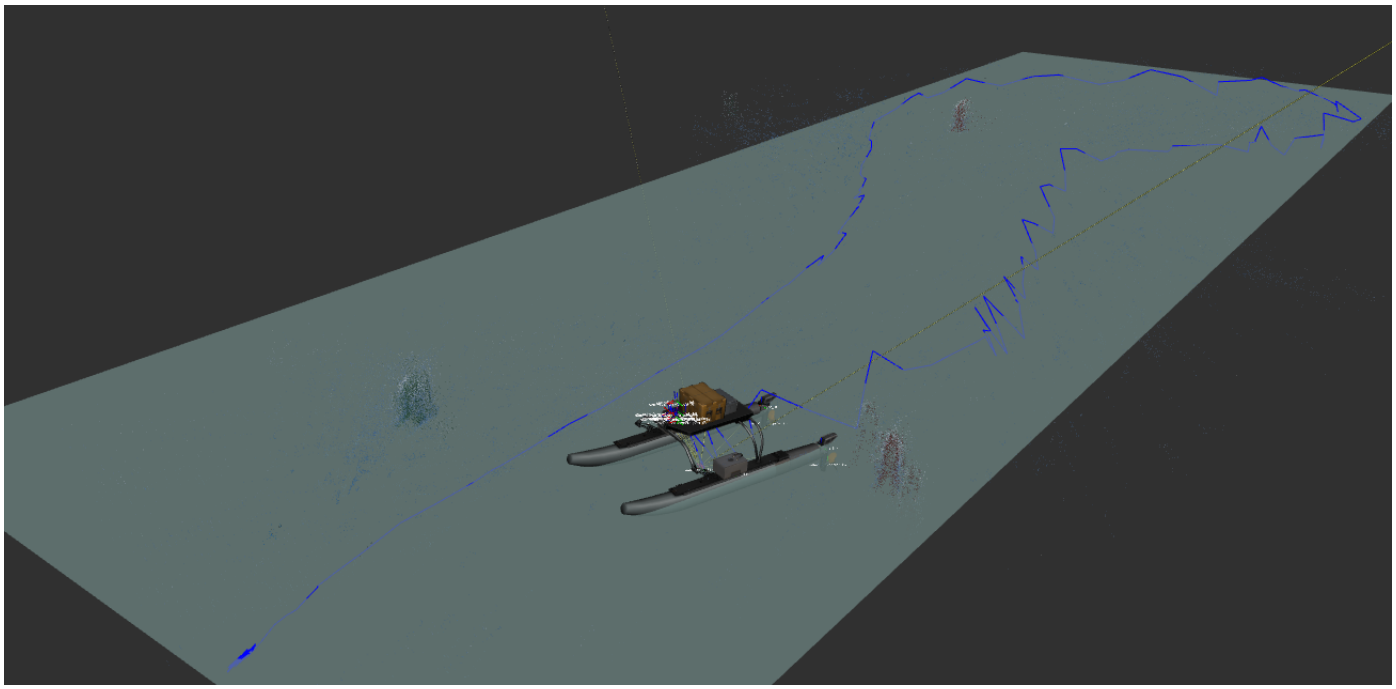


Figure 4.3: Unfiltered odometry

With filtering and using complimentary lidar scans for odometry corrections, the odometry graph was observed to be significantly smoother, as can be observed in figure 4.4. Note that the noise seen in the figure is registered points from the front beams of the vessel, as well as some from the water surface.

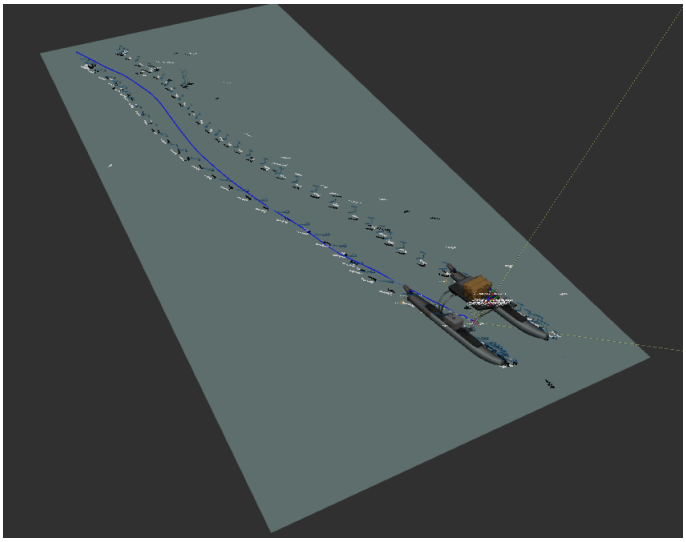


Figure 4.4: Filtered odometry, unfiltered point detection

A lot of noise from the water affected RTABMaps point detection, causing troubles with detecting loop closures. Also, the front end of the vessel beams is in the image and is getting detected as a feature.

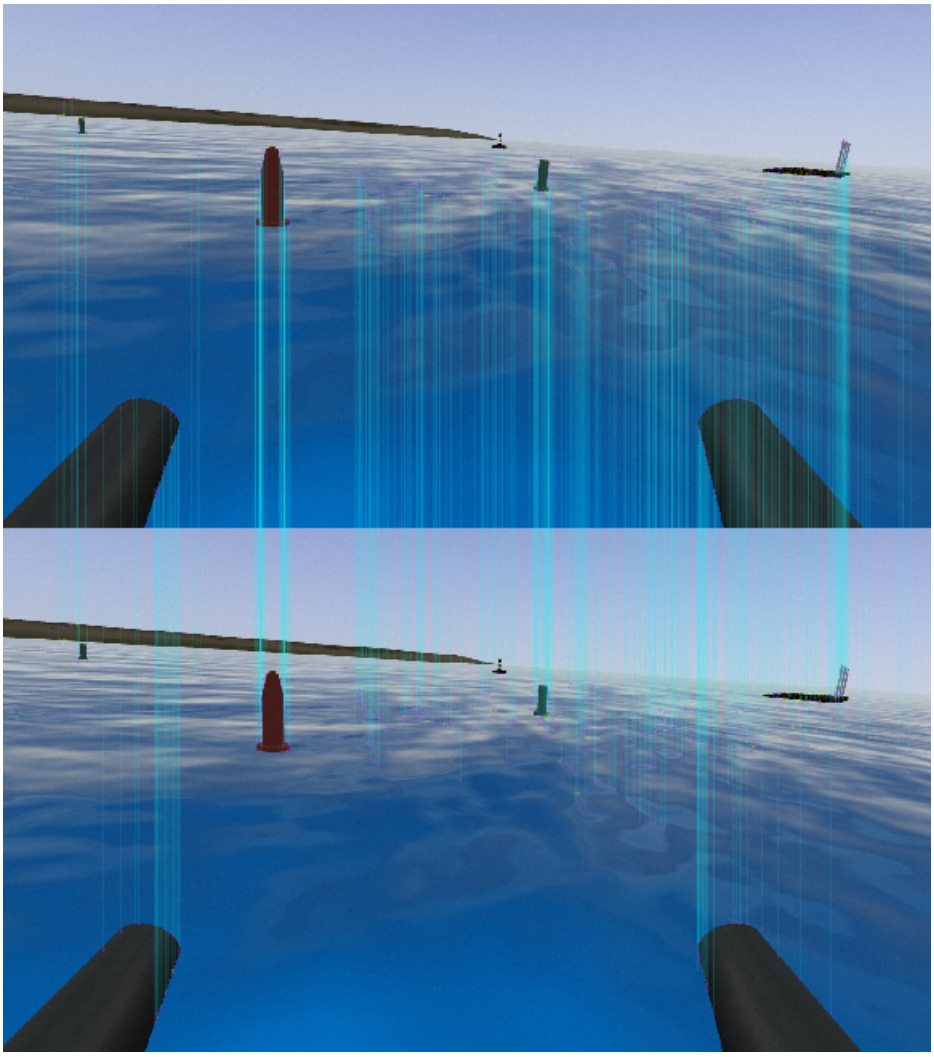


Figure 4.5: Unfiltered point detection

Then after adding parameters for ignoring points in the vessel footprint. Added a Region Of Interest within a area of 30% from the top and 40% from the bottom, this accelerated the calculations and got rid of unwanted points from the skies and vessels beams. Then a requirement of a 100 points cluster within an area of 1 meter radius was set. This led to filtering out noise from the water surface and only objects of interest was captured. Result after filtering noise, skies and beams can be seen in Figure 4.6.

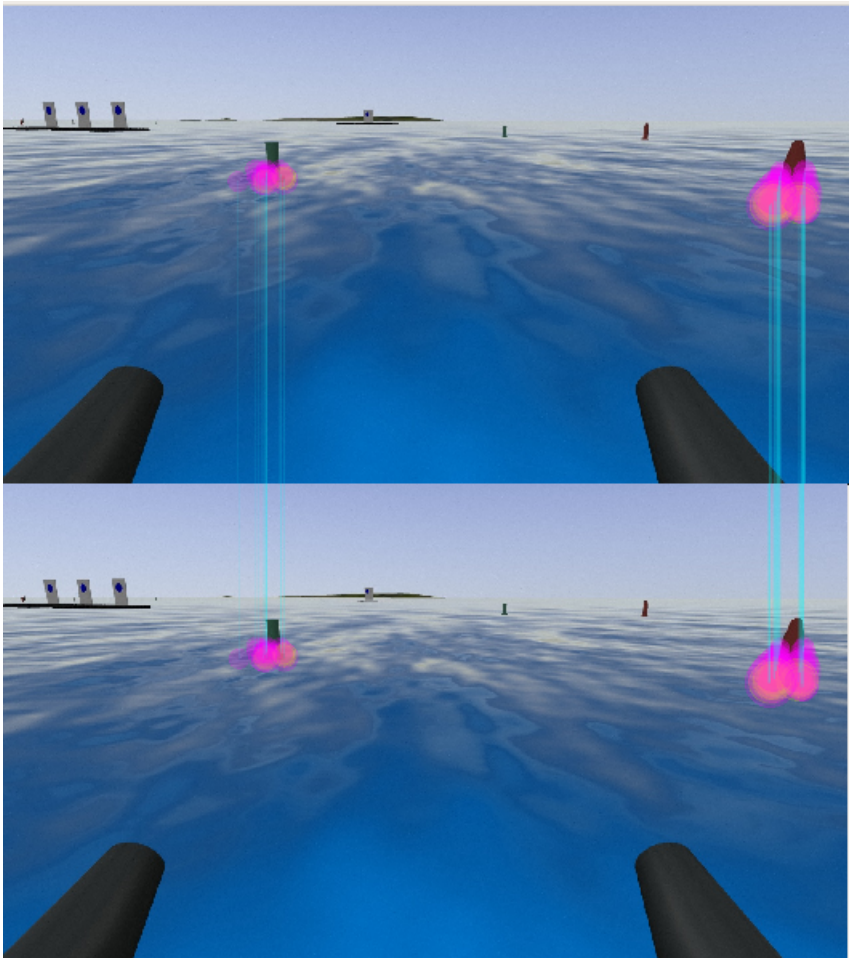


Figure 4.6: Filtered point detection

4.4 Mapping

Stereo

Mapping with stereo camera and odometry from the `robot_localization` was implemented using RTABMap ros wrapper. Raw stereo images was rectified using the `stereo_image_proc` node. The rectified stereo images was then used to create a depth image used by RTABMap to detect inliers, or matches in a image. However, the stereo mapping detected a lot of unwanted inliers in the water (see Figure 4.7). Visual odometry could not be successfully implemented, RTABMap will not launch without odometry information.

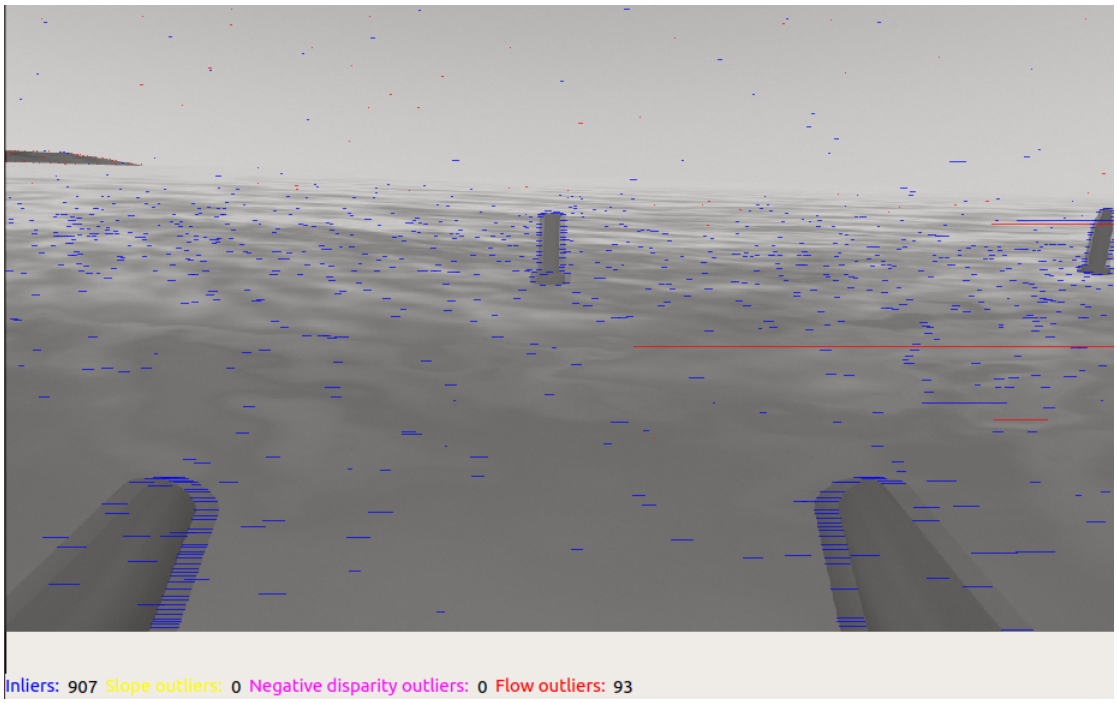


Figure 4.7: Inliers detected using stereo images

RGB-D

Stereo images was rectified using the `stereo_image_proc` node, then a depth image was created using `rtabmap_ros/disparity_to_depth` node (see Figure 4.8).

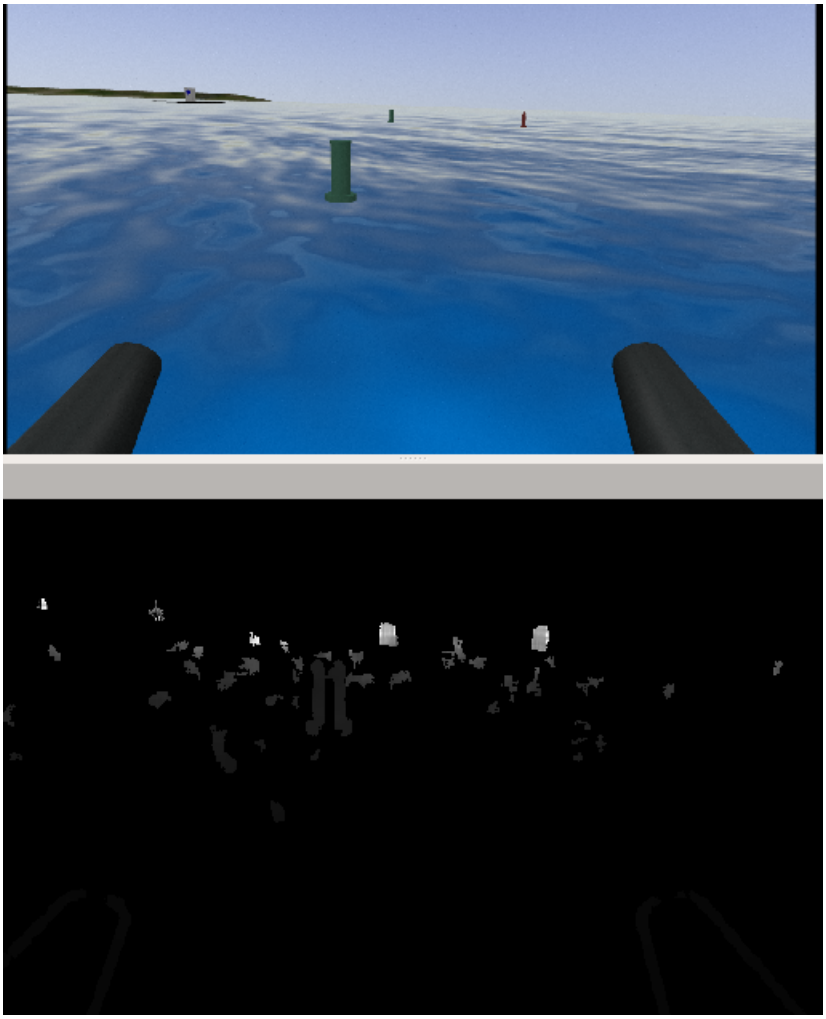


Figure 4.8: Depth image calculated from rectified stereo images

RGB and Depth images was fused together using `rgbd_sync` which publish a `rgbd_image` message to RTABMap. This is fused together and synchronized with lidar pointcloud and odometry messages from the `/ekf_localization_node`. This was the most successful method, the lidar made it possible to detect objects further away, giving RTABMap more nodes to compare and causing more loop closures detected.

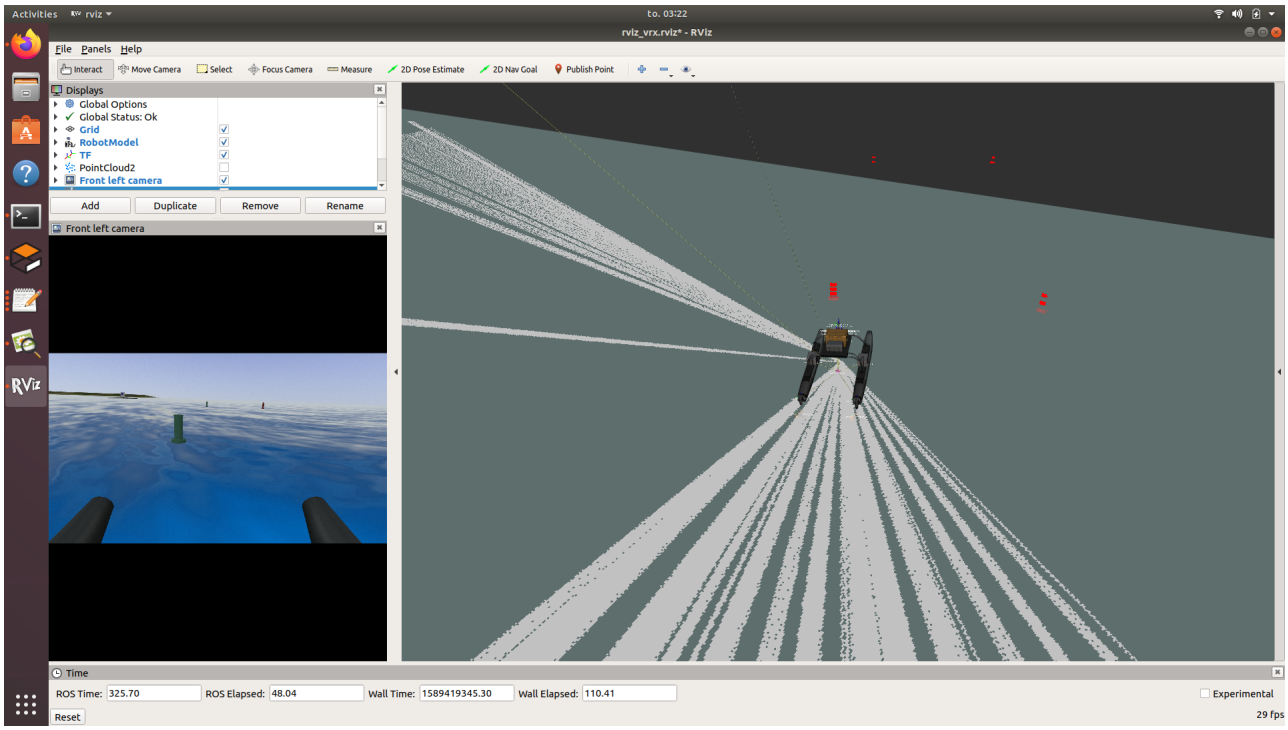


Figure 4.9: Successfully mapping with RGB-D and 3D lidar

4.5 Materials

4.5.1 Jetson TX2

The jetson TX2 were only used in the beginning, but barely used after the transition to the simulation. ROS were successfully implemented to the Jetson TX2 and prepared for further tasks with real-time object detection and RTAB mapping. The Pixhawk were also successfully implemented to the Jetson TX2. The VRX simulation was not successfully implemented on the TX2 due to issues with interfacing the GPU drivers.

4.5.2 Pixhawk MRo with GNSS reciever

The Pixhawk MRo were successfully implemented. The piwhawk was able to send IMU and GPU data to the ROS subscriber with UART. Location, orientation, acceleration was could be measured and located with the Pixhawk MRo. Could control the Pixhawk manually with a Playstation 3 controller.

4.5.3 Zed 2 Stereo Camera

The Zed 2 camera were not used to it's full potential in this project. The work that were concluded using the Zed 2 were data-collection of buoys. This can be read more about in chapter 4.6

4.5.4 Hull and Thruster

As mentioned in chapter 3.4.1 and 3.4.2 the chosen hull and thruster configuration ended up being an catamaran hull design using "X" Holonomic thruster calibration. Thus, this is what the characteristic for the prototype of the vessel will be. The material of the prototype is currently XPS Foam (Extended Polystyrene Foam). This material has been chosen because this was available in the lab and have been in use for making Ship Models. This prototype can be seen in figure 4.10

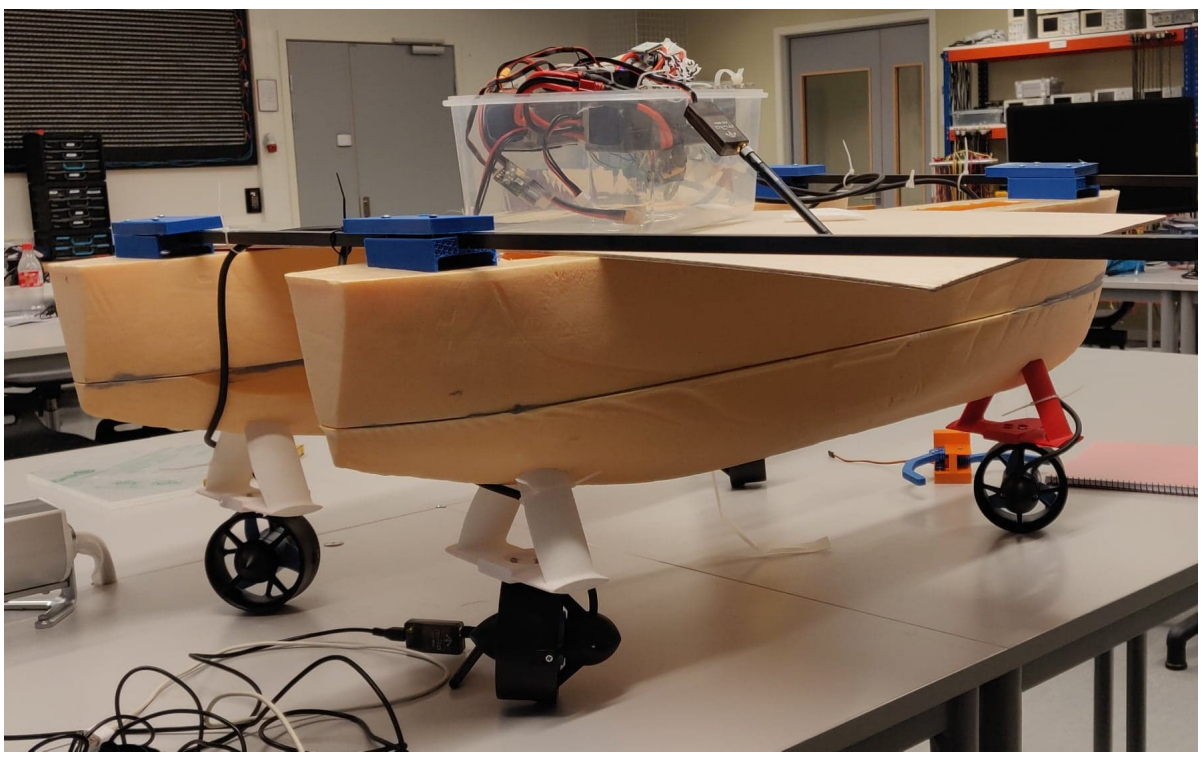


Figure 4.10: Prototype of the vessel

In the future, the boat will possibly be made out of fibre-reinforced plastic. This process is time consuming and this will only be done once the vessel satisfies all of the basic requirements like sufficient buoyancy, stability and manoeuvrability. Thus, because of the Covid-19 outbreak, this process got postponed.

As see in figure 4.10 the T200 thrusters were mounted using 3D printed mounts in the x-holnomic thruster configuration. These mounts were modelled and printed by the students from Naval Architecture. A test-run of the prototype were done in the ship model basin on campus. The basin allowed the prototype to be tested in a controlled environment including waves. This test-run were conducted by the Naval Architecture group.

4.6 Data collection and annotation

As mentioned in chapter 3.5.4 an python script was used to collect the images. This script had an simple and yet important task of both collecting and categorizing the different pictures involving the different object. The final version of the script can be found in the appendices as appendix G. This final script include functions for opening the video stream, saving an frame of the video stream and both update and retrieve an counter for each image which would be used to store them in a given order.

Thus, by running the script, with the stereo camera connected, and for instance pressing the "3" key on the keyboard will result in an image of the yellow buoy to be saved. The collected images ended up being blurred, the reason behind this is unknown, but will be discussed more in chapter 5.4. The images relentless ended up working out just fine, an sample of the collected data of the yellow buoy can be seen in figure 4.11.
Image labeling were done using the bounding box application labelIMG, the result of this can be seen in figure 4.12. This would result in an XML file containing the coordinate of the box, which would be used in the AI model.

Figure 4.11: Collected image of an yellow buoy

Figure 4.12: Bounding box of the yellow buoy

4.7 Buoy detection

The mAP score tells how likely the buoy predicts the correct colors of the buoys and is comparing the ground-truth with the prediction made by the AI.

4.7.1 Training result 1: 4000 iterations

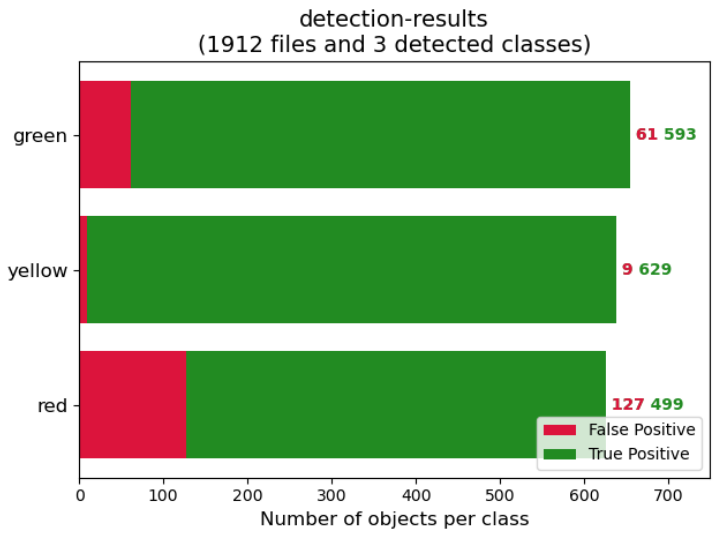


Figure 4.13: Detection result 1

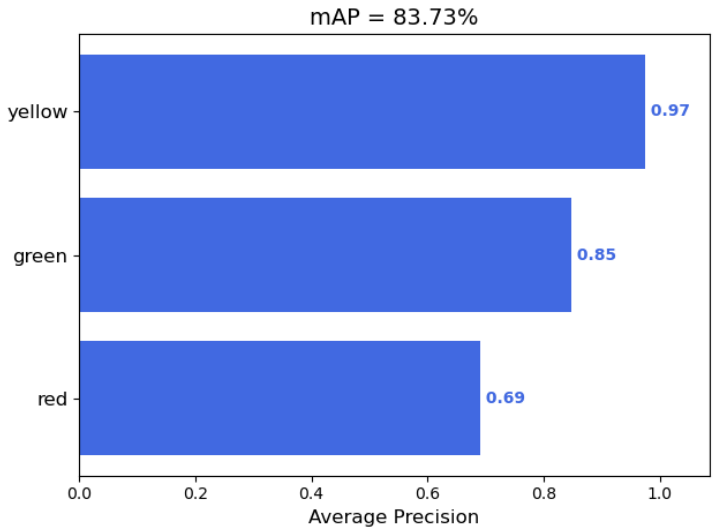


Figure 4.14: mAP 1

At 4000 iterations the mAP had a overall score of 83.73%. So far its noticeable that the yellow buoy is the easiest buoy to detect and the red one is the hardest.

4.7.2 Training result 2: 6000 iterations

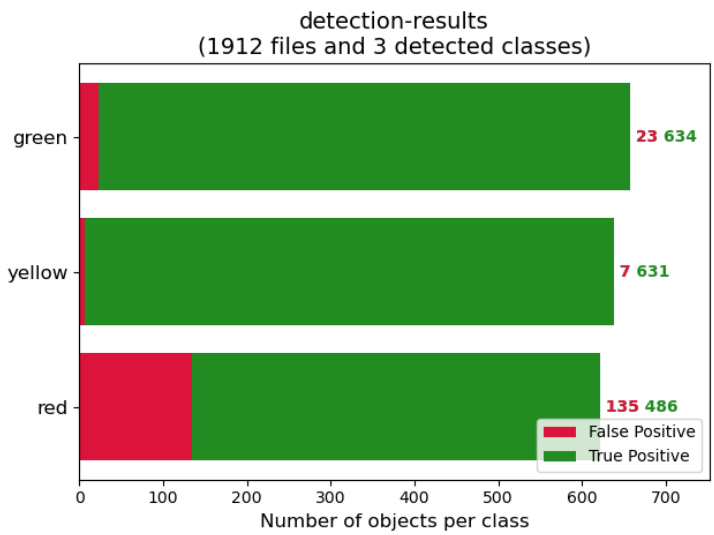


Figure 4.15: Detection result 2

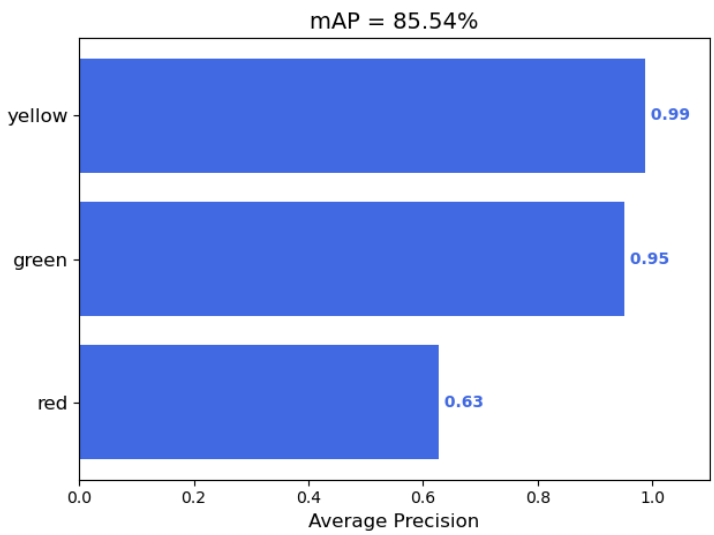


Figure 4.16: mAP 2

At 6000 iterations the green and yellow buoy detection is 95% and 99%, but still having problems detecting the red buoy, which pulls down the score to 85.54%

4.7.3 Training result 3: 7000 iterations

Figure 4.17: Detection result 3

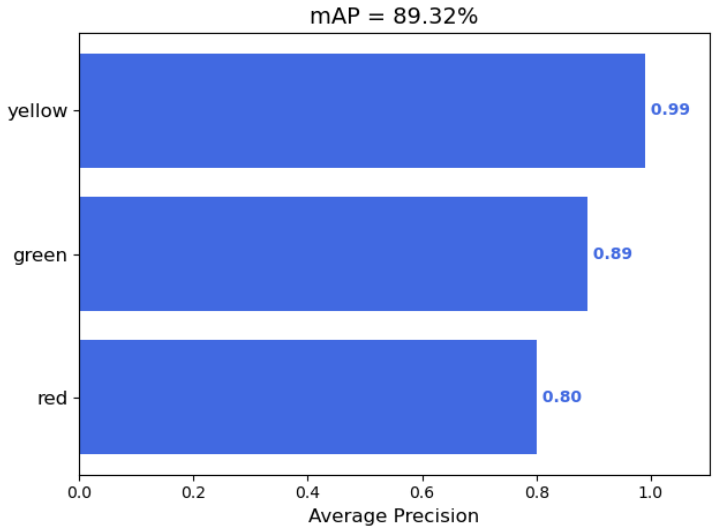


Figure 4.18: mAP 3

At 7000 iterations the overall mAP score is 89.32%. The red buoy achieves the highest score for red buoy detection so far in the training with 80%. Green buoy detection fell from 95% (fig: 4.16) to 89%. This mAP provides the highest score overall in the training process.

4.7.4 Training result 4: 8000 iterations

Figure 4.19: Detection result 4

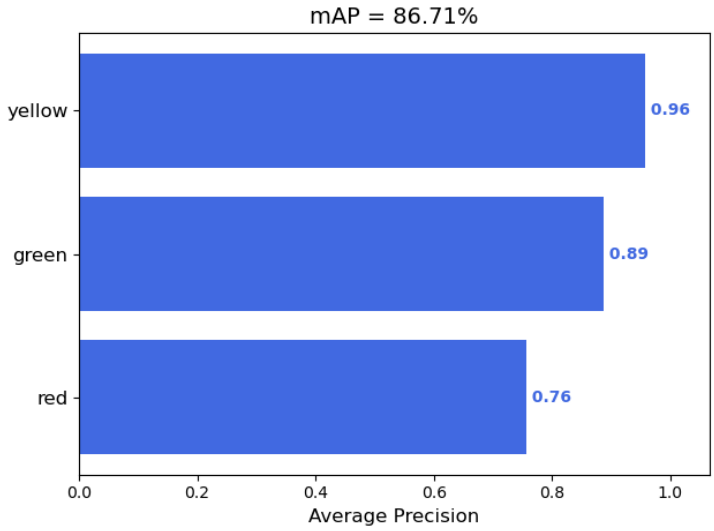


Figure 4.20: mAP 4

At 8000 iterations the overall mAP score is 86.71%. The score is lowered from the last checkpoint at 7000 iterations (4.16).

4.7.5 Object Detection

The following results are taken with the checkpoint at 7000 iterations.

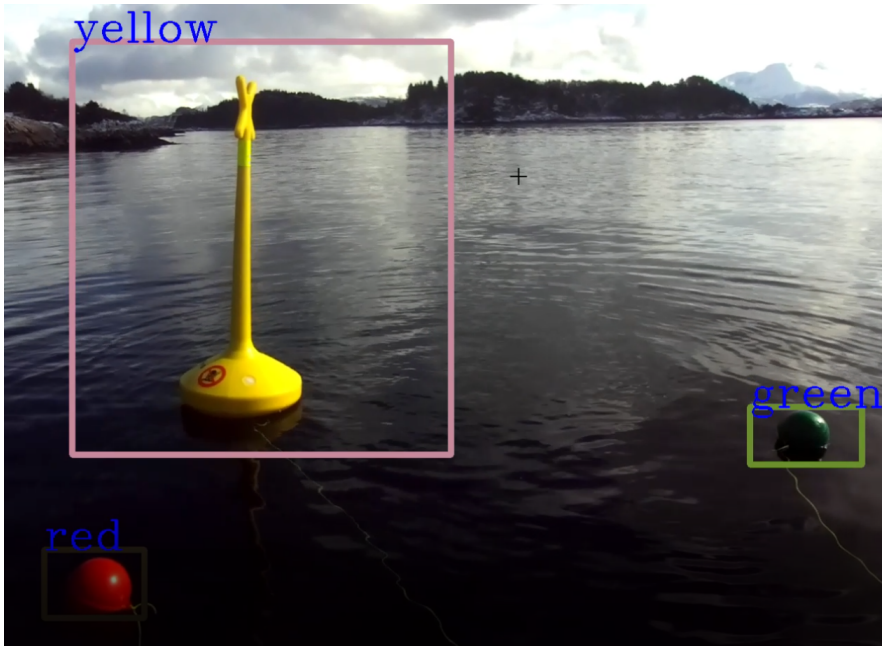


Figure 4.21: Buoy detection of all three buoys

Figure 4.22: Green buoy detection

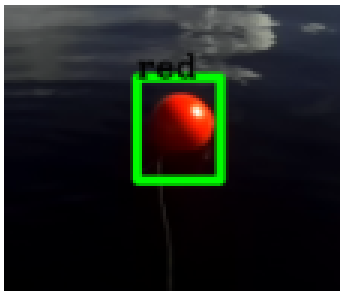


Figure 4.23: Red buoy detection

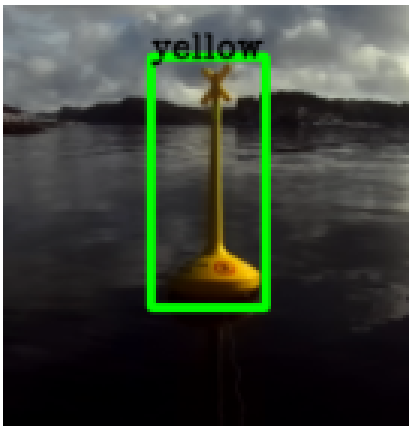


Figure 4.24: Yellow buoy detection

Chapter 5

Discussion

The bachelor thesis has changed because of the global pandemic, from creating an physical ASV to an simulated ASV. During the change new knowledge had to be gather to solve the new issues. In this chapter the reliability of the obtained results presented in the previous chapter will be discussed. The discussion will include both reflections about results before and after the transition.

5.1 Physical Vessel

Since the development of a physical vessel were halted by COVID-19, as discussed in chapter 3.2, the vessel didn't go past the prototype stage. However, the planning stage including blueprint sketching, material choosing and hardware selection have all been prepared for further development.

5.2 Mapping

The solution fusing RGB-D and 3D lidar point cloud provided the best result. At first, noise from water surface and the front end of the vessel beams created a lot of disturbances, limiting the amount of loop closures detected. After implementing filtering of points and a Region Of Interest a satisfactory result could be observed (see Figure 4.6). Although this implementation

is based in a simulated world, it provides reliability for a successful implementation of RTABMAp in a real solution. The solution should be able to be implemented with few adjustments.

5.3 Navigation

Autonomous navigation was successfully implemented in the test simulation on the Turtlebot, using the navigation stacks move base package. However, the navigation was not completely implemented on the WAM-V USV model. Global and local costmaps was set up properly, but RTABMap could not detect any known frames. This issue was not properly investigated, as a lot of time was spent on optimizing sensors and implementing mapping.

5.4 Data-collection and annotation

As mentioned in chapter 4.6 the collected images ended up being blurred by an unknown reason. The quality of the image were much better when the script were tested while being inside. One could say that the strong sun of the collection day did so that the camera could not tune completely, and thus result in blurred images. The bright sun was also a hindrance when it came to seeing what was on the screen of the laptop running the script. It is recommended to collect data on a day it is mildly cloudy so that one can more easily see how the resolution of the images is before taking the pictures.

The use of bounding boxes ended up being an superior choice of annotation for this project because it made the annotation progress less tedious, contrary to the use of polygons or semantic segmentation, and yet served good results as seen in chapter 4.7. The only flaw that this produced, were that there only were collected images of an full buoy, meaning that having a part of the buoy out of the frame would cause issues for the AI model. This flaw could also be looked on as an insurance policy for not recognize objects similar to the red buoy as an red buoy, amongst others.

5.5 Object detection

The result of the object detection algorithm is shown in section 4.7. The overall mAP score increases for every checkpoint that has been saved, except for the last checkpoint at 8000 iterations. Here the score decreased from **89.32%** (4.7.3) to **86.71%** (4.7.4).

The buoy detection was trained for 8000 iterations, every iteration above 8500 would result in NaN values which made the detection predict nothing and the mAP would be **0%**. To help prevent this the batch size and learning rate needs to be adjusted. The learning rate tuning is a trial and error process since it's not possible to analytically calculate the best possible learning rate for a given model. the mAP was based on training with batch size of four and a learning rate of 0.000005.

For the training YOLO had approximated 600 images and with similar amount of annotation files for each class for training. Since deep learning algorithms as YOLO scale with data (2.10) the YOLO algorithm could be trained on a lot more images. Training with a lower amount of data will cause the algorithm to converge slowly and the accuracy will be low. By increasing the amount of data the model will converge faster and the accuracy will increase. The training data should be images of the buoy in different lightning and weather scenarios. All images that has been used for the training was taken at the same day in a period of a few hours. Images with different lightning would improve the buoy detection, when trained with different light, it would be prepare the buoy detection in different settings and the overall score would improve.

First the R-CNN network was implemented, but because of errors and problems with installing R-CNN, YOLO was implemented instead. But in generally YOLO network is a lot faster than the R-CNN because YOLO have a simpler architecture. YOLO also works better and provides better FPS in real time object detection.

The implementation of darkflow (Translate Darknet to tensorflow for Windows) took some time to install. NVIDIA CUDA-and cuDNN had be set up correctly on the system to be able to train with the GPU. It's possible to train with a CPU, but that would increase training time a lot.

The object detection was trained on a NVIDIA GeForce 1660ti (6Gb) which trained the model in a few hours. The model could only use a batch size of two with this GPU. The training process

received an error when using a larger batch size. Training on better could make it possible to increase batch size. The object detection algorithm was trained with a learning rate of 0.000005. To reduce the training time and increase the performance of the object detection a better GPU could be implemented. It's also possible to rent servers online to train the AI instead of using a personal computer.

5.6 ROS Development Studio

ROS Development Studio is a good tool for learning ROS. The application provides a lot of guides and tutorials which were useful while working on the thesis. The robot received good results(4.1.1) in the simulator. The robot could navigate autonomously in the the mapped area. It's important for the localisation that the map is mapped thoroughly, because that will provide more data for the RTAB map database and will help with recognising patterns in the map. ROSDS is good tool for testing different type of packages. All typical packages are already preconfigured in ROSDS, which simplifies installation of packages.

5.7 Scrum to Gantt transition

In the planning phase the group though that an adjusted version of scrum would be the best fit for the work methodology. The group made a comprehensive backlog and adjustments to the standard scrum method in order to fit our cross disciplined project. Usually the MVP description is what supports the scrum backlog but the group had a lot of product requirements regarding hardware and research, the group would not include these I the scrum backlog and rather assume that each sprint contained about 30% of non-backlog work. Therefor, when deciding on sprint backlogs the group would try to estimate 70% scrum-work and document the 30% non-scrum-work in the work-hour list "attachment name of work-hour list" attachment.

Starting development, the group found out that there was a lot less coding work the the group originally though out. It was very often that once the group started working on an issue, after a bit of research, the group found that the way we planned to do it was not optimal and the group would have to change the issue completely. Therefore, the scrum backlog was very hard

to use, and the group had to make many adjustments to issues during the sprints. After experimenting with continuously changing the scrum and sprint backlog, the group decided on transferring to gantt-diagram in hopes of spending less time on project organisation and more time on development. The gantt-diagram was easier to follow, especially when it comes to tasks where there was minimal coding and more familiarizing with and tweaking an already existing solution. In addition, it became easier for supervisors to get an overview of our current state of progress.

Looking back, the group should have used gantt-diagram for the progress tracking from the beginning. It offers a greater overview of the project development phase as a whole and gives a better presentation of the project's time frame. In addition, any third-party contributors to the projects can quickly get an overview of the progression state.

5.8 Phase improvement potential

In this section, the groups most obvious improvement potential is listed seen from the groups perspective. For the phases that are not mentioned here such as testing, hardware or bachelors report, the improvement potential was minor and not worth mentioning.

5.8.1 Pre-project

The task given by NTNU was open ended and not defined, the group therefore had to create their own task and parameters for success. However, after the COVID-19 crisis escalated, the project problem description had to be re-done as mentioned in 3.2 Deviation management and current work had to be put on hold indefinitely. The group could have made a better deviation management plan in the pre-project phase which would have helped to spare time, hassle and other problems like motivation and mental state. The deviation management plan should have been more detailed and stricter with parameters for transition states and a plan for each of these transitions. An example would be like the situation where the group could not do development using hardware at the school's premises. Here the deviation management plan should have described what should have been done if this were to happen. Optimally, when the group would

find themselves in the position of not being able to develop using hardware, the plan should then have had a set transition phase with already set tasks for them to work with in order to use time more effectively.

5.8.2 Research

The group made a decisive distinction between research and development to create more solid plans and designs. This resulted in not having to derive plans as often as other groups because of the research foundation. Simulation testing could have been tested more thoroughly in the beginning of the research phase in order to test ROS solutions like the Navigation stack early and more often. This would have helped to get a greater understanding of how to tweak the parameters and implementation.

5.8.3 Planning Design

A detailed MVP-description was made, however, some of the user stories was too large in a sense where it would initially describe two user stories in one. The group were made aware of this by supervisors and improved the MVP-description after getting the feedback. The group spent time creating a finite state machine diagram for the ASV, it turned out that this was not necessary for the development due to the fact that the Navigation stack handles the states described in the diagram by itself. If greater research on the Navigation stack were done prior to this the group could have allocated time better. However, it is not a large mistake because it granted the group a greater understanding of what decision bias an ASV should have. Unfortunately, this diagram disappeared in the software Confluence and supervisors of the software was not able to recover it, therefore, it is not attached to this report. The group did not find it necessary to re-create it after the COVID-19 outbreak because of it lack of relevance to the new problem description.

5.8.4 Software

The group started working on navigation and simulation simultaneously but on different platforms. The navigation work was done in ROS Development Studio whilst several simulation

options Were researched and tested. The group ended up using the VRX simulation mentioned in chapter 4.1. It were considered to create a simulation from scratch, but it would be too challenging and time consuming with the time lift in the development phase. Extending this phase was not an option. At the time when these tasks were at the centre of attention it was still unclear whether the group should develop towards a physical or simulation prototype. Different feedback were given from supervisors saying that the group should aim towards the contest and others saying that a simulation should be prioritized and so it were more difficult to decide on what direction the project should go towards. Ultimately, it was decided on trying to complete as much of the original idea as possible and set up a simulation for later developers/students to use as their testing platform and give an extensive guide to how the next generation of developers could use it for prototyping. Considering the situation, the group managed well how to continuously find new ways of replacing our first thought out plans. However, progress staggered a bit during the implementation of the navigation stack in the VRX simulation. Creating our own WAMV boat autonomous failed due to time constraint. However, the test robot Turtlebot2 in RDS turned our completely autonomous.

Chapter 6

Conclusions

During this project, the group has studied the field of autonomous- and unmanned surface vehicles and built components for an ASV. In addition, the group have set up and configured a simulation and tested different navigation solutions for ASVs. The group intended to develop a fully functional ASV prototype.

The requirements for the ASV prototype was to be able to manoeuvre through each of the courses in the national contest AutoDrone. The prototype should avoid any obstacles in it's way, including dynamic obstacles. In addition, the ASV should have the feature of autonomous docking.

The group was not able to create a physical prototype becasue of COVID-19. Instead, the project pivoted into a direction of simulation. The requirements for the new direction was to complete as much of the started work as possible. In addition, the group should set up and configure a simulation for next years students to test out solutions. That way, this years work would help later developers from NTNU to compete in the national contest AutoDrone.

The results shows that it is possible to create all software needed for a WAM-V model to be autonomous in a simulated environment and test in a course similar to the contest.

The group was able to implement YOLO for object detection for three different essential buoys. Two of the buoys had the same shape, but different colors. The algorithm was able to track

the buoys in a live video with decent accuracy. The accuracy of the buoy detection could be improved by increasing the dataset and more training.

The group considers this project a success, and to be challenging due to it's pivot and extensive scope. The group learned to deviate in a way none of the team members has done in the past. The learning outcome is wider considering the work form and the projects cross disciplinary nature. In addition, managing a project of this size and constantly transforming knowledge to competence has been a rewarding experience. The group concludes that it is possible to develop a contest ready ASV intended for the AutoDrone competition and that the VRX simulation will be helpful for next years students considering hardware tests, software- development and testing.

Bibliography

[1] Unmanned Systems Technology. Unmanned surface vehicles (usv). https://www.unmannedsystemstechnology.com/category/supplier-directory/platforms/usv-manufacture/, 2020. [Online; accessed 15-May-2020].

[2] Thor I. Fossen. Handbook of marine craft hydrodynamics and motion control, 2011.

[3] MathWorks. Extended kalman filters. https://www.mathworks.com/help/fusion/ug/extendedkalmanloop.png. [Online; accessed 17-May-2020].

[4] Inovasjon Norge. Invest in norway - maritime. https://www.innovasjonnorge.no/en/start-page/invest-in-norway/industries/maritime1/, 2020. [Online; accessed 3-April-2020].

[5] International Maritime Organization. Autonomous shipping. http://www.imo.org/en/MediaCentre/HotTopics/Pages/Autonomous-shipping.aspx, 2020. [Online; accessed 3-April-2020].

[6] Mathworks. Convert from geodetic latitude, longitude and altitude to flat earth position. https://se.mathworks.com/help/aerotbx/ug/lla2flat.html?w.mathworks.com, 2020. [Online; accessed 10.04.2020].

[7] Royal Observatory of Belgium. How gnss works. http://gnss.be/how_tutorial.php, 2011. [Online; accessed 16-April-2020].

[8] Boat-Ed. Descriptions of hull shapes. https://www.boat-ed.com/indiana/studyGuide/Descriptions-of-Hull-Shapes/10101602_35113/, 2020. [Online; accessed 15-May-2020].

[9] Robot Platform. Robot locomotion. http://www.robotplatform.com/knowledge/Classification_of_Robots/Holonomic_and_Non-Holonomic_drive.html, 2020. [Online; accessed 27-April-2020].

[10] Intel Corporation. The basics of stereo depth vision. https://www.intelrealsense.com/stereo-depth-vision-basics/, 2020. [Online; accessed 16-April-2020].

[11] Robert Collins. Lecture 08: Introduction to stereo. http://www.cse.psu.edu/~rtc12/CSE486/lecture08.pdf, 2007. [Online; accessed 20-April-2020].

[12] limarc2000. What is image annotation? https://hackernoon.com/what-is-image-annotation-an-intro-to-5-image-annotation-services-yt6n3xfj, 2020. [Online; accessed 21-April-2020].

[13] Mathworks. What is deep learning? https://www.mathworks.com/discovery/deep-learning.html. [Online; accessed 22-April-2020].

[14] Ross Girshick Ali Farhadi Joseph Redmon, Santosh Divvala. You only look once; unified, real-time object detection. https://arxiv.org/pdf/1506.02640.pdf, 2016. [Online; accessed 15-April-2020].

[15] Jonathan Hui. Real-time object detection with yolo, yolov2 and now yolov3. https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088, 2018. [Online; accessed 15-April-2020].

[16] João Cartucho. mean average precision - this code evaluates the performance of your neural net for object recognition. https://github.com/Cartucho/mAP, 2020. [Online; accessed 29-April-2020].

[17] towardsdatascience Atul Singh. Extended kalman filter. https://towardsdatascience.com/extended-kalman-filter-ee9bd04ac5dc, 2018. [Online; accessed 17-May-2020].

[18] Gary Bishop Greg Welch. An introduction to the kalman filter. https://www.cs.unc.edu/~welch/media/pdf/kalman_intro.pdf, 2004. [Online; accessed 17-May-2020].

[19] Matworks. Extended kalman filters. https://www.mathworks.com/help/fusion/ug/extended-kalman-filters.html#bvkyzrp. [Online; accessed 17-May-2020].

[20] Blue Robotics. T200 thruster. https://bluerobotics.com/store/thrusters/t100-t200-thrusters/t200-thruster/, 2020. [Online; accessed 28.02.2020].

[21] OpenCV Team. About opencv. https://opencv.org/about/, 2020. [Online; accessed 4-May-2020].

[22] tzutalin. labelimg. https://github.com/tzutalin/labelImg, 2020. [Online; accessed 21-April-2020].

[23] Jerry Tessendorf. Simulating ocean waves. http://www.bostater.info/research_docs/synthetic/coursenotes.pdf, 1999. [Online; accessed 13-May-2020].

[24] Dr. Brian Bingham. Gazebo usv plugins, theory of operation. https://github.com/bsb808/robotx_docs/blob/master/theoryofoperation/theory_of_operation.pdf, 2018. [Online; accessed 13-May-2020].

[25] Marine Advanced Robotics. Wam-v 14 usv. http://www.wam-v.com/wam-v-14-usv, 2020. [Online; accessed 13-May-2020].

[26] The Construct. Ros development studio. https://www.mathworks.com/help/fusion/ug/extended-kalman-filters.html#bvkyzrp. [Online; accessed 19-May-2020].

[27] ROS. Stereo image proc. http://wiki.ros.org/stereo_image_proc, 2020. [Online; accessed 13-May-2020].

[28] OpenCV. Stereo block matching. https://docs.opencv.org/2.4/modules/calib3d/doc/camera_calibration_and_3d_reconstruction.html#stereobm, 2020. [Online; accessed 14-May-2020].

Appendices

A Preproject report

B Gantt diagram

C Hour list

D MVP

E BOM

F AutoDrone 2020 Rules and Task Description

G Buoy data collection

H RGB-D Mapping launch file

I RGB-D Localization file

J YOLO Training file

K YOLO mAP validation file

L YOLO Processing Video file

M YOLO Processing Image file

N YOLO Labels file

O YOLO CFG file

P Patrol file

Appendix A

Preproject report

PRE-PROJECT REPORT

FOR BACHELOR THESIS

NTNU
Kunnskap for en bedre verden

TITLE:
Autonomous surface vehicle

CANDIDATE NAMES/NUMBERS:
Mikal Aanning / 488566
Sigurd Odden / 488577
Markus Holt / 481922
Yrian Hovde Øksne / 488582

DATE:	COURSE CODE:	SUBJECT:	DOCUMENT ACCESS:
30.01.2020	IE303612	Bachelor thesis	- Open

STUDY:	PAGES/APPENDIX:	LIB.NUMBER:
AUTOMATION ENGINEERING, DATA ENGINEERING	15/3	- NA -

SUPERVISORS:

- Ottar L. Osen - Associate Professor at NTNU
- Robin T. Bye – Associate Professor at NTNU
- Girts Strazdins– Associate Professor at NTNU
- Elias Hasle – PhD. Candidate and scientific assistant at NTNU
- Anete Vagale – PhD. Candidate at NTNU

TASK/SUMMARY:

The pre-project includes formalities of labour distribution and group rules throughout the bachelor thesis. The bachelor thesis is given by NTNU in Ålesund.

The goal of the bachelor thesis is to make an autonomous surface vehicle that can handle three different obstacle courses. This includes designing, programming and building the surface vehicle with usage of computer vision and sensors.

After the project, there will be national competition between universities in Norway. The competition is divided into three different categories: design, weight and thrust versus weight.

This assignment is an examination answer given by students at NTNU campus Aalesund.

CONTENTS

CONTENTS **2**
1 INTRODUCTION **3**
2 TERM **3**
3 PROJECT ORGANIZATION **3**
3.1 Project group 3
3.1.1 Tasks for the project group – organization *3*
3.1.2 Responsibilities and tasks for project manager *4*
3.1.3 Responsibilities and tasks for secretary *4*
3.1.4 Responsibilities and tasks for other members. *5*
3.2 Supervisors and contact persons. 5
4 APPOINTMENTS **6**
4.1 Workplace and resources 6
4.2 Group norms – cooperation rules – attitude 6
5 PROJECT DESCRIPTION **7**
5.1 Problem description 7
5.2 Requirements for solution or project result - specification 7
5.2.1 Contest Eligibility Requirements *7*
5.2.2 Solution Requirements *7*
5.3 Planned procedure for development – method 9
5.4 Gathering of information – gathered and planned 9
5.5 Assessment – Analysis of risk 10
5.6 Main activities in further work 11
5.7 Progress plan – project management 11
5.7.1 Master plan *11*
5.7.2 Management utilities *11*
5.7.3 Development utilities *11*
5.7.4 Internal control – evaluation *12*
5.8 Decisions – decision-making process 13
6 DOCUMENTATION **13**
6.1 Reports and technical documents 13
7 PLANNED MEETINGS AND REPORTS **13**
7.1 Meetings 13
7.1.1 Meetings with supervisors *13*
7.1.2 Project meetings *13*
7.2 Periodical reports 14
7.2.1 Progress reports (incl. milestone) *14*
8 PLANNED DEVIATION MANAGEMENT **14**
9 EQUIPMENT/REQUIREMENTS FOR IMPLEMENTATION **14**
9.1 Hardware 14
10 REFERENCES **15**

1 INTRODUCTION

The task at hand is to develop an autonomous surface vehicle (ASV) that will participate in a national competition between universities in Norway held in June of 2020. In this pre-project report the group will define what roles each group member has and what responsibilities they have. The premises of where the group will work has been defined alongside what resources they have been provided with in form of money, supervisors and access. In addition, the groups work norms, attitude and cooperation rules will be set.

The project will be described in conjunction with what problems the group face, how the group will plan to develop a solution, gather information, assess risk and what their desired progression will look like. Other declarations like technical documentation, meeting form, frequency of meetings and how progress will be tracked has been decided on. Lastly, the group has decided how project deviation will be managed and what equipment and requirements the implementation will need.

2 TERM

ASV - Autonomous surface vehicle

IDEA - Integrated Development Environment Application

3 PROJECT ORGANIZATION

3.1 Project group

Name	**Student numbers**
Mikal Aanning Sigurd Odden Markus Holt Yrian Hovde Øksne	488566 488577 481922 488582

3.1.1 Tasks for the project group – organization

Responsibility	**Group member responsible**
Project manager	Yrian Hovde Øksne
Secretary	Markus Holt

3.1.2 Responsibilities and tasks for project manager

Responsibilities:

- Assure continuous progression towards the desired solution
- Assure that every group member has an overview of the projects state
- Assure that every group member knows what the end goal is and how to achieve it

Tasks:

- Continuously following up on how each group member is progressing
- Delegating workloads in a democratic manner and making sure that most of the group is satisfied with their given tasks
- Coordinate stand-up meetings at the start and end of a workday
- Coordinate weekly progression meetings

In addition, the project manager also has the responsibilities listed in 3.1.4.

3.1.3 Responsibilities and tasks for secretary

Responsibilities:

- Arrange documentation
- Take notes during meetings
- Economy

Tasks:

- Create and maintain a budget
- Create and maintain a timesheet where all the members update their own work hours
- Have control of meeting schedules, send updates to the participants prior the meetings
- Assure that each member document their work

In addition, the secretary has the responsibilities and tasks listed in chapter 3.1.4.

3.1.4 Responsibilities and tasks for other members.

Responsibilities:

- The group members are responsible for the tasks which are assigned to them
- The group members are responsible for logging what they have done

Tasks:

- Making sure that the assigned tasks are in progress
- Writing project report through the entire duration of the project
- The group members are responsible for logging their own working hours

3.2 Supervisors and contact persons.

- Ottar L. Osen - Associate Professor at NTNU
- Robin T. Bye - Associate Professor at NTNU
- Girts Strazdins - Associate Professor at NTNU
- Elias Hasle - PhD. Candidate and scientific assistant at NTNU
- Anete Vagale - PhD. Candidate at NTNU

4 APPOINTMENTS

4.1 Workplace and resources

As all group members live and study in Aalesund, we agreed on working at the university's campus. The supervisors have assigned the computer lab L167 to our class and we agreed to use this premises during the project. In addition, the group has the possibility to work from home during certain days or periods when it is needed. At this premises we have supervisors and lecturers close by, which helps with easier access to guidance and help if needed. The project was given a budget of about 90 000 NOK by NTNU, this is our only financial resource.

4.2 Group norms – cooperation rules – attitude

- Core time of the workday will usually be Monday to Friday from 09.00 to 16.00
- The Automation engineering students have another course: *IP300416 - Industri 4.0*. The students will not participate for Mondays, Wednesdays and Thursday every other week of a period of 8 weeks. In addition, one group member is undertaking the course: *IF300114 – Ingeniørfaglig systemteknikk og system-utvikling*. This group member will be occupied Mondays 10:15 – 14:00 and Wednesdays 10:15 – 12:00 from week 2 – 11
- Daily stand-up meeting at 09.00 before working on assigned tasks
- Status meeting every Friday 09.00
- Meeting every other week with supervisors
- Absence or delays must be reported to the group leader or other group member
- Meet precisely at the planned time

5 PROJECT DESCRIPTION

5.1 Problem description

The group not having a fitted ASV to compete in the competition spoken about by the supervisors is a bachelors project failure. This will result in a lower grade than wished for by the group. Also, this will result in NTNU not having a contestant for the competition.

An ASV may be developed to manoeuvre through each of the three courses mentioned in 5.2.2 without problems. In addition, the ASV will satisfy the parameters in the Task Scoresheet from 5.2.2 in order to harvest the most points available.

5.2 Requirements for solution or project result - specification

To mention, these requirements has not yet been concluded on, but the ones written about in 5.2 will act as a point to what the complete requirements will be. All requirements in 5.2 is a product of the introductory meeting between the group and the supervisors in conjunction with rules from Attachment 1 – RoboBoat 2020 Rules and Task Description V1. The contest eligibility requirements are a mix of the team requirements from Attachment 1 – RoboBoat 2020 Rules and Task Description V1 and the Task Scoresheet from Attachment 2 – RoboBoat 2019 Rules and Task Description_v2.

5.2.1 Contest Eligibility Requirements

To participate in the competition, the team must be consistent of 75% full time students, which must include primary school, undergraduate, graduate, and post-doctoral students. However, 25% of the team may be from an industrial, academic or governmental workforce.

One vehicle per team may be entered in the competition. Each team designate a student member as their team leader. The team leader is the only one who can speak for the team, to request vehicle deployment, run start, run end, or vehicle retrieval. The team leader must be conversationally fluent in Norwegian.

The vehicle cannot weigh more than 64kg and dimensions cannot be greater than, 91.5cm in width, 91.5cm in height or 183cm in length.

5.2.2 Solution Requirements

In an introductory meeting with supervisors, it was mentioned that the solution requirements were tightly tied up with the contest objectives.

The vehicle needs to run smoothly with the water surface and the craft needs to be mostly submerged. However, how submerged exactly is not yet defined and the group is still waiting for a statement from contest supervisors. In addition, the vehicle needs to be fully autonomous for the three courses but can be remotely controlled to the starting point of each course.

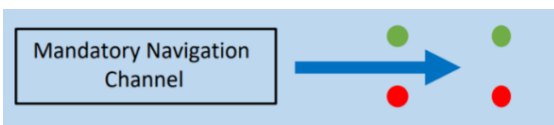
The contest has three main courses and three main categories which will be the base of the requirements. In general, on each course one can score points on thrust, thrust relative to weight and design. The three courses are named Course Alpha, Bravo and Charlie.

Task scoresheet and course overview below.

Parameters	Points
ASV + UAV weight > 140 lbs.	Disqualified!!!
140 lbs > ASV + weight > 110	-250 - 5*(w - 110)
110 lbs > ASV + weight > 70	2*(110 – w)
ASV weight ≤ 70 lbs	80 + (70 - w)
Dimensions greater than: - three feet of width or - three feet of height - six feet of length	Disqualified!!!
Thrust (t) vs weight (w)	*100*(t / w)*

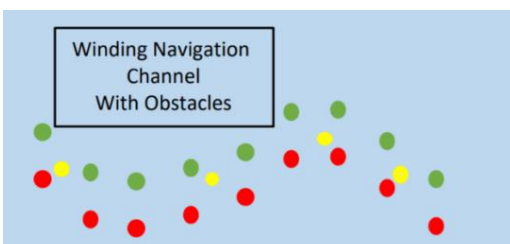
Course Alpha

The ASV needs to be able to manoeuvre in a straight line with buoys on each side.



Course Bravo

The ASV needs to be able to manoeuvre between buoys in a curve like course avoiding obstacles.



Course Charlie

This course is not yet completely defined, however, the supervisors mentioned that there will be a straight course like in Course Alpha with an advanced/dynamic obstacle rotating in the middle of the course.

5.3 Planned procedure for development – method

The group has decided on using Scrum as their main work methodology due to the project's large focus on software development and group members previous experience. As it is most common to use sprint length of two weeks, the group find this fitting because supervisor meetings are also scheduled to be every other week alongside the process report.

The Scrum Master role and its responsibilities will be rotated on by the group members interchanging every sprint to ensure that one does not feel to burdened by the responsibilities. Retrospect meetings will be held as a part of the Friday-meeting at the end of each sprint.

To mention, this project also includes hardware development. The group will bend the traditional way of executing a scrum routine to better fit this project. Exactly how this version of scrum will be executed is not yet defined, however, the group has decided to take it as it comes because of the unconventional task/subtasks that may emerge.

5.4 Gathering of information – gathered and planned

The project is in collaboration with NTNU. NTNU provides knowledge within the project and gives the group five supervisors to consult with. While writing the pre project the group has obtained relevant information of this project. Critical thinking is important to obtain the best suitable information.

Since there is a lot of information on the internet, the group must choose what type of information we can rely on. Academic articles and journals with good content and reliable references, authors and publisher are what the group find trustworthy.

Another way to gather information is through online sites like Stack Overflow. The greater answer the tutors gives the better score they'll have, and better credibility, which the group is looking for. Copied code from Stack Overflow will be documented, but code inspired from non-copyrighted solutions will not be documented.

All type of information that has been gathered will be referenced in the project and well documented.

5.5 Assessment – Analysis of risk

Risk assessment is done to create awareness of risks which can impact the result of the project. Factors which can have a negative impact on the result is:

Risk factors	Possible measures
1. Poor planning	Use sufficient time in the planning process, update the plan regularly
2. Disease	Eat healthy, workout regularly, take breaks to clear the head, ensure a good mental state and get enough hours of sleep
3. Damaging components	Buy components with good quality, research thoroughly before testing. Conduct dry tests
4. Communication problems within the group	Everyone must be at school during the core work hours, have weekly process-meetings and daily stand-up meetings
5. Group individuals lack of contribution	Ensure a good work environment and keep a good tone in the group. Complement and motivate each other.
6. Unable to perform necessary tests due to bad weather	Perform some test in pools, ensure correct behaviour in a different environment than in water e.g. test sensors and camera in a controlled environment.
7. Overestimating qualifications/skills	Do proper research in the planning phase, ask supervisors for guidance
8. Insufficient budget	Create a detailed budget and allow few deviations from the budget

Probability	Severity: Acceptable	Severity: Tolerable	Severity: Generally unacceptable
Unlikely	**8.**	**4.**	**5.**
Possible		**6. 2.**	**3. 1.**
Probable		**7.**	

Risk impact
LOW
MEDIUM
HIGH

5.6 Main activities in further work

The following activities are not in chronological order, some activities will overlap. The attached Gantt-diagram "Gantt - Progression plan.pdf" will act as a more detailed progression plan.

I. Development of bachelor report
II. Research ASV: existing solutions, surface vehicles and maritime propulsion systems
III. Design schematics and diagrams for hard- and software
IV. Selection and acquisition of hardware
V. Software development
VI. Assembling hardware
VII. Software integration
VIII. Testing
IX. Calibration
X. Documentation
XI. Meetings
XII. Pre project report
XIII. HMS

5.7 Progress plan – project management

5.7.1 Master plan

Progress plan has been developed in Instagantt using Asana and can be seen in Attachment 3, "Gantt - Progression plan.pdf". This does not include software development tasks/subtasks, these will be found in the scrum backlog once a minimum value product description has been decided on and the design process is done. In addition, there is no information about boat/hull creation because the group plan to cooperate with another group of students with competence in this discipline.

5.7.2 Management utilities

- Asana with Instagantt for project planning and time scheduling.
- Asana for Scrum methodology
- LaTeX for writing the report with the template from Master Thesis by NTNU
- SharePoint and OneDrive for filesharing
- Google drive for filesharing

5.7.3 Development utilities

Software development utilities will be decided on after or during the design process. As the group has experience with both Python and Java, it is most likely that one of the two will be chosen as the main development language and IntelliJ or PyCharm for IDEA.

The group has three labs each serving a unique purpose available to them. The computer lab mentioned in 4.3 will act as the main development lab for software, solution design and project planning. L160/Electronical lab and will be in disposal on demand. As of now, the group plan to use this lab for prototyping and assembly of electronical parts. Lastly, the heavy lab will be used to assemble hardware and conduct dry testing for both the prototype and final product.

The group plan to use the needed tools/utilities within these labs, however, some tools/utilities demands clearance or approval from supervisor Anders Sætermoen.

5.7.4 Internal control – evaluation

To strengthen internal control within the group the following procedures, amongst others, should be followed; Segregation of duties, identification of risks, daily stand-up meetings, weekly status meetings, biweekly supervisor meetings, periodically reports, promptly error corrections, reviewing and approving changes.

The segregation of duties will be done using a Gantt chart. By using a Gantt chart, we will display a timeline for the project along with the different phases and subphases that are a part of the project, including which group member that have the phase covered. A separate bar will be used to illustrate when each phase starts and ends.

By identifying the risks and the possible measures to the risks around the project at an early stage the process through the project will be more effortless. The risks and possible measures are covered in 5.5.

By having regular meetings within the project group, the workflow will be optimal due to better planning for the given period, either for the day or for the week. The regular meetings for the project group will be the daily stand-up meetings and the weekly meetings for the project group members, and the biweekly meetings with the supervisors. More information about these meetings can be found in 7.1.

By having periodically reports, we can use these to reflect on the given period`s workflow and from there look for improvements in the workflow. The different periodical reports can be found in 7.2.

By using Scrum as the main work methodology, as talked about in 5.3, the group will handle promptly error corrections, review and approve changes.

5.8 Decisions – decision-making process

In the pre-project the decisions and framework of the project is done in cooperation with the supervisors. Major changes in the project will first be discussed within the group. In case of uncertainty the group will discuss this with the supervisors.

The group will use a step-to-step process to help the group make deliberate and thoughtful decisions by organizing relevant information and defining alternatives. This strategy will be used during the whole project.

6 DOCUMENTATION

6.1 Reports and technical documents

- Electrical drawings
- Software diagrams
- Bachelor thesis
- User manuals

7 PLANNED MEETINGS AND REPORTS

7.1 Meetings

7.1.1 Meetings with supervisors

A meeting between supervisors and group members will be held every other week. The topic of discussion will be based around the process reports from 7.2.1. The purpose of this meeting is to enlighten the supervisors on the progression of the group and to get input from experienced professionals on how to move forward. In addition, the supervisors will help the group adhere to the syllabus and requirements of the project to get a better grade and understanding of the link between disciplines.

7.1.2 Project meetings

The group has planned to use daily stand-up meetings as a tool to gain perspective and overview. These meetings will be held by the project manager both at the start and the end of the workday. Each team member will give a brief summary of what they have been working on, what they will be working on and what obstacles they have or foresee.

Each Friday an hour-long status meeting will be held by the project manager to track the progress of the group and discuss how we should move forward.

7.2 Periodical reports

7.2.1 Progress reports (incl. milestone)

There will be written a report at the end of every week. The report will contain the progress of the project, which problems were met and other relevant topics for that week. Every second week there will be made a report as a summary of the two previous weeks to be presented to the supervisors. These reports will be used for discussion in the meetings with the supervisors and to give the group a better overview on the progress.

8 PLANNED DEVIATION MANAGEMENT

If the progress of the project were to get behind planned schedule, the project manager has the responsibility to gather the group in an extraordinary meeting to discuss how and why the schedule has not been met. The group will then revise the original plan such that the main goal still is achieved. The deviation will be categorised based on its impact on the project. Minor incidents can be ignored as they usually do not affect the products quality or function. Major deviation will affect the products quality and function; thus, they should be handled with high priority. Critical deviation is a major deviation which could stop the project from being able to complete.

9 EQUIPMENT/REQUIREMENTS FOR IMPLEMENTATION

9.1 Hardware

- Boat
- Camera
- Other visual sensors e.g. lidar
- Electrical motors
- Microcontroller (s)
- Instruments needed for the motor control
- Instruments needed for the sensors and camera
- Battery
- Battery controller
- Information exchange for manual control (4G, radio, RC etc.)
    - Transmitter
    - Receiver
- Junction box
- Green, red and yellow light for visual feedback

10 REFERENCES

Attachment 1 - RoboBoat 2020 Rules and Task Description V1.pdf

Attachment 2 - RoboBoat 2019 Rules and Task Description_v2.pdf

Attachment 3 – Gantt - Progression plan.pdf

Image 1 – Task Scoresheet

Image 2 – Course Alpha

Image 3 – Course Bravo

Image 4 – Course Charlie

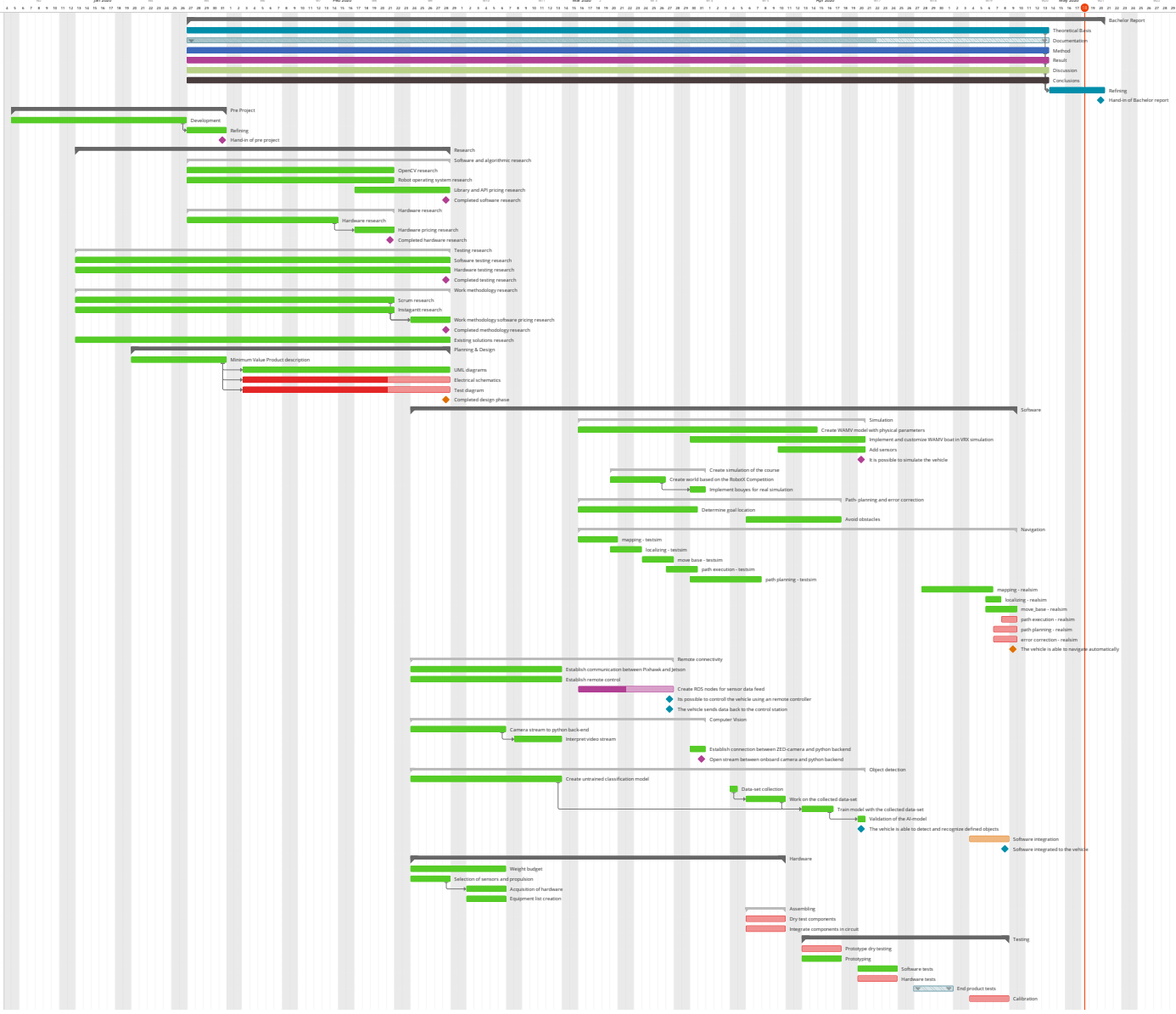
Appendix B

Gantt diagram

Nautilus

Read-only view, generated on 18 May 2020

	ACTIVITIES	ASSIGNEE	EH	START	DUE	%
	Bachelor Report		-	**27/Jan**	**20/May**	**99%**
1	Theoretical Basis	Vic All	-	27/Jan	13/May	100%
	Documentation	Vic All	-	27/Jan	13/May	80%
3	Method	Vic All	-	27/Jan	13/May	100%
4	Result	Vic All	-	27/Jan	13/May	100%
5	Discussion	Vic All	-	27/Jan	13/May	100%
6	Conclusions	Vic All	-	27/Jan	13/May	100%
7	Refining	Vic All	-	14/May	20/May	100%
8	Hand-in of Bachelor report	Vic All	-	20/May	20/May	100%
	Pre Project		-	**05/Jan**	**31/Jan**	**100%**
10	Development	Vic All	-	05/Jan	26/Jan	100%
11	Refining	Vic All	-	27/Jan	31/Jan	100%
12	Hand-in of pre project	Vic All	-	31/Jan	31/Jan	100%
	Research		-	**13/Feb**	**28/Feb**	**100%**
	Software and algorithmic re...	Yrian Hoedemaker	-	27/Jan	28/Feb	100%
15	OpenCV research	Mikel Aanning	-	27/Jan	21/Feb	100%
16	Robot operating system...	Yrian Hoedemaker	-	27/Jan	21/Feb	100%
17	Library and API pricing r...	Mikel Aanning	-	17/Feb	28/Feb	100%
18	Completed software re...	Vic All	-	28/Feb	28/Feb	100%
	Hardware research	Markus Holt	-	27/Jan	21/Feb	100%
20	Hardware research	Markus Holt	-	27/Jan	14/Feb	100%
21	Hardware pricing resea...	Markus Holt	-	17/Feb	21/Feb	100%
22	Completed hardware re...	Markus Holt	-	21/Feb	21/Feb	100%
	Testing research	Sigurd Dölen	-	13/Jan	28/Feb	100%
24	Software testing resear...	Sigurd Dölen	-	13/Jan	28/Feb	100%
25	Hardware testing resea...	Sigurd Dölen	-	13/Jan	28/Feb	100%
26	Completed testing rese...	Sigurd Dölen	-	28/Feb	28/Feb	100%
	Work methodology research	Mikel Aanning	-	13/Jan	28/Feb	100%
28	Scrum research	Mikel Aanning	-	13/Jan	21/Feb	100%
29	Instagantt research	Mikel Aanning	-	13/Jan	21/Feb	100%
30	Work methodology soft...	Mikel Aanning	-	28/Feb	28/Feb	100%
31	Completed methodolog...	Mikel Aanning	-	28/Feb	28/Feb	100%
32	Existing solutions research	Vic All	-	13/Jan	28/Feb	100%
	Planning & Design		-	**20/Jan**	**28/Feb**	**99%**
34	Minimum Value Product des...	Vic All	-	20/Jan	31/Jan	100%
35	UML diagrams	Yrian Hoedemaker	-	13/Feb	28/Feb	100%
36	Electrical schematics	Markus Holt	-	03/Feb	28/Feb	75%
37	Test diagram	Sigurd Dölen	-	03/Feb	28/Feb	75%
38	Completed design phase	Vic All	-	28/Feb	28/Feb	100%
	Software		-	**28/Feb**	**09/May**	**87%**
	Simulation	Markus Holt	-	16/Mar	20/Apr	100%
40	Create WAMV model wi...	Markus Holt	-	16/Mar	14/Apr	100%
42	Implement and customi...	Markus Holt	-	30/Mar	20/Apr	100%
43	Add sensors	Markus Holt	-	15/Apr	20/Apr	100%
44	It is possible to simulate...	Markus Holt	-	20/Apr	20/Apr	100%
	Create simulation of th...	Mikel Aanning	-	20/Mar	31/Mar	100%
46	Create world based...	Mikel Aanning	-	20/Mar	26/Mar	100%
47	Implement bouyes f...	Mikel Aanning	-	30/Mar	31/Mar	100%
	Path- planning and error co...	Unassigned	-	16/Mar	15/Apr	100%
49	Determine goal location	Unassigned	-	16/Mar	30/Mar	100%
50	Avoid obstacles	Unassigned	-	06/Apr	15/Apr	100%
	Navigation	Unassigned	-	16/Mar	09/May	67%
52	mapping - testsim	Sigurd Dölen	-	16/Mar	20/Mar	100%
53	localizing - testsim	Yrian Hoedemaker	-	20/Mar	23/Mar	100%
54	move base - testsim	Yrian Hoedemaker	-	23/Mar	27/Mar	100%
55	path execution - testsim	Yrian Hoedemaker	-	27/Mar	30/Mar	100%
56	path planning - testsim	Yrian Hoedemaker	-	30/Mar	31/Mar	100%
57	mapping - realsim	Markus Holt	-	28/Apr	06/May	100%
58	localizing - realsim	Markus Holt	-	06/May	07/May	100%
59	move_base - realsim	Unassigned	-	08/May	09/May	100%
60	path execution - realsim	Unassigned	-	08/May	09/May	0%
61	path planning - realsim	Unassigned	-	07/May	09/May	0%
62	error correction - realsim	Unassigned	-	07/May	09/May	0%
63	The vehicle is able to na...	Unassigned	-	09/May	09/May	0%
	Remote connectivity	Unassigned	-	23/Feb	27/Mar	75%
65	Establish communicatio...	Markus Holt	-	24/Feb	13/Mar	100%
66	Establish remote control	Markus Holt	-	24/Feb	13/Mar	100%
67	Create ROS nodes for s...	Markus Holt	-	16/Mar	27/Mar	50%
68	Its possible to control t...	Unassigned	-	27/Mar	27/Mar	100%
69	The vehicle sends data ...	Unassigned	-	27/Mar	27/Mar	0%
	Computer Vision	Unassigned	-	23/Feb	31/Mar	100%
71	Camera stream to pyth...	Mikel Aanning	-	23/Feb	06/Mar	100%
72	Interpret video stream	Mikel Aanning	-	06/Mar	13/Mar	100%
73	Establish connection be...	Mikel Aanning	-	30/Mar	31/Mar	100%
74	Open stream between ...	Unassigned	-	31/Mar	31/Mar	100%
	Object detection	Unassigned	-	23/Feb	20/Apr	100%
76	Create untrained classi...	Sigurd Dölen	-	23/Feb	13/Mar	100%
77	Data-set collection	Mikel Aanning	-	03/Apr	03/Apr	100%
78	Work on the collected d...	Mikel Aanning	-	06/Apr	15/Apr	100%
79	Train model with the col...	Sigurd Dölen	-	13/Apr	16/Apr	100%
80	Validation of the AI-mod...	Unassigned	-	20/Apr	20/Apr	100%
81	The vehicle is able to de...	Unassigned	-	20/Apr	20/Apr	100%
82	Software integration	Unassigned	-	02/May	08/May	0%
83	Software integrated to the v...	Unassigned	-	08/May	08/May	0%
	Hardware		-	**26/Feb**	**15/Apr**	**80%**
85	Weight budget	Markus Holt	-	26/Feb	06/Mar	100%
86	Selection of sensors and pr...	Markus Holt	-	26/Feb	28/Feb	100%
87	Acquisition of hardware	Vic All	-	02/Mar	06/Mar	100%
88	Equipment list creation	Markus Holt	-	02/Mar	06/Mar	100%
	Assembling	Markus Holt	-	06/Apr	15/Apr	0%
90	Dry test components	Unassigned	-	06/Apr	10/Apr	0%
91	Integrate components i...	Unassigned	-	06/Apr	10/Apr	0%
	Testing		-	**13/Apr**	**08/May**	**19%**
93	Prototype dry testing	Unassigned	-	13/Apr	13/Apr	0%
94	Prototyping	Unassigned	-	14/Apr	17/Apr	100%
95	Software tests	Unassigned	-	20/Apr	22/Apr	100%
96	Hardware tests	Unassigned	-	20/Apr	22/Apr	0%
	End product tests	Unassigned	-	27/Apr	01/May	0%
98	Calibration	Unassigned	-	02/May	08/May	0%



Appendix C

Hour List

Hour list

	Markus Holt		Yrian Hovde Øksne		Sigurd Odden		Mikal Aanning	
Dato	Timer	Kommentar	Timer	Kommentar	Timer	Kommentar	Timer	Kommentar
7/1/2020	4	Pre project	4	Pre project	4	Pre project	4	Pre project
8/1/2020	2	Research USV/ASV						
9/1/2020	5	Pre project	5	Pre project	5	Pre project	5	Pre project
10/1/2020	5	Meeting with supervisors and pre project	5	Meeting with supervisors and pre project	5	Meeting with supervisors and pre project	5	Pre project and Scrum Research
11/1/2020	2	Pre project	2	Pre project	2	Pre project	3	Pre project
12/1/2020								
13/1/2020		Lecture in Industry 4.0		Lecture in IF3001141		Lecture in Industry 4.0		Lecture in Industry 4.0
14/1/2020	6	Creating Gantt diagram and hour list	6	Made Gantt-diagram	6	Pre project	6	Pre project and Scrum Research
15/1/2020		Visit at Ekornes in Industry 4.0		Lecture in IF3001141		Visit at Ekornes in Industry 4.0		Visit at Ekornes in Industry 4.0
16/1/2020		Lecture in Industry 4.0	8	Research navigation solutions		Lecture in Industry 4.0		Lecture in Industry 4.0
17/1/2020	8	Meeting with Ship Design and Refining Pre Project	8	Meeting with Ship Design and refining pre project report	6,5	Meeting with Ship Design and refining pre-Project	6	Meeting with Ship Design and Refining Pre Project
18/1/2020								
19/1/2020								
20/1/2020	4	Component research		Lecture in IF3001141	4	Testing research and miscellaneous	2	Pre-planning and software obtaiment of Jira
21/1/2020	4	MVP description	4	MVP description	4	MVP description	4	Project meeting and MVP clearification
22/1/2020	3	MVP description		Lecture in IF3001141	3	MVP description	3	MVP completion
23/1/2020	3	Deployment diagram and backlog	6	Made deployment diagram	6	Deployment diagram and backlog	6	Deployment diagram and Backlog
24/1/2020	7	Research ROS and navigation. Progress meeting and writing progress report	7	Process meeting and report + ROS course	7	Research case testing and progress meeting	7	Remote controll research and process meeting
25/1/2020								
26/1/2020								
27/1/2020		Lecture in industry 4.0		Lecture in IF3001141				Lecture in Industry 4.0
28/1/2020		Lecture in industry 4.0	7	ROS course				Lecture in Industry 4.0
29/1/2020		Lecture in industry 4.0		Lecture in IF3001141			1	Lecture in Industry 4.0 + Start-up with OpenCV
30/1/2020	4	ROS course	7	ROS course	7	OpenCV with colour detection image face detection	6	Basic OpenCV with colour detection
31/1/2020	4	Process meeting and ROS tutorial	4	process meeting + ROS course	3	Process meeting and openCV with multiple colour detection	3	Process meeting and OpenCV with colour detection
1/2/2020					1	OpenCV with colour detection and video face detection		
2/2/2020								
3/2/2020	3	ROS tutorial		Lecture in IF3001141	5	OpenCV tutorial	5	OpenCV with colour detection
4/2/2020	4	ROS tutorial	6	ROS course	6	OpenCV tutorial	5	OpenCV with colour tracking
5/2/2020	4	ROS tutorial		Lecture in IF3001141	5	OpenCV tutorial with object detection	5	OpenCV with colour tracking and position output
6/2/2020	3	ROS. Work from 12	8	ROS course	5	OpenCV tutorial with AI	6	OpenCV with multiple colour tracking and position output
7/2/2020	6	Process meeting, creating weight- and cost budget	6	Process meeting and report + ROS course				
8/2/2020								
9/2/2020								
10/2/2020	5	Component research, weight- and cost budget		Lecture in IF3001141			5	OpenCV contour shape detection
11/2/2020	5	Report and component research	5	Made scrum backlog	5	Bachelor report writing and AI tutorial	5	Bachelor report writing on OpenCV
12/2/2020		Lecture in Industry 4.0		Lecture in IF3001141		Lecture in Industry 4.0		Lecture in Industry 4.0
13/2/2020		Lecture in Industry 4.0	6	wrote documentation on bachelors report		Lecture in Industry 4.0		Lecture in Industry 4.0
14/2/2020		Lecture in Industry 4.0	6	Development remote connecticity and control		Lecture in Industry 4.0		Lecture in Industry 4.0
15/2/2020								
16/2/2020								
17/2/2020	5	Designing power system		Lecture in IF3001141	5	Test cases	4	Bachelor report writing on OpenCV
18/2/2020	4	MVP clearification and updating backlog	5	MVP clearification and updating backlog + set up jetson TX2	5	MVP clearification and started on designing AI-model	4	Project meeting and MVP clearification
19/2/2020	5	Design of circuit and dimensioning power system		Lecture in IF3001141	6	Designing AI-model	6	Begin with OpenCV + Keras
20/2/2020	3	Design circuit. Work from 12	8	continue development on remote connectivity and control	2	Process meeting and meeting with partners	6	OpenCV + Keras
21/2/2020	2	Process meeting and meeting with partners	2	process meeting and meeting with partners			2	Process meeting and meeting with partners
22/2/2020								
23/2/2020					2	Case testing		
24/2/2020	6	Getting started with ROS on Jetson TX2		Lecture in IF3001141	6	Finishing up case testing and started on AI program	7	OpenCV on Odoroid - Start up
25/2/2020	7	Meeting with supervisors. Flashing Jetson TX2 and installing drivers	7	Meeting with supervisors, ROS navigation	7	Meeting with supervisors and R-CNN setup and installation	7	OpenCV on Odoroid - Installation
26/2/2020	4	TX2		Lecture in IF3001141	7	R-CNN	7	OpenCV on Odoroid - Visual
27/2/2020				Exam preperations in IF3001141	5	R-CNN	7	OpenCV on Odoroid - Visual Conclusion and dataset collection planning
28/2/2020	2	Report		Exam preperations in IF3001141				
29/2/2020	3	Report		Exam preperations in IF3001141				
1/3/2020				Exam preperations in IF3001141				
2/3/2020	6	Electrical schematic		Exam preperations in IF3001141	7	Research on AI - designing test schematic of AI	7	OpenCV on Jetson TX2 - installation
3/3/2020	7	Electrical schematic		Exam preperations in IF3001141	7	Deciding and planning of AI installing	7	OpenCV on Jetson TX2, bouye purchase and painting.
4/3/2020		Lecture in Industry 4.0		Exam preperations in IF3001141		Lecture in Industry 4.0	3	Lecture in Industry 4.0 - Jetson TX2 Flashing for CUDA
5/3/2020		Lecture in Industry 4.0		Exam preperations in IF3001141		Lecture in Industry 4.0	3	Lecture in Industry 4.0 - Jetson TX2 Flashing for CUDA
6/3/2020		Lecture in Industry 4.0		Exam preperations in IF3001141	1	Meeting with partners and lecture in Industry 4.0		Lecture in Industry 4.0
7/3/2020				Exam preperations in IF3001141				
8/3/2020				Exam preperations in IF3001141				
9/3/2020	7	Setting up Pixhawk		Exam preperations in IF3001141	7	AI - Yolo installation and implementation	7	OpenCV on Jetson TX2 installation and Visual Testing
10/3/2020	7	Communication between Jetson and Pixhawk		Exam preperations in IF3001141	7	AI - Yolo installation and implementation	7	Instagantt Update
11/3/2020	7	Communication between Jetson and Pixhawk		Exam preperations in IF3001141	5	AI - Yolo model finished -	7	OpenCV on Jetson TX2 Optimilization and Instagantt Update
12/3/2020	4	Writing ROS node to feed senor data from Pixhawk, pandemic outbreak		Exam preperations in IF3001141	4		4	Startup with cvBridge to ROS - short day due to pandemic outbreak
13/3/2020	5	Writing ROS node to feed senor data from Pixhawk		Exam preperations in IF3001141				
14/3/2020	5	Writing ROS node to feed senor data from Pixhawk		Exam preperations in IF3001141				
15/3/2020	4	Updating Gantt and backlog. Writing on report		Exam preperations in IF3001141				
16/3/2020	4	Process meeting, simulation	1	EXAM IN IF3001141 + standup + process meeting	4	Process meeting	4	Reflection of using cvBridge with ROS and project meeting
17/3/2020	5	Supervisor meeting, surface vessel hydrodynamics	1	Supervisor meeting + process meeting + flight home	4	Meeting with supervisors	4	Meeting with supervisors and planning for data collection
18/3/2020	5	Surface vessel kinematics	6	Research path planning + plan development for path planning	7	Installation of ROS on Odroid XU4 and completeting tutorials	6	Installation of ROS and researching possibilities for object mapping
19/3/2020		Travelled home	5	Started configuring ROS Development with turtlebot and environment	6	Installed ROS on VMware because of problems with Odroid XU4	6	Basic guides on ROS
20/3/2020	5	Implemented usv sim with ros in gazebo, report writing	5	Made mapping solution for turtlebot	6	ROS mapping and localization with ROS Development Studio	6	Research and guides on Gazebo and rviz
21/3/2020								
22/3/2020								
23/3/2020	6	usv sim incompatible, changed to using osrf/vrx simulation, report writing	6	made localization solution for turtlebot, report writing	6	ROS mapping and localization with ROS Development Studio	6	Research on python script to map objects in Gazebo
24/3/2020	5	simulation, report writing	6	met with supervisors and report writing, report writing	6	Meeting with supervisors and report writing	6	Installation of Gazebo on Windows and report writing
25/3/2020	5	simulation, report writing	5	started on autonomous navigation for turtlebot - met alot of errors, report wri	6	Giving navigation commands with Rviz	6	Simulated enviroment for ASV and report writing
26/3/2020	5	simulation, report writing	5	finished giving navigation commands, report writing	6	Navigation in ROS DS	6	Refining the simulated enviroment
27/3/2020	5	simulation, report writing	6	started on autonomous navigation with pre-planned path for turtlebot, report	6	RTAB mapping with Kinect Camera in ROS DS	5	Writing on the Bachelor Thesis
28/3/2020	2	report						
29/3/2020	2	report						
30/3/2020	5	simulation	6	continued making python cript with pre-planned path navigation, report writi	6	RTAB mapping with Kinect Camera in ROS DS	6	Implementation of competition course in Gazebo
31/3/2020	5	supervisor meeting, simulation	5	Met with supervisors and continued making pre-planned path navigation, rep	6	Meeting with supervisors and GPS implementation	6	Implementation of competition course in Gazebo
1/4/2020	5	simulation	6	finished making the python script for pre-planned navigation, report writing	6	GPS implementation in ROS DS and report writing	6	Data-set collection script with ZED 2
2/4/2020	5	simulation	5	report writing, report writing	6	GPS implementation in ROS DS and report writing	5	Writing on the Bachelor Thesis
3/4/2020	5	simulation	5	fusion of RTAB-mapping and navigation and report writing, report writing	5	Fusion of RTAB-mapping and navigation and report writing	5	Writing on the Bachelor Thesis
4/4/2020							3	Data-set collection
5/4/2020								
6/4/2020	4	Report writing	1	some report writing, mostly easter holiday vacation	4	Report writing	5	Labeling data-set
7/4/2020	4	Report writing	1	some report writing, mostly easter holiday vacation	3	Report writing	4	Labeling data-set
8/4/2020	3	report writing	1	some report writing, mostly easter holiday vacation	3	Report writing		
9/4/2020	2	report writing	1	some report writing, mostly easter holiday vacation	2	Report writing	2	Writing on the Bachelor Thesis
10/4/2020			1	some report writing, mostly easter holiday vacation				
11/4/2020			1	some report writing, mostly easter holiday vacation				
12/4/2020			1	some report writing, mostly easter holiday vacation				
13/4/2020			1	some report writing, mostly easter holiday vacation				
14/4/2020	6	Supervisor meeting, progress meeting and simulation	6	Meeting with supervisors, progress meetings and trying to download and inst	6	Meeting with supervisors and prepare AI training	4	Writing on the Bachelor Thesis
15/4/2020	6	simulation, issues with docker on jetson	6	tried to install VRX simulation, met nvidia docker error, discussed problem	9	Solved problem with annotation files not matching images	4	Writing on the Bachelor Thesis
16/4/2020	6	simulation, managed to implement the simulation on my own computer	6	report writing, tried various solutions for installing VRX simulation	6	Began training with smaller set of images and annotation to test AI	4	Writing on the Bachelor Thesis
17/4/2020	6	simulation	5	report writing, progress meeting	7	Finished annotation for all classses and began training the AI-model		
18/4/2020	2	report writing						
19/4/2020	5	report writing						
20/4/2020	7	Implementing navigation in simulation	6	report writing, progress meeting	6	Report writing and creating mAP for the AI-model	4	Writing on the Bachelor Thesis
21/4/2020	8	Error in the configuration files of the simulation, debugging	6	Report writing and editing patrolling/maneuvering script	6	Report writing	4	Writing on the Bachelor Thesis
22/4/2020	7	Error in the configuration files of the simulation, debugging	6	Report writing and editing patrolling/maneuvering script	6	Report writing	4	Writing on the Bachelor Thesis
23/4/2020	6	Report writing and debugging simulation configuration files	5	Report writing	6	Report writing	3	Writing on the Bachelor Thesis
24/4/2020	6	Report writing and debugging simulation configuration files	5	Report writing	5	Report writing	6	Writing on the Bachelor Thesis
25/4/2020							6	Writing on the Bachelor Thesis
26/4/2020	5	Report writing			7	Report writing	6	Writing on the Bachelor Thesis
27/4/2020	7	Fixed error in simulation, report writing	5	Report writing	6	Report writing	4	Writing on the Bachelor Thesis
28/4/2020	6	Began implementing Rtabmap in simulation	6	Report writing	6	Report writing	4	Writing on the Bachelor Thesis
29/4/2020	6	Rtbamap	5	Report writing	5	Report writing	3	Writing on the Bachelor Thesis
30/4/2020	6	issue with rtabmap, debugging	6	Report writing	7	Report writing	6	Writing on the Bachelor Thesis
1/5/2020	6	issue with rtabmap, debugging	7	Report writing	8	Report writing	6	Writing on the Bachelor Thesis
2/5/2020	5	Began implementing mapping	7	Report writing	5	Report writing	6	Writing on the Bachelor Thesis
3/5/2020	5	Mapping in Rtabmap, issue with producing map	7	Report writing	4	Report writing	6	Writing on the Bachelor Thesis
4/5/2020	9	Mapping in Rtabmap, successfully producing map	6	report writing	6	Report writing	4	Writing on the Bachelor Thesis
5/5/2020	8	Supervisor meeting and Rtabmap	7	Report writing	5	Report writing and meeting with supervisors	4	Writing on the Bachelor Thesis
6/5/2020	8	Mapping, tried Gmapping without luck	6	Report writing	5	Report writing		Studying to exam
7/5/2020	7	Mapping Rtabmap	8	Report writing	7	Report writing		Studying to exam
8/5/2020	7	Mapping Rtabmap	7	Report writing	7	Report writing		Studying to exam
9/5/2020	7	Mapping Rtabmap	6	Report writing	3	Report writing		Studying to exam
10/5/2020	7	Mapping Rtabmap	7	Report writing	3	Report writing		Studying to exam
11/5/2020	8	Report	8	Report writing	4	Report writing		Studying to exam
12/5/2020	9	Report	8	Report writing	3	Report writing		Studying to exam
13/5/2020	9	report	7	Report writing	2	Exam in IE203412 Signalbehandling and report writing		Exam in IE203412 Signalbehandling
14/5/2020	8	report	8	Report writing	9	Report writing	9	Writing on the Bachelor Thesis
15/5/2020	9	report	9	Report writing	9	Report writing	9	Writing on the Bachelor Thesis
16/5/2020	8	report	8	Report writing	9	Report writing	9	Writing on the Bachelor Thesis
17/5/2020	8	Video presentation	9	Video presentation and report writing	9	Video presentation and report writing	9	Video presentation
18/5/2020	8	Video presentation	9	Video presentation and report writing	9	Video presentation and report writing	9	Video presentation
19/5/2020	8	Video presentation					9	Video presentation
20/5/2020		Hand in and presentation of the project						
	Total work hours for Markus Holt:		**Total work hours for Yrian Hovde Øksne:**		**Total work hours for Sigurd Odden:**		**Total work hours for Mikal Aanning:**	
	505		420		466,5		426	
			Total work hours for the whole project:		1817,5			

Appendix D

MVP

MVP (Minimum Value Product) description

Target release	
Epic	
Document status	DRAFT
Document owner	Yrian Øksne
Designer	Mikal Aanning
Developers	Markus Holt
QA	Sigurd Odden

Goals

Background and strategic fit

Ottar L. Osen has put together a national competition where all participants needs to develop an autonomous surface vehicle. This vehicle must maneuver through several courses in order to collect points. In cooperation with Ottar and NTNU, our bachelor group has been selected to participate in the competition. In order for our group to get the highest grade possible, we need to cover all the syllabus (or most of it) in this project. As an autonomous vehicle needs a hull, and our group has no previous experience with designing one, we have partnered with another group of students who will develop the hull for us. Our responsibility is to design and implement the electrical hardware and develop the software for the ASV.

Assumptions

- We will fund and develop a hull in cooperation with partners
- The competition we will participate in is based on rules and courses from RoboBoat 2018/19
- We can steer the ASV remotely to the starting line
- We can map the competition area

Terms

- ASV - Autonomous Surface Vehicle
- Frame of reference - a set of criteria or stated values in relation to which measurements or judgement can be made
- TCP - Transmission Control Protocol, one of the main protocols for sending data packets
- JSON format - JavaScript Object Notation is an open-standard file format or data interchange format that uses human-readable text to transmit data objects consisting of attribute-value pairs and array data types
- SLAM - Simultaneous localization and mapping
- OpenCV - Open source computer vision
- IMU - Inertial measurement unit
- IP - Ingress Protection
- SIL - Safety Integrity Level
- HTTPS - Hypertext Transfer Protocol Secure
- Hash - Mathematical algorithm that maps data of arbitrary size to a hash of a fixed size. It's designed to be a one-way function, infeasible to invert.

Functional Requirements

#	Title	User Story	Importance	Notes
1	Color tracking	The vehicle should be able to recognize different colors on buoys.	Must Have	• Using AI model made with Keras • Using OpenCV connected to camera
2	Object detection	Obstacles should be detected in order to be placed in a coordinate system.	Must Have	• Using AI model made with Keras • Using Lidar and camera
3	Object recognition	The vehicle should be able to recognize the different objects.	Must Have	• Using AI model made with Keras • Using OpenCV connected to camera

4	Remote connectivity	Connectivity for remote computer storage and remote control.	Must Have	• WiFi
5	Room mapping	The vehicle and obstacles should be mapped in a common coordinate system.	Must Have	• Using inputs from ROS Package Gmapping and Keras AI model
6	Remote control	The user should be able to control the vehicle with a remote controller to the starting line and should be able to turn the vehicle autonomous with simple command.	Must Have	• Joystick controller or another type of controller
7	GUI for testing	The user should be presented with statistical data regarding the vehicles route, actions and registered obstacles/buoys through a graphical user interface.	Nice to have	• Using ROS control center
8	Database storage	The vehicle should send data regarding its route, actions and registered obstacles/buoys for statistical analysis to a remote storage.	Nice to have	• Using remote connectivity, TCP, JSON format
9	Component weight	The vehicle should not weigh more than 63 kg.	Must have	• High thrust-to-weight ratio • Competition requirement
10	Power system	The vehicles components should be powered and properly managed.	Must Have	• Power the thrusters, lidar, micro-controller etc. (Battery)
11	Determine position	The vehicle must determine position in its frame of reference.	Must have	• Using input data from lidar and IMU, outputting a position
12	Plan most effective path	The vehicle must plan the most effective path to some goal location.	Must have	• Using input from room map, fleet position and goal, outputting shortest path
13	Avoid collision	The vehicle must avoid collisions if the occasion rises.	Must have	• Using input from object detector, vehicle position • Output new path
14	Execute on planned path	The vehicle should be able to execute on the most effective path given.	Must have	• Using input from planning algorithm • Outputs motor commands
15	Path error correction	The vehicle needs to adjust its path when it is out of bounds	Must have	• Using Line of Sight or similar algorithm

Non-Functional Requirements

#	Title	User Story	Importance	Notes
1	Security	The system should be able to deny remote connectivity and control without valid password. Secure communication has to be used to transfer data.	Must Have	• It should not be possible to crack user passwords. • Uses HTTPS and MD5 Hash
2	Performance	The system should be able to process data from Lidar and camera simultaneously. The time used to make decisions should not have a negative impact on the maneuverability.	Must Have	• High quality equipment • Satisfying computational power • Intelligent structure of decision making hierarchy
3	Reliability	The safety integrity level (SIL) of the vehicle should be as high as possible to prevent damages. The ingress protection (IP) of the equipment should be at least IP67 to prevent water damages.	Must Have	• High SIL • IP67 or better

4	Maintainability	The vehicle should be designed such that the probability of performing a successful repair action within a given time is high.	Nice to Have	• High hardware management
5	Usability	The vehicle should be intuitive and easy to control.	Must Have	• Everyone should be able to control the ASV.
6	Documentation	The project should be thoroughly documented in the source code, alongside circuit drawings and other diagrams.	Must have	• The project will be delivered with extended documentation
7	Measurability	The system should be able to take accurate measurements for better precision. The hardware should be measurable in case errors appears.	Must Have	• High precision
8	Presentable	The hardware should be well structured and organized.	Nice to have	• High hardware management

Appendix E

BOM

Estimated budget							
Number of components	Component	Name		Price (NOK)	Weight (kg)	Average power consumption (W)	Link
1	Microcontroller	Jetson TX2 Developer kit	NVIDIA	4000	0,085	8	https://developer.nvidia.com/embedded/jetson-tx2-developer-kit
2	Battery	Li-4S-18Ah	Blue Robotics	2995	1,152		https://www.jmrobotics.no/p/lithium-ion-batteri-148v-18ah
1	Camera	Zed 2	Stereolabs	4490	0,124	2	https://www.stereolabs.com/zed-2/
4	Motor controller	ESC	Blue robotics	250	0,016		https://www.jmrobotics.no/p/basic-esc
4	Propulsion	T-200	Blue robotics	2000	0,688	645	https://www.jmrobotics.no/p/t200-thruster
1	Flight controller, Cool-kit	Pixhawk mRo 2.4.6	mRo	2600	0,038	5	https://store.mrobotics.io/product-p/mro-pixhawk1-fullkit-mr.htm
1	GNSS reciever	NEO-M8	mRo, uBlox	part of the kit	0,05		
1	SIK-Telemetry 915/433Hz transmitter and reciever	Telemetry	mRo	part of the kit	0,005		
1	Power module	mRo classic power module	mRo	part of the kit	0,005		
	Planned to buy but not done because the boat was not completed:						
	Junction box	IP67 or higher	Biltema		0,3		
	Cables		School		0,15		
	Fuses						
	Hull	Self made					
	SUM			16335	2,613	660	

Appendix F

Autodrone 2020 Rules and Task Description

Rules and Task Descriptions

Version 0.3, Created Feb 18, 2020

AUTODRONE

The rules herein are based on the RoboBoat Rules Task Description with permission from the RoboBoat International Competition

AutoDrone Rules and Task Descriptions

1 Objective

AutoDrone is a national championship in operation of Autonomous Sea Drones (ASD) established to strengthen the maritime higher education in Norway and as a response to the increasing interest for autonomous operation of surface ships. New technology for smarter ships and new ways of solving sea-based transport more efficiently and cleaner represents challenges and opportunities for the maritime sector. The AutoDrone championship will act as an incubator for developing, testing and benchmarking innovations in the maritime domain.

This vision is achieved by providing a venue and mechanism, whereby practitioners of robotics and maritime autonomy come together at AutoDrone to share knowledge, innovate, and collaboratively advance the technology of ASD systems.

2 AutoDrone Information and Updates

All questions, comments, and suggestions should be posted on the AutoDrone Questions facebook group. Teams are encouraged to actively participate in the online community and monitor it for latest news and updates regarding the AutoDrone Championship.

3 Venue Overview

The championship in 2020 will be held at Strandpromenaden 50, Horten, in conjunction with the Kongsberg Maritime Subsea facilities there. Horten Autonomy Test arena will be used, and the area around the pier will be divided into a single competition course with 4 different missions.

Figure 1 Strandpromenaden 50 Aerial View

4 Team Village

During the competition, each team is provided with a covered workspace. Drone and other equipment will be locked up overnight. Electricity (one outlet) and internet connection (wireless) are available in the workspace.

5 Schedule

The competition is held, rain or shine. The general competition's schedule is available on the AutoDrone website. The schedule is subject to changes due to inclement weather stoppages (lightning, etc.), and safety considerations. It is the Team's responsibility to check the website for the most up to date version of the schedule.

6 Participation and Eligibility Requirements

There is no requirement for teams to be associated with a school or university. Please consider that at least 3 members are needed for safe AutoDrone operations. Faculty, industrial and governmental partners may be used to support the on-site team.

One drone per team may be entered in the competition. Each team must designate a team member as their *team leader*. The team leader is the only person allowed to speak for the team, to request drone deployment, run start, run end, or drone retrieval.

The team leader must be conversationally fluent in Norwegian or English to communicate with AutoDrone staff.

7 Registration and Fees

To participate in the competition, all teams must register via the registration responsible listed at www.autodrone.no and submit the registration fee.

8 Team Deliverables

In addition to the mission tasks, each team must document some of their efforts leading up to the competition by adding media content to the AutoDrone facebook page.

Each team leader is responsible for adhering to the instructions and deadlines listed on the AutoDrone website.

8.1 Facebook contribution

Teams must contribute to the AutoDrone facebook page with the following information:

- Team information (name and team contact information).
- Team member information (name, picture, contact information).
- Media (pictures, video, etc.) taken during development and testing.
- Link to team websites if available.

9 Competition General Requirements

The following is a list of minimal requirements for a sea drone to be permitted access to a course. **Teams that arrive at the competition failing to meet these requirements will not be permitted on the course, until they modify their drone to meet all the requirements.**

9.1 Sea Drone Requirements

- **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 an action camera with an unobstructed view from the front of the drone.

9.2 Interference

Interference with course elements may result in a run termination. Any sea drone entangled in, dragging, pushing or damaging competition elements or the landscape may be deemed interfering.

All drones must stay within the bounds of their assigned course. Any drone leaving its assigned course may be deemed interfering.

9.3 Judges' Decisions

All decisions of the Judges are final.

9.4 Sea Drone Recovery

No team member is allowed in the water any time. Competition officials will be responsible for recovering lost drones. Officials will make all reasonable efforts to recover a lost drone but cannot guarantee that they will be able to do so. All teams recognize that by entering the competition, they risk damage to, or the loss of, their drone. The judges, officials, host and sponsors can take no responsibility for such damage or loss.

10 Weight and Thrust Measurements

Drones are weighed before every deployment.

Thrust is measured after the drone is deployed in the water. The thrust value used is the highest scale reading that is stable for at least two seconds. Teams may opt to repeat their thrust measurement at each deployment.

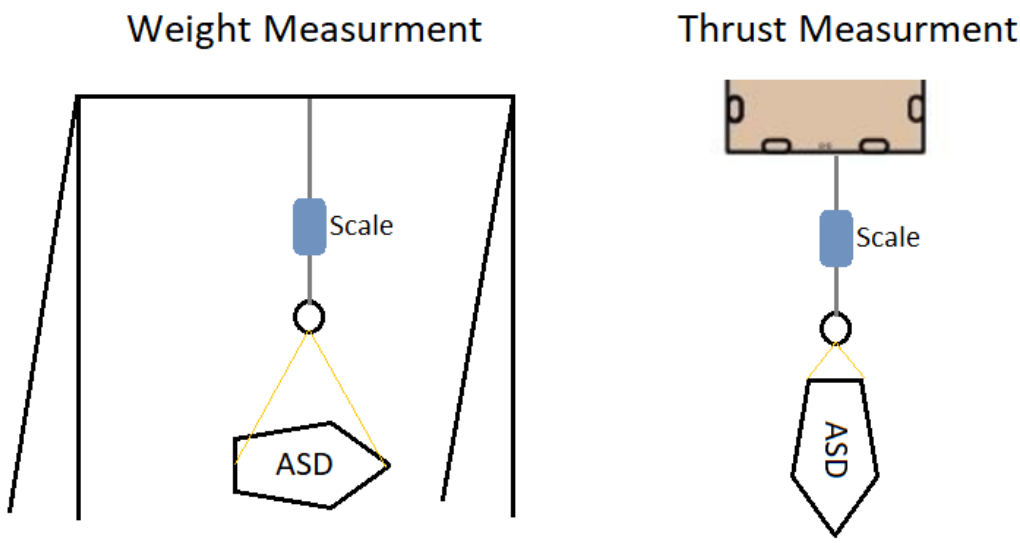


Figure 2 Weight and Thrust Measurement setup

10.1 Task scoresheet

Table 1: Weight and thrust scoresheet

Parameters	Points
ASD weight > 70kg.	**Disqualified**
Dimensions greater than: - 0.9m width or - 0.9m feet of height - 1.8 m of length	**Disqualified**
Thrust (t) vs weight (w)	*100*(t / w)*

11 Mission Tasks

Teams demonstrate advanced autonomous drone behavior through performance of the mission tasks. Mission tasks will be performed according to the championship schedule. One task at a time with one team at a time in a given order. A single GPS position, representing the center of the task's 'entrance' will be provided at the competition (in decimal degree format).

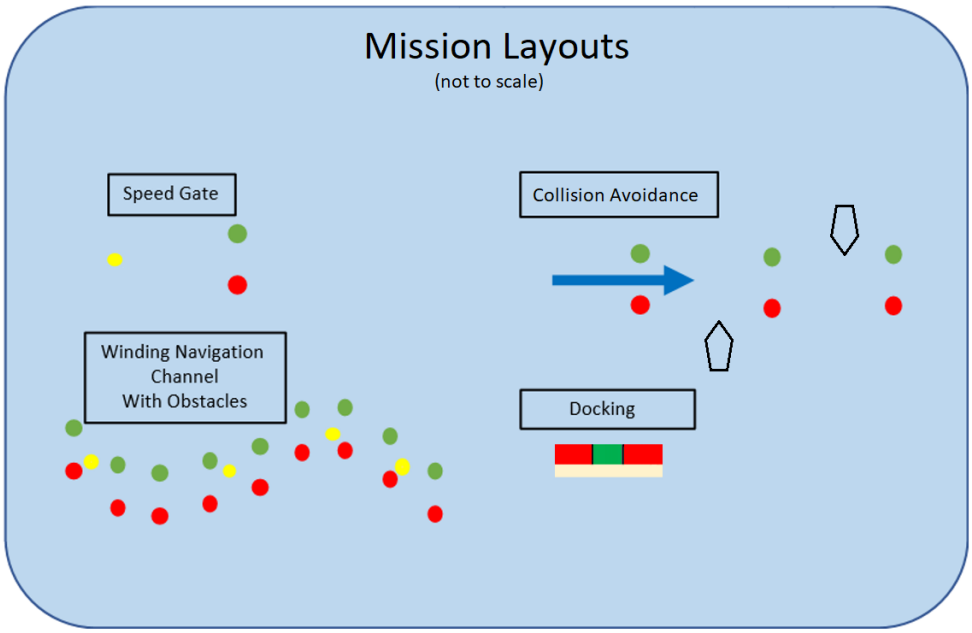
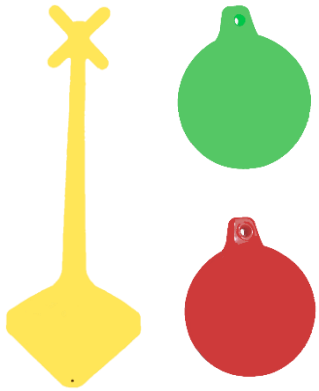


Figure 3 Mission layouts

11.1 Buoys

The championship will use only three types of buoys, two round of equal size and shape but of different color (Ø155 mm), and one yellow beach marker buoy:



Red buoy: Art.nr. 25-017 from Biltema
Green buoy: Art.nr. 25-017 from Biltema spraypainted green (Art.nr. 36-577 from Biltema)
Yellow buoy: from Sommerbutikken

Figure 4 Championship buoys

11.2 Tries

Each team will be able allowed two attempts for each mission task, the time of the attempts will be scheduled.

11.1 Obstacle Channel

Successful completion of the Obstacle Channel task demonstrates the ability to sense and maneuver through a complex path, staying within the defined pathway, and avoiding contact with obstacles along the way.

The drone passes between multiple sets of gates designated by pairs of red and green buoys. The entire drone should pass through all sets of the gates without touching the buoys. To score points for a gate it must be passed correctly one time, no points are scored on subsequent passing of same gate. The drone must also avoid intermittent yellow buoys placed within the pathway described by the location of the red and green pairs of buoys (gates) hitting an obstacle buoy will give penalty points.

This is a timed mission and for every minute spent on the mission points are subtracted from an initial starting number of points.

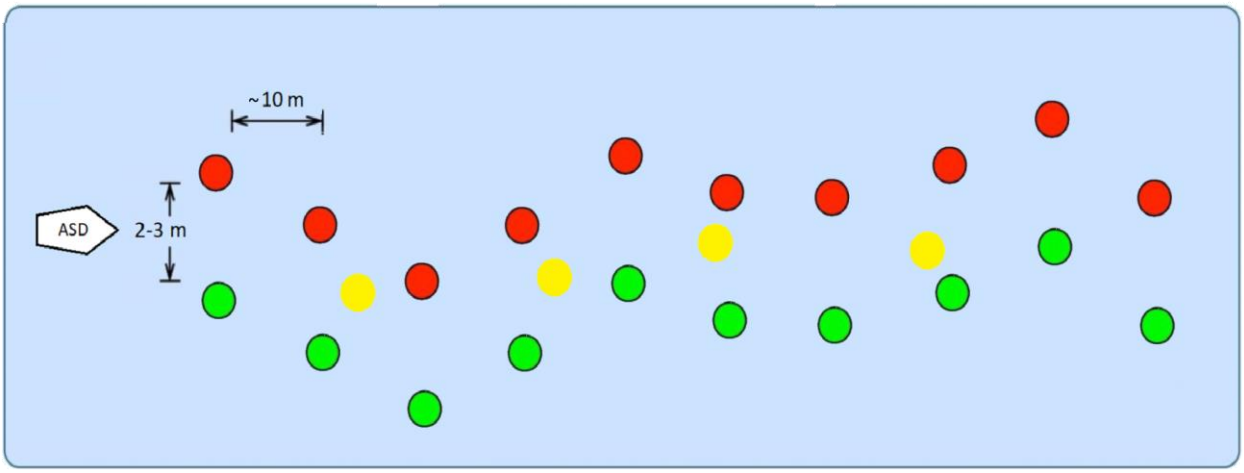


Figure 5 Obstacle Channel overview

Points		
Gate	**Obstacle**	**Time**
+5	-10	100 - 10/min

11.2 Collision avoidance

The purpose of the Collision Avoidance task is to demonstrate complex path planning. The drones must move through an area with crossing traffic from both port and starboard. The ASD should behave according to Convention on the International Regulations for Preventing Collisions at Sea (COLREGs). The direction of the arriving crossing traffic will be randomized.

The ASD must hold a speed below 4 knots and complete the mission in less than 10 minutes.

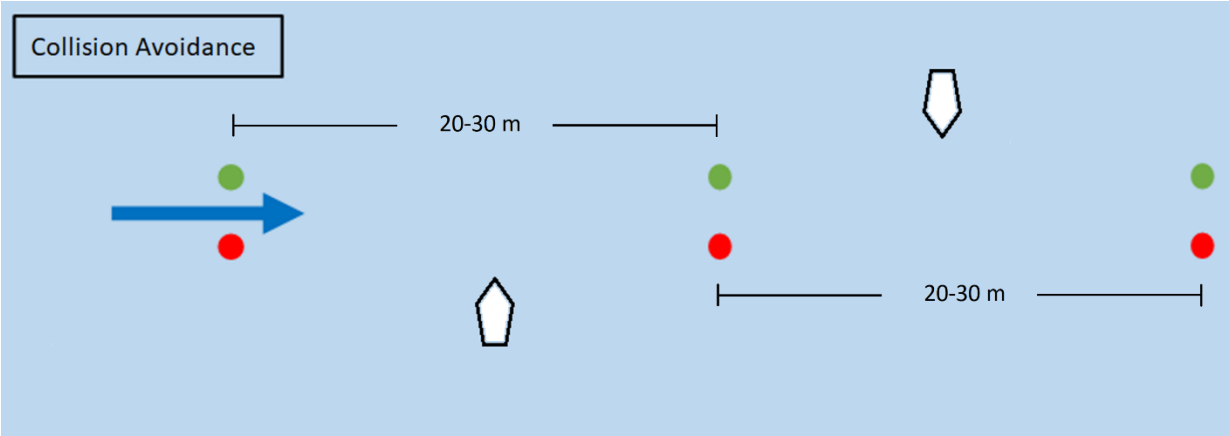


Figure 6 Collision Avoidance (crossing traffic for illustration only)

Points		
Gate	**Collision**	**COLREGs**
+10	-30	+30/rule followed

11.3 Visual Docking

Successful completion of the Visual Docking task will demonstrate the ability to localize the docking area, and maneuver into docking position in a defined area.

The docking area will be an area of a pier marked with a green and red buoy two meters apart. The ASD should navigate to the area between the buoys and dock with its port or starboard side to the pier. It should be docked at least for 30 sec (can use thrusters to keep the ASD at dock). Then after this time the ASD must leave the dock.

Points are given for reaching the docking area, docking correctly and then for every second the ASD is kept docked.

The mission must be completed in less than 10 minutes.

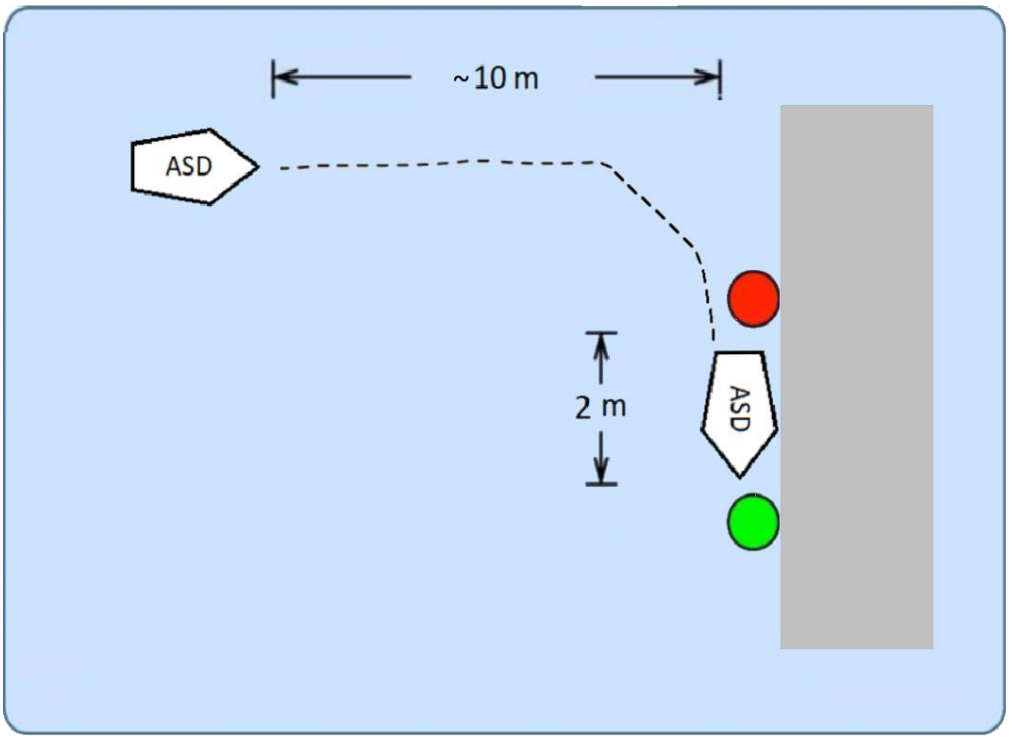


Figure 7 Docking (illustration only)

Points		
Dock Reached	**Docking correct**	**Docking time**
+10	+30	+ 2 /sec docked

11.4 Speed Gate

Successful completion of the Speed Gate task demonstrates the ASD's hull form efficiency coupled with its propulsion system, and the resulting maneuverability. Furthermore, it demonstrates object recognition and decision making with respect to sensing the task elements.

The drone must enter through the gate buoys, go around the Mark buoy (counterclockwise or clockwise), and exit through the same gate buoys, as quickly as possible. The gate buoys are moored 2-3m apart, and the Mark buoy is placed 20-30m from the gate buoys.

This is a timed challenge. Time starts when the bow (front) of the drone crosses the Gate buoys (entry) and stops when the stern (back) of the drone crosses the Gate buoys (exit).

Completing the course in less than 10 minutes gives 20 points, additional points are given to the three fastest ASDs.

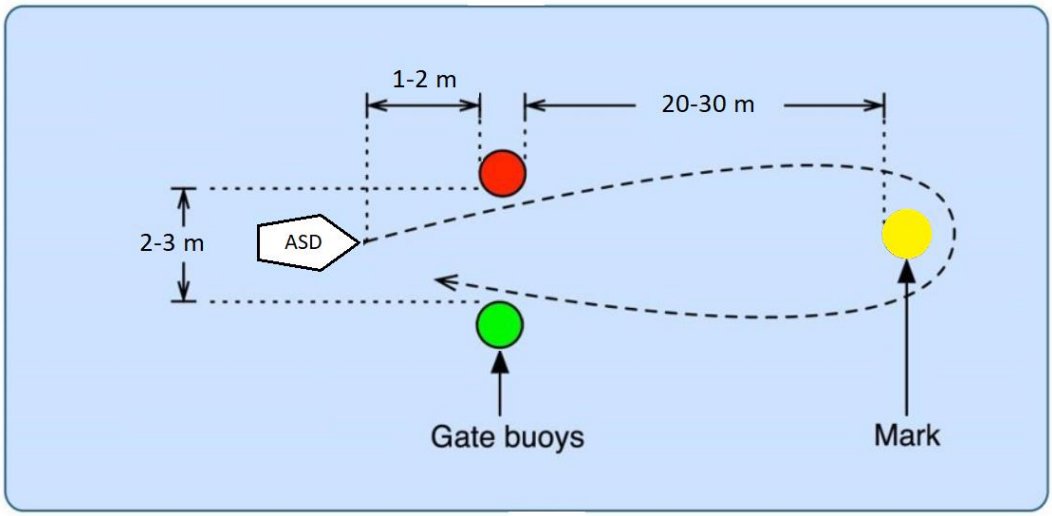


Figure 8 Speed Gate (Illustration only)

Scoring	
Time	**Points**
Fastest	80
2nd	50
3rd	20

12 Communication with Championship Mission Task Server

To earn points for their mission task completions teams must be able to communicate with the championship mission task server. All competitor Operator Control Stations (OCS) will be required to have the ability to connect to a wired Ethernet network (provided) by competition organizers) and request an IP address from a DHCP server. Once connected, they should establish a TCP connection to a server, with a given address and port number and provide GPS data through an NMEA 0183 sentence (to be used with OpenCPN chart plotter).

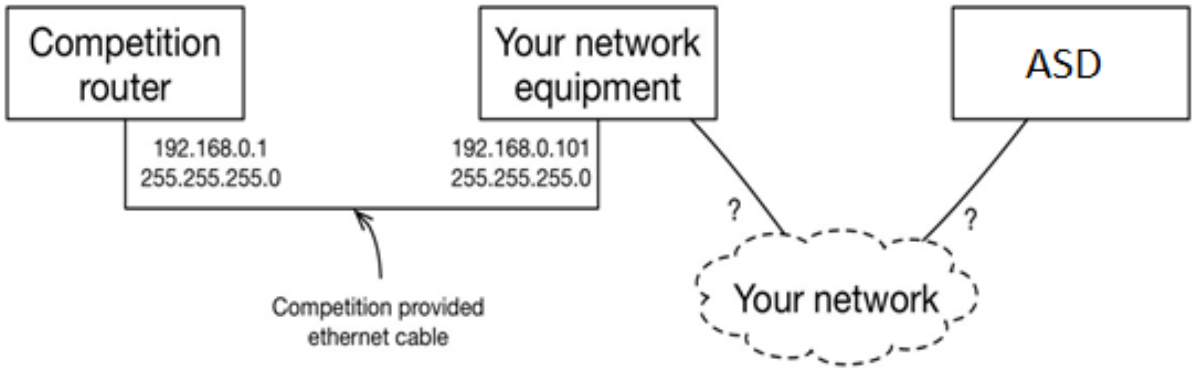


Figure 9 Virtual network connectivity overview

13 Appendices

13.1 Visual Feedback

This section describes a lighting system that will serve as a visual status indicator for the team's drone.

With unmanned systems being integrated into everyday use, it is safety critical for these systems to indicate their operational status. Resources and general guidelines will be outlined in this document to permit teams to acquire, integrate, and test a system that meets the safety requirement set forth for AutoDrone operations.

The lighting system shall consist of, at minimum, three lights: Red, Amber/Yellow, and Green/Blue. Lights shall be arranged in a vertical configuration and mounted such that they provide a **360 degree daylight visibility**, when viewed from shore or nearby vessel.

Lighting system colors shall correspond with the applicable mode of the team's autonomous system, as indicated in table below. The lights may be flashing or steady on/off according to the state of the system.

Table 2: Light color and correlating modes

Color	Mode
Red	E-Stop active (propulsion disabled)
Amber/Yellow	Tele-Operation / Manual Operation
Green/Blue	Autonomous operation

Visual indication system can be purchased commercially or can be a custom array of RGB LEDs. Keeping the below specifications in mind, design and selection of the final system is the team's decision.

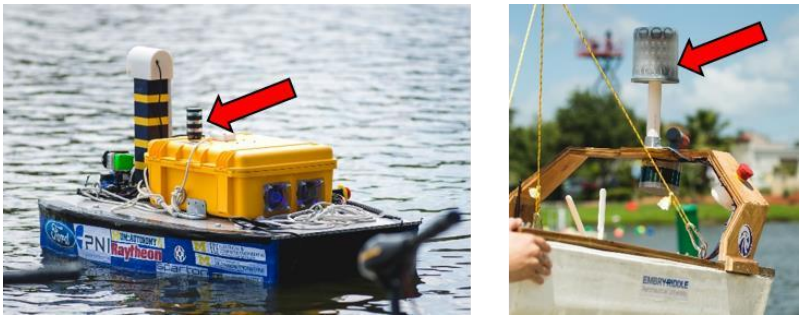


Figure 10 Visual Feedback Indicators (Left – Commercial; Right – Custom)

To provide visibility in sunlight, teams should use lighting systems that have clear enclosures for the light to shine through; rather than colored enclosures with standard light bulbs.

13.2 Onboard Emergency Stop

All drones must have an onboard emergency stop capable of being actuated by personnel from a support craft. For personnel safety, the switch may be triggered from a distance by a wooden or plastic pole/paddle. Keeping this in mind, teams should select rugged and reliable components for their safety system.

A large, red button should be installed in such a way that safety personnel, from the support craft can easily actuate the button. The engage/disengage button should be red in color and have a 'press to activate and turn to reset' feature. This button, momentary contact switch or not, on actuation, should cut power to the thrusters immediately on actuation. The thrusters must remain in a powered-down state until the judge gives permission for the team to reinitialize the system. An example of a suitable button is shown in Figure 11.

Figure 11 Example of a Kill Switch

Appendix G

Buoy data collection

```python
import cv2
import numpy

# Open the ZED camera
capture = cv2.VideoCapture(0)

# Set the video resolution to HD720 (2560*720)
capture.set(cv2.CAP_PROP_FRAME_WIDTH, 2560)
capture.set(cv2.CAP_PROP_FRAME_HEIGHT, 720)

# The variables for the image counters
red_counter = 0
green_counter = 0
yellow_counter = 0

# Updating the counter of the given object
def update_counter(object):
    if object == 'red':
        global red_counter
        red_counter = (red_counter + 1)
    elif object == 'green':
        global green_counter
        green_counter = (green_counter + 1)
    elif object == 'yellow':
        global yellow_counter
        yellow_counter = (yellow_counter + 1)

# Retrieving the counter for the given object
def retrieve_counter(object):
    if object == 'red':
        return red_counter
    elif object == 'green':
        return green_counter
    elif object == 'yellow':
        return yellow_counter

# Saving the image of the given object using the given lens
def save_image(object, lens):
    image_path = './bouyData/{}{}.png'.format(object, retrieve_counter(object))
    cv2.imwrite(image_path, left_right_image[lens])  # 0 will return the image from the left lens, 1 will return right.
    update_counter(object)
    print('{} image counter: {}'.format(object, retrieve_counter(object)))
```

```


while capture.isOpened():
    # Get a new frame from camera
    _, frame = capture.read()
    # Extract left and right images from side-by-side
    left_right_image = numpy.split(frame, 2, axis=1)
    # Display images
    cv2.imshow("right", left_right_image[0])
    cv2.imshow("left", left_right_image[1])

    key = cv2.waitKey(1)

    # IF waited at least 1 ms AND 'q' is pressed then EXIT the loop.
    if key & 0xFF == ord('q'):
        print('Escape hit, closing...')
        break

    # IF waited at least 1 ms AND '1' is pressed then save the Red frame.
    elif key & 0xFF == ord('1'):
        save_image(object='red', lens=0)
        save_image(object='red', lens=1)

    # IF waited at least 1 ms AND '2' is pressed then save the Green frame.
    elif key & 0xFF == ord("2"):
        save_image(object='green', lens=0)
        save_image(object='green', lens=1)

    # IF waited at least 1 ms AND '3' is pressed then save the Yellow frame.
    elif key & 0xFF == ord("3"):
        save_image(object='yellow', lens=0)
        save_image(object='yellow', lens=1)

#   Destroy all of windows
cv2.destroyAllWindows()
#   Release the captured frame
capture.release()

```

Appendix H

RGB-D Mapping launch file

```
<?xml version="1.0"?>
<!-- RGB-D mapping using RTABMap with ROS. WAM-V localization nodes is
based on osrf/vrx.  -->
<launch>
<!-- Arguments which can be called -->
   <!-- Localization mode -->
   <arg name="localization"  default="false"/>
   <!-- Path to database -->
   <arg name="database_path"      default="/home/markus/output.db"/>
   <!-- Choose visualization -->
   <arg name="rviz" default="false" />
   <arg name="rtabmapviz" default="false" />
   <!-- Use simulated time -->
   <param name="use_sim_time" type="bool" value="true"/>


<!-- Publishes fixed joint static transforms (cameras, lidars, etc) to
/tf -->
  <node ns="wamv" pkg="robot_state_publisher"
type="robot_state_publisher" name="rob_st_pub">
    <param name="tf_prefix" value="wamv" />
  </node>

<!-- Publishes revolute joint static transforms (gps and imu) to /tf --
>
    <node ns="wamv" name="joint_state_publisher"
pkg="joint_state_publisher" type="joint_state_publisher"
output="screen">
      <param name="gui" value="false" />
    </node>

  <!-- Extended kalman filter fusing imu and gps into combined
odometry/tf -->
  <node ns="wamv/robot_localization" pkg="robot_localization"
type="ekf_localization_node"
        name="ekf_localization" clear_params="false">
    <param name="sensor_timeout" value="2.0"/>
    <param name="two_d_mode" value="false"/>
    <param name="map_frame" value="map"/>
    <param name="odom_frame" value="wamv/odom"/>
    <param name="base_link_frame" value="wamv/base_link"/>
    <param name="world_frame" value="wamv/odom"/>
    <param name="publish_tf" value="true"/>
    <param name="frequency" value="60"/>
    <param name="imu0" value="/wamv/sensors/imu/imu/data"/>
    <!-- IMU measures orientation, angular velocity, and linear
acceleration -->
    <rosparam param="imu0_config">[false, false, false,
                                   true,  true,  true,
                                   false,  false,  false,
                                   true,  true,  true,
                                   true,  true,  true]</rosparam>
    <param name="imu0_differential" value="false"/>
    <param name="imu0_remove_gravitational_acceleration" value="true"/>

    <param name="odom0" value="/wamv/robot_localization/odometry/gps"/>
    <!-- GPS only reliably measures absolute position -->
    <rosparam param="odom0_config">[true,  true,  true,
```

```
                                    false, false, false,
                                    false, false, false,
                                    false, false, false,
                                    false, false, false]</rosparam>

    <param name="odom0_differential" value="false"/>
    <param name="smooth_lagged_data" value="true"/>
  </node>

  <!-- Produces local odometry from GPS to be used in Kalman filter -->
  <node ns="wamv/robot_localization" pkg="robot_localization"
type="navsat_transform_node"
        name="navsat_transform_node" respawn="true" output="screen">
    <param name="tf_prefix" value="wamv" />
    <param name="frequency" value="60"/>
    <param name="magnetic_declination_radians" value="0"/>
    <param name="broadcast_utm_transform" value="true"/>
    <param name="wait_for_datum" value="true"/>
    <param name="use_odometry_yaw" value="true"/>
    <rosparam param="datum">[21.30996, -157.8901]</rosparam>
    <param name="yaw_offset" value="0"/>
    <param name="publish_filtered_gps" value="true"/>
    <remap to="/wamv/sensors/gps/gps/fix"
from="/wamv/robot_localization/gps/fix" />
  </node>


<!-- Using RTABMap packet, description and list of parameters can be
found:
https://github.com/introlab/rtabmap/blob/master/corelib/include/rtabmap
/core/Parameters.h#L161 -->
<group ns="rtabmap">

  <!-- Creating a nodelet manager in order to avoid topic serialization
-->
  <node pkg="nodelet" type="nodelet" name="standalone_nodelet"
args="manager"/>

<!-- Disparity to depth -->
  <node pkg="nodelet" type="nodelet" name="disparity_to_depth"
args="standalone rtabmap_ros/disparity_to_depth manager"
output="screen">
      <remap from="disparity"
to="/wamv/sensors/cameras/stereo_camera/disparity"/>
  </node>

    <!-- Use rgbd synchronization -->
    <node pkg="nodelet" type="nodelet" name="rgbd_sync"
args="standalone rtabmap_ros/rgbd_sync manager" output="screen">
      <remap from="rgb/image"
to="/wamv/sensors/cameras/stereo_camera/left/image_rect_color"/>
      <remap from="depth/image"     to="depth"/>
      <remap from="rgb/camera_info"
to="/wamv/sensors/cameras/stereo_camera/left/camera_info"/>
      <param name="queue_size"      type="int"    value="100"/>
      <param name="approx_sync"     type="bool"   value="true"/>
    </node>
```

```
  <!-- Visual SLAM -->
  <node name="rtabmap" pkg="rtabmap_ros" type="rtabmap" output="screen"
args="--uinfo">
    <param name="database_path"             type="string"  value="$(arg
database_path)"/>
    <param name="subscribe_rgb"             type="bool"
       value="false"/>
    <param name="subscribe_rgbd"    type="bool"   value="true"/>
    <param name="subscribe_scan_cloud"      type="bool"
       value="true"/>
    <param name="approx_sync"       type="bool"   value="true"/>
    <param name="frame_id"          type="string"
       value="wamv/base_link"/>
    <param name="subscribe_odom"    type="bool"   value="true"/>
    <param name="visual_odometry"   type="bool"   value="false"/>

    <!-- must be false for appearance-based mode -->
    <param name="wait_for_transform" type="bool"   value="true"/>
    <param name="map_negative_poses_ignored" type="bool"
value="false"/>
    <!-- refresh grid map even if we are not moving-->
    <param name="map_negative_scan_empty_ray_tracing" type="bool"
value="false"/>
    <param name="map_always_update" type="bool"   value="true"/>

    <remap from="odom"
       to="/wamv/robot_localization/odometry/filtered"/>
    <remap from="scan_cloud"
to="/wamv/sensors/lidars/lidar_wamv/points"/>
    <remap from="rgbd_image"        to="rgbd_image"/>
     <!-- g2o=1, GTSAM=2 -->
    <param name="RGBD/OptimizeStrategy" type="string" value="2"/>
    <param name="Optimizer/Strategy"  type="string" value="2"/> <!--
GTSAM global optimization -->
    <!-- Do odometry correction with consecutive laser scans -->
    <param name="RGBD/NeighborLinkRefining" type="string"
value="true"/>
    <param name="RGBD/OptimizeRobust" type="string" value="true"/>
    <!-- should be 0 if RGBD/OptimizeRobust is true -->
    <param name="RGBD/OptimizeMaxError" type="string" value="0"/>

    <!-- Projected occupancy grid characteristic -->
    <param name="Grid/MaxGroundHeight"             type="string"
value="1.5"/>
    <param name="Grid/MaxObstacleHeight"    type="string" value="4"/>
    <param name="Grid/NoiseFilteringRadius" type="string" value="1"/>
    <param name="Grid/MinClusterSize"              type="string"
value="100"/>
    <param name="Grid/FlatObstacleDetected" type="bool"
value="false"/>
    <param name="Grid/RayTracing"           type="bool"   value="true"/>
    <param name="Grid/FootprintLength"             type="string"
value="5"/>
    <param name="Grid/FootprintWidth"              type="string"
value="2"/>
    <param name="Grid/FootprintHeight"             type="string"
value="1.8"/>
```

```
    <param name="Grid/DepthRoiRatios"              type="string"
value="0.0 0.0 0.3 0.4"/>
    <param name="Grid/RangeMin"                    type="string"
value="2"/>
    <param name="Grid/RangeMax"                    type="string"
value="0"/>
    <param name="Grid/FromDepth"           type="bool"
value="false"/>
    <param name="GridGlobal/FootprintRadius"       type="string"
value="3"/>
    <param name="Vis/MinInliers"     type="string" value="10"/>
    <param name="Vis/EstimationType" type="string" value="1"/> <!-- use
3D to 3D estimation -->
    <param name="Vis/MaxDepth"       type="string" value="10"/>
    <param name="Vis/MinDepth"       type="string" value="2"/>
    <param name="Vis/RoiRatios"            type="string" value="0.0 0.0
0.3 0.4"/>
    <param name="Kp/RoiRatios"             type="string" value="0.0 0.0
0.3 0.4"/>
    <param name="Kp/MinDepth"       type="string" value="2"/>
    <param name="queue_size" type="int" value="5000"/>

         <!-- localization mode -->
    <param     if="$(arg localization)" name="Mem/IncrementalMemory"
type="string" value="false"/>
    <param unless="$(arg localization)" name="Mem/IncrementalMemory"
type="string" value="true"/>
    <param name="Mem/InitWMWithAllNodes" type="string" value="$(arg
localization)"/>

    <!-- output -->
    <remap from="grid_map" to="/map"/>
  </node>
</group>

<!-- visualization with rtabmapviz -->
  <node if="$(arg rtabmapviz)" pkg="rtabmap_ros" type="rtabmapviz"
name="rtabmapviz" args="-d $(find
rtabmap_ros)/launch/config/rgbd_gui.ini" output="screen">
    <param name="subscribe_odom_info" type="bool" value="true"/>
    <param name="queue_size" type="int" value="1000"/>
    <param name="subscribe_depth"  type="bool"   value="true"/>
    <param name="subscribe_scan_cloud" type="bool" value="true"/>
    <param name="subscribe_rgbd"   type="bool"   value="true"/>
    <param name="frame_id"            type="string"
value="wamv/base_link"/>
    <param name="wait_for_transform"  type="bool"   value="true"/>
    <remap from="rgbd_image"        to="rgbd_image"/>
    <remap from="scan_cloud"
to="/wamv/sensors/lidars/lidar_wamv/points"/>
    <remap from="odom"
to="/wamv/robot_localization/odometry/filtered"/>
    <remap from="mapData" to="mapData"/>
  </node>

<!-- Visualisation in RVIZ -->
  <arg name="rviz_config" default="$(find
wamv_gazebo)/config/rviz_vrx.rviz" />
```

```
  <node if="$(arg rviz)" pkg="rviz" type="rviz"
        name="wamv_visualization" args="-d $(arg rviz_config)" />

</launch>
```

Appendix I

RGB-D Localization launch file

```
<?xml version="1.0"?>
<!-- RGB-D localization using rtabmap with ROS. Wam-v localization
nodes is based on osrf/vrx simulation.  -->
<launch>
<!-- Arguments which can be called -->
   <arg name="localization"  default="true"/>
<!-- Path to database -->
   <arg name="database_path"      default="/home/markus/output.db"/>
<!-- Choose visualization -->
   <arg name="rviz" default="false" />
   <arg name="rtabmapviz" default="false" />
<!-- Use simulated time -->
   <param name="use_sim_time" type="bool" value="true"/>


<!-- Publishes fixed joint static transforms (cameras, lidars, etc) to
/tf -->
  <node ns="wamv" pkg="robot_state_publisher"
type="robot_state_publisher" name="rob_st_pub">
    <param name="tf_prefix" value="wamv" />
  </node>

<!-- Publishes revolute joint static transforms (gps and imu) to /tf --
>
    <node ns="wamv" name="joint_state_publisher"
pkg="joint_state_publisher" type="joint_state_publisher"
output="screen">
      <param name="gui" value="false" />
    </node>

  <!-- Extended kalman filter fusing imu and gps into combined
odometry/tf -->
  <node ns="wamv/robot_localization" pkg="robot_localization"
type="ekf_localization_node"
        name="ekf_localization" clear_params="false">
    <param name="sensor_timeout" value="2.0"/>
    <param name="two_d_mode" value="false"/>
    <param name="map_frame" value="map"/>
    <param name="odom_frame" value="wamv/odom"/>
    <param name="base_link_frame" value="wamv/base_link"/>
    <param name="world_frame" value="wamv/odom"/>
    <param name="publish_tf" value="true"/>
    <param name="frequency" value="60"/>
    <param name="imu0" value="/wamv/sensors/imu/imu/data"/>
    <!-- IMU measures orientation, angular velocity, and linear
acceleration -->
    <rosparam param="imu0_config">[false, false, false,
                                   true,  true,  true,
                                   false,  false,  false,
                                   true,  true,  true,
                                   true,  true,  true]</rosparam>
    <param name="imu0_differential" value="false"/>
    <param name="imu0_remove_gravitational_acceleration" value="true"/>

    <param name="odom0" value="/wamv/robot_localization/odometry/gps"/>
    <!-- GPS only reliably measures absolute position -->
    <rosparam param="odom0_config">[true,  true,  true,
                                   false, false, false,
```

```
                                    false, false, false,
                                    false, false, false,
                                    false, false, false]</rosparam>

    <param name="odom0_differential" value="false"/>
    <param name="smooth_lagged_data" value="true"/>
  </node>

  <!-- Produces local odometry from GPS to be used in Kalman filter -->
  <node ns="wamv/robot_localization" pkg="robot_localization"
type="navsat_transform_node"
        name="navsat_transform_node" respawn="true" output="screen">
    <param name="tf_prefix" value="wamv" />
    <param name="frequency" value="60"/>
    <param name="magnetic_declination_radians" value="0"/>
    <param name="broadcast_utm_transform" value="true"/>
    <param name="wait_for_datum" value="true"/>
    <param name="use_odometry_yaw" value="true"/>
    <rosparam param="datum">[21.30996, -157.8901]</rosparam>
    <param name="yaw_offset" value="0"/>
    <param name="publish_filtered_gps" value="true"/>
    <remap to="/wamv/sensors/gps/gps/fix"
from="/wamv/robot_localization/gps/fix" />
  </node>

<!-- ROS navigation stack move_base -->
<group ns="planner">
  <remap from="obstacles_cloud" to="/rtabmap/cloud_obstacles"/>
  <remap from="ground_cloud" to="/rtabmap/cloud_ground"/>
  <remap from="map" to="map"/>
  <remap from="move_base_simple/goal" to="/planner_goal"/>

    <node pkg="move_base" type="move_base" respawn="false"
name="move_base" output="screen">
     <param name="base_global_planner" value="navfn/NavfnROS"/>
     <rosparam file="/home/markus/costmap_common_params.yaml"
command="load" ns="global_costmap" />
     <rosparam file="/home/markus/costmap_common_params.yaml"
command="load" ns="local_costmap" />
     <rosparam file="/home/markus/local_costmap_params.yaml"
command="load"/>
     <rosparam file="/home/markus/global_costmap_params.yaml"
command="load"/>
     <rosparam file="/home/markus/base_local_planner_params.yaml"
command="load"/>
   </node>

   <param name="cmd_vel/abtr_priority" value="10"/>
</group>


  <node pkg="nodelet" type="nodelet" name="obstacles_detection"
args="load rtabmap_ros/obstacles_detection stereo_nodelet">
    <remap from="cloud" to="/wamv/sensors/lidars/lidar_wamv/points"/>
    <remap from="obstacles" to="/planner_cloud"/>

    <param name="frame_id" type="string" value="wamv/base_link"/>
    <param name="map_frame_id" type="string" value="map"/>
```

```
    <param name="wait_for_transform" type="bool" value="true"/>
    <param name="min_cluster_size" type="int" value="50"/>
    <param name="max_obstacles_height" type="double" value="5.0"/>
  </node>


<group ns="rtabmap">

  <!-- Creating a nodelet manager in order to avoid topic serialization
-->
  <node pkg="nodelet" type="nodelet" name="standalone_nodelet"
args="manager"/>

<!-- Disparity to depth -->
  <node pkg="nodelet" type="nodelet" name="disparity_to_depth"
args="standalone rtabmap_ros/disparity_to_depth manager"
output="screen">
      <remap from="disparity"
to="/wamv/sensors/cameras/stereo_camera/disparity"/>
  </node>

    <!-- Use rgbd synchronization -->
    <node pkg="nodelet" type="nodelet" name="rgbd_sync"
args="standalone rtabmap_ros/rgbd_sync manager" output="screen">
      <remap from="rgb/image"
to="/wamv/sensors/cameras/stereo_camera/left/image_rect_color"/>
      <remap from="depth/image"     to="depth"/>
      <remap from="rgb/camera_info"
to="/wamv/sensors/cameras/stereo_camera/left/camera_info"/>
      <param name="queue_size"      type="int"    value="100"/>
      <param name="approx_sync"     type="bool"   value="true"/>
    </node>

  <!-- Visual SLAM -->
  <node name="rtabmap" pkg="rtabmap_ros" type="rtabmap" output="screen"
args="--uinfo">
    <param name="database_path"            type="string"  value="$(arg
database_path)"/>
    <param name="subscribe_rgb"            type="bool"
       value="false"/>
    <param name="subscribe_rgbd"    type="bool"   value="true"/>
    <param name="subscribe_scan"    type="bool"   value="false"/>
    <param name="subscribe_scan_cloud"     type="bool"
       value="true"/>
    <param name="subscribe_depth"   type="bool"   value="false"/>
    <param name="approx_sync"       type="bool"   value="true"/>
    <param name="frame_id"          type="string"
       value="wamv/base_link"/>
    <param name="subscribe_odom"    type="bool"   value="true"/>
    <param name="visual_odometry"   type="bool"   value="false"/>

<!-- must be false for appearance-based mode -->
    <param name="wait_for_transform" type="bool"   value="true"/>
    <param name="map_negative_poses_ignored" type="bool"
value="false"/>
<!-- refresh grid map even if we are not moving-->
    <param name="map_negative_scan_empty_ray_tracing" type="bool"
value="false"/>
```

```
    <param name="map_always_update" type="bool"   value="true"/>


    <remap from="goal_out"  to="current_goal"/>
    <remap from="move_base" to="/planner/move_base"/>
    <remap from="odom"
       to="/wamv/robot_localization/odometry/filtered"/>
    <remap from="scan_cloud"
to="/wamv/sensors/lidars/lidar_wamv/points"/>
    <remap from="rgbd_image"        to="rgbd_image"/>
     <!-- g2o=1, GTSAM=2 -->
    <param name="RGBD/OptimizeStrategy" type="string" value="2"/>
    <param name="Optimizer/Strategy"  type="string" value="2"/> <!--
GTSAM global optimization -->
    <!-- Do odometry correction with consecutive laser scans -->
    <param name="RGBD/NeighborLinkRefining" type="string"
value="true"/>
    <param name="RGBD/OptimizeRobust" type="string" value="true"/>
    <!-- should be 0 if RGBD/OptimizeRobust is true -->
    <param name="RGBD/OptimizeMaxError" type="string" value="0"/>

    <!-- Projected occupancy grid characteristic -->
    <param name="Grid/MaxGroundHeight"            type="string"
value="1.5"/>
    <param name="Grid/MaxObstacleHeight"   type="string" value="4"/>
    <param name="Grid/NoiseFilteringRadius" type="string" value="1"/>
    <param name="Grid/MinClusterSize"             type="string"
value="100"/>
    <param name="Grid/FlatObstacleDetected" type="bool"
value="false"/>
    <param name="Grid/RayTracing"          type="bool"   value="true"/>
    <param name="Grid/FootprintLength"            type="string"
value="5"/>
    <param name="Grid/FootprintWidth"             type="string"
value="2"/>
    <param name="Grid/FootprintHeight"            type="string"
value="1.8"/>
    <param name="Grid/DepthRoiRatios"             type="string"
value="0.0 0.0 0.3 0.4"/>
    <param name="Grid/RangeMin"                   type="string"
value="2"/>
    <param name="Grid/RangeMax"                   type="string"
value="0"/>
    <param name="Grid/FromDepth"           type="bool"   value="true"/>
    <param name="GridGlobal/FootprintRadius"      type="string"
value="3"/>
    <param name="Vis/MinInliers"     type="string" value="10"/>
    <param name="Vis/EstimationType" type="string" value="1"/> <!-- use
3D to 3D estimation -->
    <param name="Vis/MaxDepth"       type="string" value="10"/>
    <param name="Vis/MinDepth"       type="string" value="2"/>
    <param name="Vis/RoiRatios"            type="string" value="0.0 0.0
0.3 0.4"/>
    <param name="Kp/RoiRatios"             type="string" value="0.0 0.0
0.3 0.4"/>
    <param name="Kp/MinDepth"       type="string" value="2"/>
<!-- max depth can be higher with 2D to 3D estimation -->
    <param name="queue_size" type="int" value="5000"/>
```

```
        <!-- localization mode -->
    <param     if="$(arg localization)" name="Mem/IncrementalMemory"
type="string" value="false"/>
    <param unless="$(arg localization)" name="Mem/IncrementalMemory"
type="string" value="true"/>
    <param name="Mem/InitWMWithAllNodes" type="string" value="$(arg
localization)"/>

    <!-- output -->
    <remap from="grid_map" to="/map"/>
  </node>
</group>

<!-- visualization with rtabmapviz -->
  <node if="$(arg rtabmapviz)" pkg="rtabmap_ros" type="rtabmapviz"
name="rtabmapviz" args="-d $(find
rtabmap_ros)/launch/config/rgbd_gui.ini" output="screen">
    <param name="subscribe_odom_info" type="bool" value="true"/>
    <param name="queue_size" type="int" value="1000"/>
    <param name="subscribe_depth"  type="bool"   value="true"/>
    <param name="subscribe_scan_cloud" type="bool" value="true"/>
    <param name="subscribe_rgbd"   type="bool"   value="true"/>
    <param name="frame_id"            type="string"
value="wamv/base_link"/>
    <param name="wait_for_transform"  type="bool"   value="true"/>
    <remap from="rgbd_image"        to="rgbd_image"/>
    <remap from="scan_cloud"
to="/wamv/sensors/lidars/lidar_wamv/points"/>
    <remap from="odom"
to="/wamv/robot_localization/odometry/filtered"/>
    <remap from="mapData" to="mapData"/>
  </node>

<!-- Visualisation in RVIZ -->
  <arg name="rviz_config" default="$(find
wamv_gazebo)/config/rviz_vrx.rviz" />
  <node if="$(arg rviz)" pkg="rviz" type="rviz"
        name="wamv_visualization" args="-d $(arg rviz_config)" />


    <!-- Set articulation angles -->
    <node pkg="vrx_gazebo" type="key2thrust_angle.py"
name="key2thrust_angle" output="screen">
      <param name="max_angle" value="$(eval pi/2)"/>
      <remap from="left_thrust_angle"
to="/wamv/thrusters/left_thrust_angle"/>
      <remap from="right_thrust_angle"
to="/wamv/thrusters/right_thrust_angle"/>
    </node>
    <!-- Convert Twist messages to Drive messages -->
    <node pkg="vrx_gazebo" type="twist2thrust.py" name="twist2thrust"
output="screen">
      <remap from="left_cmd" to="/wamv/thrusters/left_thrust_cmd"/>
      <remap from="right_cmd" to="/wamv/thrusters/right_thrust_cmd"/>
    </node>

</launch>
```

Appendix J

YOLO Training file

```python
import cv2
from darkflow.net.build import TFNet
import numpy as np
import time
import tensorflow as tf
gpu_options = tf.GPUOptions(allow_growth=True)
#load: "bin/yolo.weights"
options = {
    "model": "cfg/yolo3c.cfg",
    "load": 7000,
    "gpu": 1.0,
    "batch": 2,
    "epoch": 100,
    "train": True,
    "annotation": "C:/Users/sigur/PycharmProjects/Yoloo/darkflow/Bannoation",
    "dataset": "C:/Users/sigur/PycharmProjects/Yoloo/darkflow/Bimages"}

tfnet = TFNet(options)

tfnet.train()
# saves the built graph to a protobuf file (.pb)
tfnet.savepb()
```

Appendix K

YOLO mAP validation file

The following code comes from [16], but is configured and edited for the thesis.

```python
import glob
import json
import os
import shutil
import operator
import sys
import argparse
import math
import cv2
import numpy as np

MINOVERLAP = 0.60 # default value is 0.5 (defined in the PASCAL VOC2012 challenge)

parser = argparse.ArgumentParser()
parser.add_argument('-na', '--no-animation', help="no animation is shown.", action="store_true")
parser.add_argument('-np', '--no-plot', help="no plot is shown.", action="store_true")
parser.add_argument('-q', '--quiet', help="minimalistic console output.", action="store_true")
# argparse receiving list of classes to be ignored (e.g., python main.py --ignore person book)
parser.add_argument('-i', '--ignore', nargs='+', type=str, help="ignore a list of classes.")
# argparse receiving list of classes with specific IoU (e.g., python main.py --set-class-iou person 0.7)
parser.add_argument('--set-class-iou', nargs='+', type=str, help="set IoU for a specific class.")
args = parser.parse_args()

'''
    0,0 ------> x (width)
     |
     |  (Left,Top)
     |      *_________
     |      |         |
            |         |
     y      |_________|
  (height)            *
                (Right,Bottom)
'''

# if there are no classes to ignore then replace None by empty list
if args.ignore is None:
```

```python
    args.ignore = []

specific_iou_flagged = False
if args.set_class_iou is not None:
    specific_iou_flagged = True

# make sure that the cwd() is the location of the python script (so that every path makes sense)
os.chdir(os.path.dirname(os.path.abspath(__file__)))

GT_PATH = os.path.join(os.getcwd(), 'input', 'ground-truth')
DR_PATH = os.path.join(os.getcwd(), 'input', 'detection-results')
# if there are no images then no animation can be shown
IMG_PATH = os.path.join(os.getcwd(), 'input', 'images-optional')
if os.path.exists(IMG_PATH):
    for dirpath, dirnames, files in os.walk(IMG_PATH):
        if not files:
            # no image files found
            args.no_animation = True
else:
    args.no_animation = True

# try to import OpenCV if the user didn't choose the option --no-animation
show_animation = False
if not args.no_animation:
    try:
        import cv2
        show_animation = True
    except ImportError:
        print("\"opencv-python\" not found, please install to visualize the results.")
        args.no_animation = True

# try to import Matplotlib if the user didn't choose the option --no-plot
draw_plot = False
if not args.no_plot:
    try:
        import matplotlib.pyplot as plt
        draw_plot = True
    except ImportError:
        print("\"matplotlib\" not found, please install it
```

```
to get the resulting plots.")
        args.no_plot = True


def log_average_miss_rate(precision, fp_cumsum, num_images):
    """
        log-average miss rate:
            Calculated by averaging miss rates at 9 evenly spaced FPPI points
            between 10e-2 and 10e0, in log-space.

        output:
                lamr | log-average miss rate
                mr | miss rate
                fppi | false positives per image

        references:
            [1] Dollar, Piotr, et al. "Pedestrian Detection: An Evaluation of the
                State of the Art." Pattern Analysis and Machine Intelligence, IEEE
                Transactions on 34.4 (2012): 743 - 761.
    """

    # if there were no detections of that class
    if precision.size == 0:
        lamr = 0
        mr = 1
        fppi = 0
        return lamr, mr, fppi

    fppi = fp_cumsum / float(num_images)
    mr = (1 - precision)

    fppi_tmp = np.insert(fppi, 0, -1.0)
    mr_tmp = np.insert(mr, 0, 1.0)

    # Use 9 evenly spaced reference points in log-space
    ref = np.logspace(-2.0, 0.0, num = 9)
    for i, ref_i in enumerate(ref):
        # np.where() will always find at least 1 index, since min(ref) = 0.01 and min(fppi_tmp) = -1.0
        j = np.where(fppi_tmp <= ref_i)[-1][-1]
        ref[i] = mr_tmp[j]

```

```
    # log(0) is undefined, so we use the np.maximum(1e-10, ref)
    lamr = math.exp(np.mean(np.log(np.maximum(1e-10, ref))))

    return lamr, mr, fppi

"""
 throw error and exit
"""
def error(msg):
    print(msg)
    sys.exit(0)

"""
 check if the number is a float between 0.0 and 1.0
"""
def is_float_between_0_and_1(value):
    try:
        val = float(value)
        if val > 0.0 and val < 1.0:
            return True
        else:
            return False
    except ValueError:
        return False

"""
 Calculate the AP given the recall and precision array
    1st) We compute a version of the measured precision/recall curve with
         precision monotonically decreasing
    2nd) We compute the AP as the area under this curve by numerical integration.
"""
def voc_ap(rec, prec):
    """
    --- Official matlab code VOC2012---
    mrec=[0 ; rec ; 1];
    mpre=[0 ; prec ; 0];
    for i=numel(mpre)-1:-1:1
            mpre(i)=max(mpre(i),mpre(i+1));
    end
    i=find(mrec(2:end)~=mrec(1:end-1))+1;
    ap=sum((mrec(i)-mrec(i-1)).*mpre(i));
    """
```

```
    rec.insert(0, 0.0) # insert 0.0 at begining of list
    rec.append(1.0) # insert 1.0 at end of list
    mrec = rec[:]
    prec.insert(0, 0.0) # insert 0.0 at begining of list
    prec.append(0.0) # insert 0.0 at end of list
    mpre = prec[:]
    """
     This part makes the precision monotonically decreasing
        (goes from the end to the beginning)
        matlab: for i=numel(mpre)-1:-1:1
                    mpre(i)=max(mpre(i),mpre(i+1));
    """
    # matlab indexes start in 1 but python in 0, so I have to do:
    #     range(start=(len(mpre) - 2), end=0, step=-1)
    # also the python function range excludes the end, resulting in:
    #     range(start=(len(mpre) - 2), end=-1, step=-1)
    for i in range(len(mpre)-2, -1, -1):
        mpre[i] = max(mpre[i], mpre[i+1])
    """
     This part creates a list of indexes where the recall changes
        matlab: i=find(mrec(2:end)~=mrec(1:end-1))+1;
    """
    i_list = []
    for i in range(1, len(mrec)):
        if mrec[i] != mrec[i-1]:
            i_list.append(i) # if it was matlab would be i + 1
    """
     The Average Precision (AP) is the area under the curve
        (numerical integration)
        matlab: ap=sum((mrec(i)-mrec(i-1)).*mpre(i));
    """
    ap = 0.0
    for i in i_list:
        ap += ((mrec[i]-mrec[i-1])*mpre[i])
    return ap, mrec, mpre


"""
 Convert the lines of a file to a list
"""
```

```python
def file_lines_to_list(path):
    # open txt file lines to a list
    with open(path) as f:
        content = f.readlines()
    # remove whitespace characters like `\n` at the end of each line
    content = [x.strip() for x in content]
    return content

"""
 Draws text in image
"""
def draw_text_in_image(img, text, pos, color, line_width):
    font = cv2.FONT_HERSHEY_PLAIN
    fontScale = 1
    lineType = 1
    bottomLeftCornerOfText = pos
    cv2.putText(img, text,
            bottomLeftCornerOfText,
            font,
            fontScale,
            color,
            lineType)
    text_width, _ = cv2.getTextSize(text, font, fontScale, lineType)[0]
    return img, (line_width + text_width)

"""
 Plot - adjust axes
"""
def adjust_axes(r, t, fig, axes):
    # get text width for re-scaling
    bb = t.get_window_extent(renderer=r)
    text_width_inches = bb.width / fig.dpi
    # get axis width in inches
    current_fig_width = fig.get_figwidth()
    new_fig_width = current_fig_width + text_width_inches
    propotion = new_fig_width / current_fig_width
    # get axis limit
    x_lim = axes.get_xlim()
    axes.set_xlim([x_lim[0], x_lim[1]*propotion])

"""
 Draw plot using Matplotlib
"""
def draw_plot_func(dictionary, n_classes, window_title,
```

```
plot_title, x_label, output_path, to_show, plot_color, true_p_bar):
    # sort the dictionary by decreasing value, into a list of tuples
    sorted_dic_by_value = sorted(dictionary.items(), key=operator.itemgetter(1))
    # unpacking the list of tuples into two lists
    sorted_keys, sorted_values = zip(*sorted_dic_by_value)
    #
    if true_p_bar != "":
        """
         Special case to draw in:
            - green -> TP: True Positives (object detected and matches ground-truth)
            - red -> FP: False Positives (object detected but does not match ground-truth)
            - pink -> FN: False Negatives (object not detected but present in the ground-truth)
        """
        fp_sorted = []
        tp_sorted = []
        for key in sorted_keys:
            fp_sorted.append(dictionary[key] - true_p_bar[key])
            tp_sorted.append(true_p_bar[key])
        plt.barh(range(n_classes), fp_sorted, align='center', color='crimson', label='False Positive')
        plt.barh(range(n_classes), tp_sorted, align='center', color='forestgreen', label='True Positive', left=fp_sorted)
        # add legend
        plt.legend(loc='lower right')
        """
         Write number on side of bar
        """
        fig = plt.gcf() # gcf - get current figure
        axes = plt.gca()
        r = fig.canvas.get_renderer()
        for i, val in enumerate(sorted_values):
            fp_val = fp_sorted[i]
            tp_val = tp_sorted[i]
            fp_str_val = " " + str(fp_val)
            tp_str_val = fp_str_val + " " + str(tp_val)
            # trick to paint multicolor with offset:
            # first paint everything and then repaint the first number
```

```
            t = plt.text(val, i, tp_str_val, color='forestgreen', va='center', fontweight='bold')
            plt.text(val, i, fp_str_val, color='crimson', va='center', fontweight='bold')
            if i == (len(sorted_values)-1): # largest bar
                adjust_axes(r, t, fig, axes)
    else:
        plt.barh(range(n_classes), sorted_values, color=plot_color)
        """
         Write number on side of bar
        """
        fig = plt.gcf() # gcf - get current figure
        axes = plt.gca()
        r = fig.canvas.get_renderer()
        for i, val in enumerate(sorted_values):
            str_val = " " + str(val) # add a space before
            if val < 1.0:
                str_val = " {0:.2f}".format(val)
            t = plt.text(val, i, str_val, color=plot_color, va='center', fontweight='bold')
            # re-set axes to show number inside the figure
            if i == (len(sorted_values)-1): # largest bar
                adjust_axes(r, t, fig, axes)
    # set window title
    fig.canvas.set_window_title(window_title)
    # write classes in y axis
    tick_font_size = 12
    plt.yticks(range(n_classes), sorted_keys, fontsize=tick_font_size)
    """
     Re-scale height accordingly
    """
    init_height = fig.get_figheight()
    # comput the matrix height in points and inches
    dpi = fig.dpi
    height_pt = n_classes * (tick_font_size * 1.4) # 1.4 (some spacing)
    height_in = height_pt / dpi
    # compute the required figure height
    top_margin = 0.15 # in percentage of the figure height
    bottom_margin = 0.05 # in percentage of the figure height
    figure_height = height_in / (1 - top_margin - bottom_margin)
    # set new height
```

```
    if figure_height > init_height:
        fig.set_figheight(figure_height)

    # set plot title
    plt.title(plot_title, fontsize=14)
    # set axis titles
    # plt.xlabel('classes')
    plt.xlabel(x_label, fontsize='large')
    # adjust size of window
    fig.tight_layout()
    # save the plot
    fig.savefig(output_path)
    # show image
    if to_show:
        plt.show()
    # close the plot
    plt.close()

"""
 Create a ".temp_files/" and "output/" directory
"""
TEMP_FILES_PATH = ".temp_files"
if not os.path.exists(TEMP_FILES_PATH): # if it doesn't exist already
    os.makedirs(TEMP_FILES_PATH)
output_files_path = "output"
if os.path.exists(output_files_path): # if it exist already
    # reset the output directory
    shutil.rmtree(output_files_path)

os.makedirs(output_files_path)
if draw_plot:
    os.makedirs(os.path.join(output_files_path, "classes"))
if show_animation:
    os.makedirs(os.path.join(output_files_path, "images", "detections_one_by_one"))

"""
 ground-truth
     Load each of the ground-truth files into a temporary ".json" file.
     Create a list of all the class names present in the ground-truth (gt_classes).
"""
```

```
# get a list with the ground-truth files
ground_truth_files_list = glob.glob(GT_PATH + '/*.txt')
if len(ground_truth_files_list) == 0:
    error("Error: No ground-truth files found!")
ground_truth_files_list.sort()
# dictionary with counter per class
gt_counter_per_class = {}
counter_images_per_class = {}

gt_files = []
for txt_file in ground_truth_files_list:
    #print(txt_file)
    file_id = txt_file.split(".txt", 1)[0]
    file_id = os.path.basename(os.path.normpath(file_id))
    # check if there is a correspondent detection-results file
    temp_path = os.path.join(DR_PATH, (file_id + ".txt"))
    if not os.path.exists(temp_path):
        error_msg = "Error. File not found: {}\n".format(temp_path)
        error_msg += "(You can avoid this error message by running extra/intersect-gt-and-dr.py)"
        error(error_msg)
    lines_list = file_lines_to_list(txt_file)
    # create ground-truth dictionary
    bounding_boxes = []
    is_difficult = False
    already_seen_classes = []
    for line in lines_list:
        try:
            if "difficult" in line:
                    class_name, left, top, right, bottom, _difficult = line.split()
                    is_difficult = True
            else:
                    class_name, left, top, right, bottom = line.split()
        except ValueError:
            error_msg = "Error: File " + txt_file + " in the wrong format.\n"
            error_msg += " Expected: <class_name> <left> <top> <right> <bottom> ['difficult']\n"
            error_msg += " Received: " + line
            error_msg += "\n\nIf you have a <class_name> with spaces between words you should remove them\n"
            error_msg += "by running the script \"
```

```
remove_space.py\" or \"rename_class.py\" in the \"extra/\" folder."
            error(error_msg)
        # check if class is in the ignore list, if yes skip
        if class_name in args.ignore:
            continue
        bbox = left + " " + top + " " + right + " " + bottom
        if is_difficult:
            bounding_boxes.append({"class_name":class_name, "bbox":bbox, "used":False, "difficult":True})
            is_difficult = False
        else:
            bounding_boxes.append({"class_name":class_name, "bbox":bbox, "used":False})
            # count that object
            if class_name in gt_counter_per_class:
                gt_counter_per_class[class_name] += 1
            else:
                # if class didn't exist yet
                gt_counter_per_class[class_name] = 1

            if class_name not in already_seen_classes:
                if class_name in counter_images_per_class:
                    counter_images_per_class[class_name] += 1
                else:
                    # if class didn't exist yet
                    counter_images_per_class[class_name] = 1
                already_seen_classes.append(class_name)


    # dump bounding_boxes into a ".json" file
    new_temp_file = TEMP_FILES_PATH + "/" + file_id + "_ground_truth.json"
    gt_files.append(new_temp_file)
    with open(new_temp_file, 'w') as outfile:
        json.dump(bounding_boxes, outfile)

gt_classes = list(gt_counter_per_class.keys())
# let's sort the classes alphabetically
gt_classes = sorted(gt_classes)
n_classes = len(gt_classes)
#print(gt_classes)
```

```
#print(gt_counter_per_class)

"""
 Check format of the flag --set-class-iou (if used)
    e.g. check if class exists
"""
if specific_iou_flagged:
    n_args = len(args.set_class_iou)
    error_msg = \
        '\n --set-class-iou [class_1] [IoU_1] [class_2] [IoU_2] [...]'
    if n_args % 2 != 0:
        error('Error, missing arguments. Flag usage:' + error_msg)
    # [class_1] [IoU_1] [class_2] [IoU_2]
    # specific_iou_classes = ['class_1', 'class_2']
    specific_iou_classes = args.set_class_iou[::2] # even
    # iou_list = ['IoU_1', 'IoU_2']
    iou_list = args.set_class_iou[1::2] # odd
    if len(specific_iou_classes) != len(iou_list):
        error('Error, missing arguments. Flag usage:' + error_msg)
    for tmp_class in specific_iou_classes:
        if tmp_class not in gt_classes:
                    error('Error, unknown class \"' + tmp_class + '\". Flag usage:' + error_msg)
    for num in iou_list:
        if not is_float_between_0_and_1(num):
            error('Error, IoU must be between 0.0 and 1.0. Flag usage:' + error_msg)

"""
 detection-results
     Load each of the detection-results files into a temporary ".json" file.
"""
# get a list with the detection-results files
dr_files_list = glob.glob(DR_PATH + '/*.txt')
dr_files_list.sort()

for class_index, class_name in enumerate(gt_classes):
    bounding_boxes = []
    for txt_file in dr_files_list:
        #print(txt_file)
        # the first time it checks if all the corresponding ground-truth files exist
```

```
        file_id = txt_file.split(".txt",1)[0]
        file_id = os.path.basename(os.path.normpath(file_id))
        temp_path = os.path.join(GT_PATH, (file_id + ".txt"))
        if class_index == 0:
            if not os.path.exists(temp_path):
                error_msg = "Error. File not found: {}\n".format(temp_path)
                error_msg += "(You can avoid this error message by running extra/intersect-gt-and-dr.py)"
                error(error_msg)
        lines = file_lines_to_list(txt_file)
        for line in lines:
            try:
                tmp_class_name, confidence, left, top, right, bottom = line.split()
            except ValueError:
                error_msg = "Error: File " + txt_file + " in the wrong format.\n"
                error_msg += " Expected: <class_name> <confidence> <left> <top> <right> <bottom>\n"
                error_msg += " Received: " + line
                error(error_msg)
            if tmp_class_name == class_name:
                #print("match")
                bbox = left + " " + top + " " + right + " " +bottom
                bounding_boxes.append({"confidence": confidence, "file_id":file_id, "bbox":bbox})
                #print(bounding_boxes)
    # sort detection-results by decreasing confidence
    bounding_boxes.sort(key=lambda x:float(x['confidence']), reverse=True)
    with open(TEMP_FILES_PATH + "/" + class_name + "_dr.json", 'w') as outfile:
        json.dump(bounding_boxes, outfile)

"""
 Calculate the AP for each class
"""
sum_AP = 0.0
ap_dictionary = {}
lamr_dictionary = {}
# open file to store the output
with open(output_files_path + "/output.txt", 'w') as
```

```
output_file:
    output_file.write("# AP and precision/recall per class \n")
    count_true_positives = {}
    for class_index, class_name in enumerate(gt_classes):
        count_true_positives[class_name] = 0
        """
         Load detection-results of that class
        """
        dr_file = TEMP_FILES_PATH + "/" + class_name + "_dr.json"
        dr_data = json.load(open(dr_file))

        """
         Assign detection-results to ground-truth objects
        """
        nd = len(dr_data)
        tp = [0] * nd # creates an array of zeros of size nd
        fp = [0] * nd
        for idx, detection in enumerate(dr_data):
            file_id = detection["file_id"]
            if show_animation:
                # find ground truth image
                ground_truth_img = glob.glob1(IMG_PATH, file_id + ".*")
                #tifCounter = len(glob.glob1(myPath,"*.tif"))
                if len(ground_truth_img) == 0:
                    error("Error. Image not found with id : " + file_id)
                elif len(ground_truth_img) > 1:
                    error("Error. Multiple image with id : " + file_id)
                else: # found image
                    #print(IMG_PATH + "/" + ground_truth_img[0])
                    # Load image
                    img = cv2.imread(IMG_PATH + "/" + ground_truth_img[0])
                    # load image with draws of multiple detections
                    img_cumulative_path = output_files_path + "/images/" + ground_truth_img[0]
                    if os.path.isfile(img_cumulative_path):
```

```
                        img_cumulative = cv2.imread(img_cumulative_path)
                    else:
                        img_cumulative = img.copy()
                    # Add bottom border to image
                    bottom_border = 60
                    BLACK = [0, 0, 0]
                    img = cv2.copyMakeBorder(img, 0, bottom_border, 0, 0, cv2.BORDER_CONSTANT, value=BLACK)
            # assign detection-results to ground truth object if any
            # open ground-truth with that file_id
            gt_file = TEMP_FILES_PATH + "/" + file_id + "_ground_truth.json"
            ground_truth_data = json.load(open(gt_file))
            ovmax = -1
            gt_match = -1
            # load detected object bounding-box
            bb = [ float(x) for x in detection["bbox"].split() ]
            for obj in ground_truth_data:
                # look for a class_name match
                if obj["class_name"] == class_name:
                    bbgt = [ float(x) for x in obj["bbox"].split() ]
                    bi = [max(bb[0],bbgt[0]), max(bb[1],bbgt[1]), min(bb[2],bbgt[2]), min(bb[3],bbgt[3])]
                    iw = bi[2] - bi[0] + 1
                    ih = bi[3] - bi[1] + 1
                    if iw > 0 and ih > 0:
                        # compute overlap (IoU) = area of intersection / area of union
                        ua = (bb[2] - bb[0] + 1) * (bb[3] - bb[1] + 1) + (bbgt[2] - bbgt[0]
                                                 + 1) * (bbgt[3] - bbgt[1] + 1) - iw * ih
                        ov = iw * ih / ua
                        if ov > ovmax:
                            ovmax = ov
                            gt_match = obj

            # assign detection as true positive/don't care/false positive
            if show_animation:
                status = "NO MATCH FOUND!" # status is only used in the animation
```

```
            # set minimum overlap
            min_overlap = MINOVERLAP
            if specific_iou_flagged:
                if class_name in specific_iou_classes:
                    index = specific_iou_classes.index(class_name)
                    min_overlap = float(iou_list[index])
            if ovmax >= min_overlap:
                if "difficult" not in gt_match:
                        if not bool(gt_match["used"]):
                            # true positive
                            tp[idx] = 1
                            gt_match["used"] = True
                            count_true_positives[class_name] += 1
                            # update the ".json" file
                            with open(gt_file, 'w') as f:
                                    f.write(json.dumps(ground_truth_data))
                            if show_animation:
                                status = "MATCH!"
                        else:
                            # false positive (multiple detection)
                            fp[idx] = 1
                            if show_animation:
                                status = "REPEATED MATCH!"
            else:
                # false positive
                fp[idx] = 1
                if ovmax > 0:
                    status = "INSUFFICIENT OVERLAP"

            """
             Draw image to show animation
            """
            if show_animation:
                height, widht = img.shape[:2]
                # colors (OpenCV works with BGR)
                white = (255,255,255)
                light_blue = (255,200,100)
                green = (0,255,0)
                light_red = (30,30,255)
                # 1st line
                margin = 10
                v_pos = int(height - margin - (
```

```
bottom_border / 2.0))
                text = "Image: " + ground_truth_img[0] + " "
                img, line_width = draw_text_in_image(img, text, (margin, v_pos), white, 0)
                text = "Class [" + str(class_index) + "/" + str(n_classes) + "]: " + class_name + " "
                img, line_width = draw_text_in_image(img, text, (margin + line_width, v_pos), light_blue, line_width)
                if ovmax != -1:
                    color = light_red
                    if status == "INSUFFICIENT OVERLAP":
                        text = "IoU: {0:.2f}% ".format(ovmax*100) + "< {0:.2f}% ".format(min_overlap*100)
                    else:
                        text = "IoU: {0:.2f}% ".format(ovmax*100) + ">= {0:.2f}% ".format(min_overlap*100)
                        color = green
                    img, _ = draw_text_in_image(img, text, (margin + line_width, v_pos), color, line_width)
                # 2nd line
                v_pos += int(bottom_border / 2.0)
                rank_pos = str(idx+1) # rank position (idx starts at 0)
                text = "Detection #rank: " + rank_pos + " confidence: {0:.2f}% ".format(float(detection["confidence"])*100)
                img, line_width = draw_text_in_image(img, text, (margin, v_pos), white, 0)
                color = light_red
                if status == "MATCH!":
                    color = green
                text = "Result: " + status + " "
                img, line_width = draw_text_in_image(img, text, (margin + line_width, v_pos), color, line_width)

                font = cv2.FONT_HERSHEY_SIMPLEX
                if ovmax > 0: # if there is intersections between the bounding-boxes
                    bbgt = [ int(round(float(x))) for x in gt_match["bbox"].split() ]
                    cv2.rectangle(img,(bbgt[0],bbgt[1]),(bbgt[2],bbgt[3]),light_blue,2)
                    cv2.rectangle(img_cumulative,(bbgt[0],bbgt[1]),(bbgt[2],bbgt[3]),light_blue,2)
```

```
                    cv2.putText(img_cumulative, class_name, (bbgt[0],bbgt[1] - 5), font, 0.6, light_blue, 1, cv2.LINE_AA)
                bb = [int(i) for i in bb]
                cv2.rectangle(img,(bb[0],bb[1]),(bb[2],bb[3]),color,2)
                cv2.rectangle(img_cumulative,(bb[0],bb[1]),(bb[2],bb[3]),color,2)
                cv2.putText(img_cumulative, class_name, (bb[0],bb[1] - 5), font, 0.6, color, 1, cv2.LINE_AA)
                # show image
                cv2.imshow("Animation", img)
                cv2.waitKey(20) # show for 20 ms
                # save image to output
                output_img_path = output_files_path + "/images/detections_one_by_one/" + class_name + "_detection" + str(idx) + ".jpg"
                cv2.imwrite(output_img_path, img)
                # save the image with all the objects drawn to it
                cv2.imwrite(img_cumulative_path, img_cumulative)

        #print(tp)
        # compute precision/recall
        cumsum = 0
        for idx, val in enumerate(fp):
            fp[idx] += cumsum
            cumsum += val
        cumsum = 0
        for idx, val in enumerate(tp):
            tp[idx] += cumsum
            cumsum += val
        #print(tp)
        rec = tp[:]
        for idx, val in enumerate(tp):
            rec[idx] = float(tp[idx]) / gt_counter_per_class[class_name]
        #print(rec)
        prec = tp[:]
        for idx, val in enumerate(tp):
            prec[idx] = float(tp[idx]) / (fp[idx] + tp[idx])
        #print(prec)

        ap, mrec, mprec = voc_ap(rec[:], prec[:])
```

```
        sum_AP += ap
        text = "{0:.2f}%".format(ap*100) + " = " + class_name + " AP " #class_name + " AP = {0:.2f}%".format(ap*100)
        """
         Write to output.txt
        """
        rounded_prec = [ '%.2f' % elem for elem in prec ]
        rounded_rec = [ '%.2f' % elem for elem in rec ]
        output_file.write(text + "\n Precision: " + str(rounded_prec) + "\n Recall :" + str(rounded_rec) + "\n\n")
        if not args.quiet:
            print(text)
        ap_dictionary[class_name] = ap

        n_images = counter_images_per_class[class_name]
        lamr, mr, fppi = log_average_miss_rate(np.array(rec), np.array(fp), n_images)
        lamr_dictionary[class_name] = lamr

        """
         Draw plot
        """
        if draw_plot:
            plt.plot(rec, prec, '-o')
            # add a new penultimate point to the list (mrec[-2], 0.0)
            # since the last line segment (and respective area) do not affect the AP value
            area_under_curve_x = mrec[:-1] + [mrec[-2]] + [mrec[-1]]
            area_under_curve_y = mprec[:-1] + [0.0] + [mprec[-1]]
            plt.fill_between(area_under_curve_x, 0, area_under_curve_y, alpha=0.2, edgecolor='r')
            # set window title
            fig = plt.gcf() # gcf - get current figure
            fig.canvas.set_window_title('AP ' + class_name)
            # set plot title
            plt.title('class: ' + text)
            #plt.suptitle('This is a somewhat long figure title', fontsize=16)
            # set axis titles
            plt.xlabel('Recall')
            plt.ylabel('Precision')
```

```
            # optional - set axes
            axes = plt.gca() # gca - get current axes
            axes.set_xlim([0.0,1.0])
            axes.set_ylim([0.0,1.05]) # .05 to give some extra space
            # Alternative option -> wait for button to be pressed
            #while not plt.waitforbuttonpress(): pass # wait for key display
            # Alternative option -> normal display
            #plt.show()
            # save the plot
            fig.savefig(output_files_path + "/classes/" + class_name + ".png")
            plt.cla() # clear axes for next plot

    if show_animation:
        cv2.destroyAllWindows()

    output_file.write("\n# mAP of all classes\n")
    mAP = sum_AP / n_classes
    text = "mAP = {0:.2f}%".format(mAP*100)
    output_file.write(text + "\n")
    print(text)

"""
 Draw false negatives
"""
pink = (203,192,255)
'''
for tmp_file in gt_files:
    ground_truth_data = json.load(open(tmp_file))
    #print(ground_truth_data)
    # get name of corresponding image
    start = TEMP_FILES_PATH + '/'
    img_id = tmp_file[tmp_file.find(start)+len(start): tmp_file.rfind('_ground_truth.json')]
    img_cumulative_path = output_files_path + "/images/" + img_id + ".jpg"
    img = cv2.imread(img_cumulative_path)
    if img is None:
        img_path = IMG_PATH + '/' + img_id + ".jpg"
        img = cv2.imread(img_path)
    # draw false negatives
    for obj in ground_truth_data:
        if not obj['used']:
```

```
            bbgt = [ int(round(float(x))) for x in obj["bbox"].split() ]
            cv2.rectangle(img,(bbgt[0],bbgt[1]),(bbgt[2],bbgt[3]),pink,2)
    cv2.imwrite(img_cumulative_path, img)
'''
# remove the temp_files directory
shutil.rmtree(TEMP_FILES_PATH)

"""
 Count total of detection-results
"""
# iterate through all the files
det_counter_per_class = {}
for txt_file in dr_files_list:
    # get lines to list
    lines_list = file_lines_to_list(txt_file)
    for line in lines_list:
        class_name = line.split()[0]
        # check if class is in the ignore list, if yes skip
        if class_name in args.ignore:
            continue
        # count that object
        if class_name in det_counter_per_class:
            det_counter_per_class[class_name] += 1
        else:
            # if class didn't exist yet
            det_counter_per_class[class_name] = 1
#print(det_counter_per_class)
dr_classes = list(det_counter_per_class.keys())


"""
 Plot the total number of occurences of each class in the ground-truth
"""
if draw_plot:
    window_title = "ground-truth-info"
    plot_title = "ground-truth\n"
    plot_title += "(" + str(len(ground_truth_files_list)) + " files and " + str(n_classes) + " classes)"
    x_label = "Number of objects per class"
    output_path = output_files_path + "/ground-truth-info.png"
    to_show = False
```

```
    plot_color = 'forestgreen'
    draw_plot_func(
        gt_counter_per_class,
        n_classes,
        window_title,
        plot_title,
        x_label,
        output_path,
        to_show,
        plot_color,
        '',
        )

"""
 Write number of ground-truth objects per class to results.txt
"""
with open(output_files_path + "/output.txt", 'a') as output_file:
    output_file.write("\n# Number of ground-truth objects per class\n")
    for class_name in sorted(gt_counter_per_class):
        output_file.write(class_name + ": " + str(gt_counter_per_class[class_name]) + "\n")

"""
 Finish counting true positives
"""
for class_name in dr_classes:
    # if class exists in detection-result but not in ground-truth then there are no true positives in that class
    if class_name not in gt_classes:
        count_true_positives[class_name] = 0
#print(count_true_positives)

"""
 Plot the total number of occurences of each class in the "detection-results" folder
"""
if draw_plot:
    window_title = "detection-results-info"
    # Plot title
    plot_title = "detection-results\n"
    plot_title += "(" + str(len(dr_files_list)) + " files and "
```

```
    count_non_zero_values_in_dictionary = sum(int(x) > 0 for x in list(det_counter_per_class.values()))
    plot_title += str(count_non_zero_values_in_dictionary) + " detected classes)"
    # end Plot title
    x_label = "Number of objects per class"
    output_path = output_files_path + "/detection-results-info.png"
    to_show = False
    plot_color = 'forestgreen'
    true_p_bar = count_true_positives
    draw_plot_func(
        det_counter_per_class,
        len(det_counter_per_class),
        window_title,
        plot_title,
        x_label,
        output_path,
        to_show,
        plot_color,
        true_p_bar
        )

"""
 Write number of detected objects per class to output.txt
"""
with open(output_files_path + "/output.txt", 'a') as output_file:
    output_file.write("\n# Number of detected objects per class\n")
    for class_name in sorted(dr_classes):
        n_det = det_counter_per_class[class_name]
        text = class_name + ": " + str(n_det)
        text += " (tp:" + str(count_true_positives[class_name]) + ""
        text += ", fp:" + str(n_det - count_true_positives[class_name]) + ")\n"
        output_file.write(text)

"""
 Draw log-average miss rate plot (Show lamr of all classes in decreasing order)
"""
if draw_plot:
    window_title = "lamr"
    plot_title = "log-average miss rate"
```

```
    x_label = "log-average miss rate"
    output_path = output_files_path + "/lamr.png"
    to_show = False
    plot_color = 'royalblue'
    draw_plot_func(
        lamr_dictionary,
        n_classes,
        window_title,
        plot_title,
        x_label,
        output_path,
        to_show,
        plot_color,
        ""
        )

"""
 Draw mAP plot (Show AP's of all classes in decreasing order)
"""
if draw_plot:
    window_title = "mAP"
    plot_title = "mAP = {0:.2f}%".format(mAP*100)
    x_label = "Average Precision"
    output_path = output_files_path + "/mAP.png"
    to_show = True
    plot_color = 'royalblue'
    draw_plot_func(
        ap_dictionary,
        n_classes,
        window_title,
        plot_title,
        x_label,
        output_path,
        to_show,
        plot_color,
        ""
        )

```

Appendix L

YOLO Processing Video file

```python
import cv2
from darkflow.net.build import TFNet
import numpy as np
import time
import numpy
option = {
    'model': 'cfg/yolo3c.cfg',
    'load': 7000,
    'threshold': 0.4315,
    'gpu': 1.0
}

tfnet = TFNet(option)

capture = cv2.VideoCapture('Test_video2.mp4')
colors = [tuple(255 * np.random.rand(3)) for i in range(5)]


while (capture.isOpened()):
    stime = time.time()
    ret, frame = capture.read()

    if ret:
        results = tfnet.return_predict(frame)
        for color, result in zip(colors, results):
            tl = (result['topleft']['x'], result['topleft']['y'])
            br = (result['bottomright']['x'], result['bottomright']['y'])
            label = result['label']
            frame = cv2.rectangle(frame, tl, br, color, 7)
            frame = cv2.putText(frame, label, tl, cv2.FONT_HERSHEY_COMPLEX, 2, (255, 0, 0), 2)
        cv2.imshow('frame', frame)
        print('FPS {:.1f}'.format(1 / (time.time() - stime)))
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    else:
        capture.release()
        cv2.destroyAllWindows()
        break

```

Appendix M

YOLO labels file

```
green
red
yellow
```

Appendix N

YOLO CFG configuration file

```
[net]
# Testing
batch=1
subdivisions=1
# Training
# batch=64
# subdivisions=8
width=608
height=608
channels=3
momentum=0.9
decay=0.0005
angle=0
saturation = 1.5
exposure = 1.5
hue=0
#0.001 - 0.0000005
learning_rate=0.0000005
burn_in=1000
max_batches = 500200
policy=steps
steps=400000,450000
scales=.1,.1

[convolutional]
batch_normalize=1
filters=32
size=3
stride=1
pad=1
activation=leaky

[maxpool]
size=2
stride=2

[convolutional]
batch_normalize=1
filters=64
size=3
stride=1
pad=1
activation=leaky

[maxpool]
size=2
```

```
stride=2

[convolutional]
batch_normalize=1
filters=128
size=3
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=64
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=128
size=3
stride=1
pad=1
activation=leaky

[maxpool]
size=2
stride=2

[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=128
size=1
stride=1
pad=1
activation=leaky

```

```
[convolutional]
batch_normalize=1
filters=256
size=3
stride=1
pad=1
activation=leaky

[maxpool]
size=2
stride=2

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=3
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=256
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
```

```
filters=512
size=3
stride=1
pad=1
activation=leaky

[maxpool]
size=2
stride=2

[convolutional]
batch_normalize=1
filters=1024
size=3
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=1024
size=3
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=512
size=1
stride=1
pad=1
activation=leaky

[convolutional]
batch_normalize=1
filters=1024
size=3
```

```
stride=1
pad=1
activation=leaky


#######

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=1024
activation=leaky

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=1024
activation=leaky

[route]
layers=-9

[convolutional]
batch_normalize=1
size=1
stride=1
pad=1
filters=64
activation=leaky

[reorg]
stride=2

[route]
layers=-1,-4

[convolutional]
batch_normalize=1
size=3
stride=1
pad=1
filters=1024
```

```
activation=leaky

[convolutional]
size=1
stride=1
pad=1
filters=40
activation=linear


[region]
anchors =  0.57273, 0.677385, 1.87446, 2.06253, 3.33843, 5.47434, 7.88282, 3.52778, 9.77052, 9.16828
bias_match=1
classes=3
coords=4
num=5
softmax=1
jitter=.3
rescore=1

object_scale=5
noobject_scale=1
class_scale=1
coord_scale=1

absolute=1
thresh = .1
random=1

```

Appendix O

Patrol file

```python
#!/usr/bin/env python
#Author @Yrian_Hovde_Øksne Date
#This script has the purpose of moving a ROS robot through a predetermined path by
#using a list of waypoints from the rviz coordinate system as input for the Patrol class



import rospy
import actionlib
import tf
from move_base_msgs.msg import MoveBaseAction, MoveBaseGoal

#The list of points to patrol, to set new waypoints, use rviz terminal information.
#set points with this format '['point name', (x-coord, y-coord, angle) ]'
waypoints = [
    ['one', (-2.778, 0.219, 1.584)],
    ['two', (-2.792, 3.165, -0.033)],
    ['three', (3.956, 3.053, -1.605)],
    ['four', (3.976, -3.025, 3.108)],
    ['five', (0.347, -2.977, 2.369)]
]

class Patrol:

    def __init__(self):
        self.client = actionlib.SimpleActionClient('move_base', MoveBaseAction)
        self.client.wait_for_server()

    def set_goal_to_point(self, point):

        goal = MoveBaseGoal()
        goal.target_pose.header.frame_id = "map"
        goal.target_pose.header.stamp = rospy.Time.now()
        goal.target_pose.pose.position.x = point[0]
        goal.target_pose.pose.position.y = point[1]
        quaternion = tf.transformations.quaternion_from_euler(0.0, 0.0, point[2])
        goal.target_pose.pose.orientation.x = quaternion[0]
        goal.target_pose.pose.orientation.y = quaternion[1]
        goal.target_pose.pose.orientation.z = quaternion[2]
        goal.target_pose.pose.orientation.w = quaternion[3]
```

```python

        self.client.send_goal(goal)
        wait = self.client.wait_for_result()
        if not wait:
            rospy.logerr("Action server not available")
            rospy.signal_shutdown("Action server not available")
        else:
            return self.client.get_result()


if __name__ == '__main__':
    rospy.init_node('patrolling')
    try:
        p = Patrol()
        while not rospy.is_shutdown():
            for i, w in enumerate(waypoints):
                rospy.loginfo("Sending waypoint %d - %s", i, w[0])
                p.set_goal_to_point(w[1])
    except rospy.ROSException:
        rospy.logerr("Something went wrong when sending the waypoints")
```

Appendix P

WAM-V simulation model

```
<?xml version="1.0" encoding="utf-8"?>
<!--
=====================================================================
============ -->
<!-- |    This document was autogenerated by xacro from
/home/markus/vrx_ws/src/vrx/wamv_gazebo/urdf/wamv_gazebo.urdf.xacro | -
->
<!-- |    EDITING THIS FILE BY HAND IS NOT RECOMMENDED
| -->
<!--
=====================================================================
============ -->
<!-- Basic WAM-V with gazebo plugins for dynamics -->
<robot name="WAM-V">
  <link name="left_battery_link">
    <visual name="left_battery_visual">
      <origin rpy="0 0 0" xyz="0 -1.03 -.45"/>
      <geometry>
        <mesh
filename="package://vrx_gazebo/models/battery/mesh/battery.dae"/>
      </geometry>
    </visual>
    <collision name="left_battery_collision">
      <origin rpy="0 0 0" xyz="0 0 .15"/>
      <geometry>
        <box size=".6 .4 .31"/>
      </geometry>
    </collision>
    <inertial>
      <origin rpy=" 0 0 0" xyz="0 0 .15"/>
      <mass value="23.5"/>
      <inertia ixx="0.5015291666666667" ixy="0" ixz="0"
iyy="0.8931958333333333" iyz="0" izz="1.0183333333333333"/>
    </inertial>
  </link>
  <joint name="left_chasis_battery_joint" type="fixed">
    <origin rpy="0 0 0" xyz="0 1 0.45"/>
    <parent link="base_link"/>
    <child link="left_battery_link"/>
  </joint>
  <link name="right_battery_link">
    <visual name="right_battery_visual">
      <origin rpy="0 0 0" xyz="0 -1.03 -.45"/>
      <geometry>
        <mesh
filename="package://vrx_gazebo/models/battery/mesh/battery.dae"/>
      </geometry>
    </visual>
    <collision name="right_battery_collision">
      <origin rpy="0 0 0" xyz="0 0 .15"/>
      <geometry>
        <box size=".6 .4 .31"/>
      </geometry>
    </collision>
    <inertial>
      <origin rpy=" 0 0 0" xyz="0 0 .15"/>
      <mass value="23.5"/>
```

```
      <inertia ixx="0.5015291666666667" ixy="0" ixz="0"
iyy="0.8931958333333333" iyz="0" izz="1.0183333333333333"/>
    </inertial>
  </link>
  <joint name="right_chasis_battery_joint" type="fixed">
    <origin rpy="0 0 0" xyz="0 -1 0.45"/>
    <parent link="base_link"/>
    <child link="right_battery_link"/>
  </joint>
  <link name="base_link">
  </link>
  <joint name="dummy_joint" type="fixed">
    <parent link="base_link"/>
    <child link="dummy_link"/>
  </joint>
  <!-- Basic model of the 16' WAM-V USV -->
  <link name="dummy_link">
    <visual>
      <geometry>
        <mesh filename="package://wamv_description/models/WAM-V-
Base/mesh/WAM-V-Base.dae"/>
      </geometry>
    </visual>
    <!-- Main float -->
    <collision name="left_float">
      <origin rpy="0 1.57 0" xyz="-0.4 1.03 0.2"/>
      <geometry>
        <cylinder length="4" radius="0.2"/>
      </geometry>
    </collision>
    <collision name="left_mid_float">
      <origin rpy="0 1.38 0" xyz="1.85 1.03 0.3"/>
      <geometry>
        <cylinder length="0.5" radius="0.17"/>
      </geometry>
    </collision>
    <collision name="left_front_float">
      <origin rpy="0 1.3 0" xyz="2.3 1.03 0.4"/>
      <geometry>
        <cylinder length="0.45" radius="0.12"/>
      </geometry>
    </collision>
    <!-- Front beam -->
    <collision name="front_left_beam_lower">
      <origin rpy="0.78 0 0" xyz="0.9 0.85 1"/>
      <geometry>
        <cylinder length="0.5" radius="0.04"/>
      </geometry>
    </collision>
    <collision name="front_left_beam_upper">
      <origin rpy="1.35 0 0" xyz="0.9 0.6 1.18"/>
      <geometry>
        <cylinder length="0.2" radius="0.04"/>
      </geometry>
    </collision>
    <!-- Mid beam -->
    <collision name="mid_left_beam_lower">
      <origin rpy="0.1 0.25 0" xyz="-0.65 0.99 0.7"/>
```

```
      <geometry>
        <cylinder length="0.45" radius="0.03"/>
      </geometry>
    </collision>
    <collision name="mid_left_beam_medium">
      <origin rpy="0.75 0.25 0" xyz="-0.57 0.87 1.05"/>
      <geometry>
        <cylinder length="0.32" radius="0.03"/>
      </geometry>
    </collision>
    <collision name="mid_left_beam_upper">
      <origin rpy="1.35 0.25 0" xyz="-0.55 0.65 1.17"/>
      <geometry>
        <cylinder length="0.3" radius="0.03"/>
      </geometry>
    </collision>
    <!-- Rear beam -->
    <collision name="rear_left_beam_lower">
      <origin rpy="0 -0.15 0" xyz="-0.74 1.03 0.7"/>
      <geometry>
        <cylinder length="0.45" radius="0.03"/>
      </geometry>
    </collision>
    <collision name="rear_left_beam_medium">
      <origin rpy="0.75 -0.15 0" xyz="-0.79 0.91 1.05"/>
      <geometry>
        <cylinder length="0.32" radius="0.03"/>
      </geometry>
    </collision>
    <collision name="rear_left_beam_upper">
      <origin rpy="1.45 -0.15 0" xyz="-0.81 0.67 1.18"/>
      <geometry>
        <cylinder length="0.3" radius="0.03"/>
      </geometry>
    </collision>
    <!-- Joint -->
    <collision name="left_joint">
      <origin rpy="0 -0.6 0" xyz="0.58 1.03 0.6"/>
      <geometry>
        <box size="0.65 0.2 0.1"/>
      </geometry>
    </collision>
    <!-- Main float -->
    <collision name="right_float">
      <origin rpy="0 1.57 0" xyz="-0.4 -1.03 0.2"/>
      <geometry>
        <cylinder length="4" radius="0.2"/>
      </geometry>
    </collision>
    <collision name="right_mid_float">
      <origin rpy="0 1.38 0" xyz="1.85 -1.03 0.3"/>
      <geometry>
        <cylinder length="0.5" radius="0.17"/>
      </geometry>
    </collision>
    <collision name="right_front_float">
      <origin rpy="0 1.3 0" xyz="2.3 -1.03 0.4"/>
      <geometry>
```

```
    <cylinder length="0.45" radius="0.12"/>
  </geometry>
</collision>
<!-- Front beam -->
<collision name="front_right_beam_lower">
  <origin rpy="-0.78 0 0" xyz="0.9 -0.85 1"/>
  <geometry>
    <cylinder length="0.5" radius="0.04"/>
  </geometry>
</collision>
<collision name="front_right_beam_upper">
  <origin rpy="-1.35 0 0" xyz="0.9 -0.6 1.18"/>
  <geometry>
    <cylinder length="0.2" radius="0.04"/>
  </geometry>
</collision>
<!-- Mid beam -->
<collision name="mid_right_beam_lower">
  <origin rpy="-0.1 0.25 0" xyz="-0.65 -0.99 0.7"/>
  <geometry>
    <cylinder length="0.45" radius="0.03"/>
  </geometry>
</collision>
<collision name="mid_right_beam_medium">
  <origin rpy="-0.75 0.25 0" xyz="-0.57 -0.87 1.05"/>
  <geometry>
    <cylinder length="0.32" radius="0.03"/>
  </geometry>
</collision>
<collision name="mid_right_beam_upper">
  <origin rpy="-1.35 0.25 0" xyz="-0.55 -0.65 1.17"/>
  <geometry>
    <cylinder length="0.3" radius="0.03"/>
  </geometry>
</collision>
<!-- Rear beam -->
<collision name="rear_right_beam_lower">
  <origin rpy="0 -0.15 0" xyz="-0.74 -1.03 0.7"/>
  <geometry>
    <cylinder length="0.45" radius="0.03"/>
  </geometry>
</collision>
<collision name="rear_right_beam_medium">
  <origin rpy="-0.75 -0.15 0" xyz="-0.79 -0.91 1.05"/>
  <geometry>
    <cylinder length="0.32" radius="0.03"/>
  </geometry>
</collision>
<collision name="rear_right_beam_upper">
  <origin rpy="-1.45 -0.15 0" xyz="-0.81 -0.67 1.18"/>
  <geometry>
    <cylinder length="0.3" radius="0.03"/>
  </geometry>
</collision>
<!-- Joint -->
<collision name="right_joint">
  <origin rpy="0 -0.6 0" xyz="0.58 -1.03 0.6"/>
  <geometry>
```

```
        <box size="0.65 0.2 0.1"/>
      </geometry>
    </collision>
    <!-- Top base -->
    <collision name="top_base">
      <origin rpy="0 0 0" xyz="0 -0 1.25"/>
      <geometry>
        <box size="1.85 1 0.1"/>
      </geometry>
    </collision>
    <inertial>
      <!-- From WAM-V spec sheet -->
      <mass value="180.0"/>
      <inertia ixx="120.0" ixy="0.0" ixz="0.0" iyy="393" iyz="0.0"
izz="446.0"/>
    </inertial>
  </link>
  <link name="left_engine_link">
    <visual>
      <geometry>
        <mesh
filename="package://wamv_description/models/engine/mesh/engine.dae"/>
      </geometry>
    </visual>
    <collision name="left_engine_vertical_axis_collision">
      <origin rpy="0 0 0" xyz="-0.16 0 -0.24"/>
      <geometry>
        <box size="0.2 0.15 0.83"/>
      </geometry>
    </collision>
    <collision name="left_engine_rear_end_collision">
      <origin rpy="0 0 0" xyz="-0.34 0 0.12"/>
      <geometry>
        <box size="0.12 0.15 0.12"/>
      </geometry>
    </collision>
    <inertial>
      <mass value="15"/>
      <inertia ixx="0.889245" ixy="0.0" ixz="0.0" iyy="0.911125"
iyz="0.0" izz="0.078125"/>
    </inertial>
  </link>
  <link name="left_propeller_link">
    <visual>
      <geometry>
        <mesh
filename="package://wamv_description/models/propeller/mesh/propeller.da
e"/>
      </geometry>
    </visual>
    <collision name="left_propeller_collision">
      <origin rpy="0 1.57 0" xyz="-0.08 0 0"/>
      <geometry>
        <cylinder length="0.18" radius="0.24"/>
      </geometry>
    </collision>
    <inertial>
      <mass value="0.5"/>
```

```
      <inertia ixx="0.008545" ixy="0.0" ixz="0.0" iyy="0.008545"
iyz="0.0" izz="0.0144"/>
    </inertial>
  </link>
  <joint name="left_chasis_engine_joint" type="revolute">
    <axis xyz="0 0 1"/>
    <limit effort="10" lower="-3.14159265359" upper="3.14159265359"
velocity="10"/>
    <origin rpy="0.0 0.0 0.0" xyz="-2.373776 1.027135 0.318237"/>
    <parent link="base_link"/>
    <child link="left_engine_link"/>
  </joint>
  <joint name="left_engine_propeller_joint" type="continuous">
    <axis rpy="0 0 0" xyz="1 0 0"/>
    <parent link="left_engine_link"/>
    <child link="left_propeller_link"/>
    <origin rpy="0 0 0" xyz="-0.278156 0 -0.509371"/>
    <limit effort="100" velocity="100"/>
    <dynamics damping="0.05" friction="0.05"/>
  </joint>
  <link name="right_engine_link">
    <visual>
      <geometry>
        <mesh
filename="package://wamv_description/models/engine/mesh/engine.dae"/>
      </geometry>
    </visual>
    <collision name="right_engine_vertical_axis_collision">
      <origin rpy="0 0 0" xyz="-0.16 0 -0.24"/>
      <geometry>
        <box size="0.2 0.15 0.83"/>
      </geometry>
    </collision>
    <collision name="right_engine_rear_end_collision">
      <origin rpy="0 0 0" xyz="-0.34 0 0.12"/>
      <geometry>
        <box size="0.12 0.15 0.12"/>
      </geometry>
    </collision>
    <inertial>
      <mass value="15"/>
      <inertia ixx="0.889245" ixy="0.0" ixz="0.0" iyy="0.911125"
iyz="0.0" izz="0.078125"/>
    </inertial>
  </link>
  <link name="right_propeller_link">
    <visual>
      <geometry>
        <mesh
filename="package://wamv_description/models/propeller/mesh/propeller.da
e"/>
      </geometry>
    </visual>
    <collision name="right_propeller_collision">
      <origin rpy="0 1.57 0" xyz="-0.08 0 0"/>
      <geometry>
        <cylinder length="0.18" radius="0.24"/>
      </geometry>
```

```
    </collision>
    <inertial>
      <mass value="0.5"/>
      <inertia ixx="0.008545" ixy="0.0" ixz="0.0" iyy="0.008545"
iyz="0.0" izz="0.0144"/>
    </inertial>
  </link>
  <joint name="right_chasis_engine_joint" type="revolute">
    <axis xyz="0 0 1"/>
    <limit effort="10" lower="-3.14159265359" upper="3.14159265359"
velocity="10"/>
    <origin rpy="0.0 0.0 0.0" xyz="-2.373776 -1.027135 0.318237"/>
    <parent link="base_link"/>
    <child link="right_engine_link"/>
  </joint>
  <joint name="right_engine_propeller_joint" type="continuous">
    <axis rpy="0 0 0" xyz="1 0 0"/>
    <parent link="right_engine_link"/>
    <child link="right_propeller_link"/>
    <origin rpy="0 0 0" xyz="-0.278156 0 -0.509371"/>
    <limit effort="100" velocity="100"/>
    <dynamics damping="0.05" friction="0.05"/>
  </joint>
  <gazebo>
    <plugin filename="libusv_gazebo_thrust_plugin.so"
name="wamv_gazebo_thrust">
      <cmdTimeout>1.0</cmdTimeout>
      <robotNamespace>wamv</robotNamespace>
      <thruster>
        <!-- Required Parameters -->
        <linkName>left_propeller_link</linkName>
        <propJointName>left_engine_propeller_joint</propJointName>
        <engineJointName>left_chasis_engine_joint</engineJointName>
        <cmdTopic>thrusters/left_thrust_cmd</cmdTopic>
        <angleTopic>thrusters/left_thrust_angle</angleTopic>
        <enableAngle>true</enableAngle>
        <!-- Optional Parameters -->
        <mappingType>1</mappingType>
        <maxCmd>1.0</maxCmd>
        <maxForceFwd>250.0</maxForceFwd>
        <maxForceRev>-100.0</maxForceRev>
        <maxAngle>1.57079632679</maxAngle>
      </thruster>
      <thruster>
        <!-- Required Parameters -->
        <linkName>right_propeller_link</linkName>
        <propJointName>right_engine_propeller_joint</propJointName>
        <engineJointName>right_chasis_engine_joint</engineJointName>
        <cmdTopic>thrusters/right_thrust_cmd</cmdTopic>
        <angleTopic>thrusters/right_thrust_angle</angleTopic>
        <enableAngle>true</enableAngle>
        <!-- Optional Parameters -->
        <mappingType>1</mappingType>
        <maxCmd>1.0</maxCmd>
        <maxForceFwd>250.0</maxForceFwd>
        <maxForceRev>-100.0</maxForceRev>
        <maxAngle>1.57079632679</maxAngle>
      </thruster>
```

```
    </plugin>
  </gazebo>
  <!--Gazebo Plugin for simulating WAM-V dynamics-->
  <gazebo>
    <plugin filename="libusv_gazebo_dynamics_plugin.so"
name="usv_dynamics_wamv_dynamics_plugin">
      <bodyName>base_link</bodyName>
      <!-- Must be same as the ocean model!-->
      <waterLevel>0</waterLevel>
      <waterDensity>997.8</waterDensity>
      <!-- Added mass -->
      <xDotU>0.0</xDotU>
      <yDotV>0.0</yDotV>
      <nDotR>0.0</nDotR>
      <!-- Linear and quadratic drag -->
      <xU>51.3</xU>
      <xUU>72.4</xUU>
      <yV>40.0</yV>
      <yVV>0.0</yVV>
      <zW>500.0</zW>
      <kP>50.0</kP>
      <mQ>50.0</mQ>
      <nR>400.0</nR>
      <nRR>0.0</nRR>
      <!-- General dimensions -->
      <!--<boatArea>2.2</boatArea>-->
      <hullRadius>0.213</hullRadius>
      <boatWidth>2.4</boatWidth>
      <boatLength>4.9</boatLength>
      <!-- Length discretization, AKA, "N" -->
      <length_n>2</length_n>
      <!-- Wave model -->
      <wave_model>ocean_waves</wave_model>
    </plugin>
  </gazebo>
  <link name="imu_wamv_link">
    <visual name="imu_wamv_visual">
      &gt;

      <geometry>
        <box size="0.05 0.025 0.005"/>
      </geometry>
      <material name="imu_wamv_visual_material">
        <color rgba="1.0 0.0 00 1.0"/>
      </material>
    </visual>
    <inertial>
      <mass value="0.1"/>
      <inertia ixx="0.000083" ixy="0.0" ixz="0.0" iyy="0.000083"
iyz="0.0" izz="0.0125"/>
    </inertial>
  </link>
  <joint name="imu_wamv_joint" type="revolute">
    <axis xyz="0 0 1"/>
    <limit effort="1000.0" lower="0.0" upper="0" velocity="0"/>
    <origin rpy="0 0 3.14159" xyz="0.3 -0.2 1.3"/>
    <parent link="base_link"/>
    <child link="imu_wamv_link"/>
```

```
  </joint>
  <gazebo>
    <plugin filename="libhector_gazebo_ros_imu.so"
name="imu_plugin_imu_wamv">
      <updateRate>15.0</updateRate>
      <bodyName>imu_wamv_link</bodyName>
      <topicName>wamv/sensors/imu/imu/data</topicName>
      <serviceName>wamv/sensors/imu_service</serviceName>
      <!-- Manually prepend namespace to tf frame. -->
      <frameId>wamv/imu_wamv_link</frameId>
      <alwaysOn>true</alwaysOn>
      <accelOffset>0.0 0.0 0.0</accelOffset>
      <accelDrift>0.0 0.0 0.0</accelDrift>
      <accelDriftFrequency>0.00027 0.00027
0.000027</accelDriftFrequency>
      <accelGaussianNoise>0.0 0.0 0.0</accelGaussianNoise>
      <rateOffset>0.0 0.0 0.0</rateOffset>
      <rateDrift>0.0 0.0 0.0</rateDrift>
      <rateDriftFrequency>0.00027 0.00027 0.000027</rateDriftFrequency>
      <rateGaussianNoise>0.0 0.0 0.0</rateGaussianNoise>
      <yawOffset>0.0</yawOffset>
      <yawDrift>0.0</yawDrift>
      <yawDriftFrequency>0.00027</yawDriftFrequency>
      <yawGaussianNoise>0.0</yawGaussianNoise>
    </plugin>
  </gazebo>
  <link name="gps_wamv_link">
    <visual name="gps_wamv_visual">
      <geometry>
        <mesh
filename="package://wamv_gazebo/models/gps/mesh/gps.dae"/>
      </geometry>
    </visual>
    <collision name="gps_wamv_collision_base">
      <origin rpy="0 0 0" xyz="0 0 0.025"/>
      <geometry>
        <cylinder length="0.05" radius="0.015"/>
      </geometry>
    </collision>
    <collision name="gps_wamv_collision_antenna">
      <origin rpy="0 0 0" xyz="0 0 0.11"/>
      <geometry>
        <cylinder length="0.1" radius="0.15"/>
      </geometry>
    </collision>
    <inertial>
      <mass value="1"/>
      <inertia ixx="0.006458" ixy="0.0" ixz="0.0" iyy="0.006458"
iyz="0.0" izz="0.01125"/>
    </inertial>
  </link>
  <joint name="gps_wamv_joint" type="revolute">
    <axis xyz="0 0 1"/>
    <limit effort="1000.0" lower="0.0" upper="0" velocity="0"/>
    <origin rpy="0 0 0" xyz="-0.85 100.0 100.0"/>
    <parent link="base_link"/>
    <child link="gps_wamv_link"/>
  </joint>
```

```
  <gazebo>
    <plugin filename="libhector_gazebo_ros_gps.so"
name="gps_plugin_gps_wamv">
      <updateRate>15.0</updateRate>
      <alwaysOn>true</alwaysOn>
      <bodyName>gps_wamv_link</bodyName>
      <!-- Manually prepend namespace to tf frame. -->
      <frameId>wamv/gps_wamv_link</frameId>
      <topicName>wamv/sensors/gps/gps/fix</topicName>
      <!-- Note that the hector_gazebo_ros_gps plugin uses NWU
coordinates
             for reporting velocities.  This may cause problems if
localization
             requires velocity in ENU.
             See Issue #64 in VRX project for details
             https://bitbucket.org/osrf/vrx/issues/64 -->

<velocityTopicName>wamv/sensors/gps/gps/fix_velocity</velocityTopicName
>
      <!-- Location of origin of Gazebo Sand Island map -->
      <referenceLatitude>21.30996</referenceLatitude>
      <referenceLongitude>-157.8901</referenceLongitude>
      <referenceAltitude>0</referenceAltitude>
      <referenceHeading>90</referenceHeading>
      <offset>0.0 0.0 0.0</offset>
      <drift>0.0 0.0 0.0</drift>
      <gaussianNoise>0 0 0</gaussianNoise>
      <velocityOffset>0.0 0.0 0.0</velocityOffset>
      <velocityDrift>0.0 0.0 0.0</velocityDrift>
      <velocityGaussianNoise>0.0 0.0 0.0</velocityGaussianNoise>
    </plugin>
  </gazebo>
  <gazebo>
    <plugin filename="libgazebo_ros_p3d.so" name="p3d_plugin_p3d_wamv">
      <bodyName>base_link</bodyName>
      <frameName>map</frameName>
      <updateRate>10.0</updateRate>
      <topicName>wamv/sensors/position/p3d_wamv</topicName>
      <xyzOffset>0 0 0</xyzOffset>
      <rpyOffset>0 0 0</rpyOffset>
      <gaussianNoise>0</gaussianNoise>
    </plugin>
  </gazebo>
  <!-- Monocular Camera -->
  <link name="stereo_left_camera_link">
    <visual name="stereo_left_camera_visual">
      <origin rpy="0 0 1.57079632679" xyz="-0.033 0 0"/>
      <geometry>
        <mesh
filename="package://vrx_gazebo/models/mono_camera/mesh/mono_camera.dae"
/>
      </geometry>
    </visual>
    <collision name="stereo_left_camera_lens_collision">
      <origin rpy="0 1.57079632679 0" xyz="0.018 0 0"/>
      <geometry>
        <cylinder length="0.042" radius="0.015"/>
      </geometry>
```

```
    </collision>
    <collision name="stereo_left_camera_box_collision">
      <origin rpy="0 0 0" xyz="-0.0205 0 0"/>
      <geometry>
        <box size="0.036 0.03 0.03"/>
      </geometry>
    </collision>
    <!-- Model inertia as box with <size>0.078 0.03 0.03</size> -->
    <inertial>
      <mass value="0.3"/>
      <inertia ixx="0.0000450" ixy="0" ixz="0" iyy="0.0001746" iyz="0"
izz="0.0001746"/>
    </inertial>
  </link>
  <!-- Define a link for the optical frame.  This will match the sensor
frame and
         be oriented in the camera convention. -->
  <link name="stereo_left_camera_link_optical">
    </link>
  <!-- Sensor post -->
  <link name="stereo_left_camera_post_link">
    <visual>
      <geometry>
        <cylinder length="0.1765" radius="0.0076"/>
      </geometry>
      <material name="stereo_left_camera_post_material">
        <color rgba="0.0 0.0 0.0 1.0"/>
      </material>
    </visual>
    <collision name="stereo_left_camera_post_collision">
      <geometry>
        <cylinder length="0.1765" radius="0.0076"/>
      </geometry>
    </collision>
    <inertial>
      <mass value="0.15885"/>
      <inertia ixx="0.000414671703375" ixy="0" ixz="0"
iyy="0.000414671703375" iyz="0" izz="4.587588e-06"/>
    </inertial>
  </link>
  <!-- Sensor post arm -->
  <link name="stereo_left_camera_post_arm_link">
    <visual name="stereo_left_camera_post_arm_visual">
      <origin rpy="-1.0471975512 0 -1.57079632679" xyz="-0.038 0 -
0.003"/>
      <geometry>
        <mesh
filename="package://vrx_gazebo/models/sensor_post/mesh/sensor_post_arm.
dae"/>
      </geometry>
    </visual>
    <collision name="stereo_left_camera_post_arm_collision">
      <origin rpy="-1.20427718388 0 -1.57079632679" xyz="0 0 0"/>
      <geometry>
        <cylinder length="0.065" radius="0.008"/>
      </geometry>
    </collision>
    <inertial>
```

```
      <origin rpy="-1.20427718388 0 -1.57079632679" xyz="0 0 0"/>
      <mass value="0.1"/>
      <inertia ixx="0.00003680833" ixy="0" ixz="0" iyy="0.00003680833"
iyz="0" izz="0.00000320000"/>
    </inertial>
  </link>
  <joint name="base_to_stereo_left_camera_post_joint" type="fixed">
    <origin rpy="0 0 0" xyz="0.7 0.1 1.38475"/>
    <parent link="base_link"/>
    <child link="stereo_left_camera_post_link"/>
  </joint>
  <!-- Sensor post to sensor post arm joint -->
  <joint
name="stereo_left_camera_post_to_stereo_left_camera_post_arm_joint"
type="fixed">
    <origin rpy="0 0 0" xyz="0.03 0 0.08825"/>
    <parent link="stereo_left_camera_post_link"/>
    <child link="stereo_left_camera_post_arm_link"/>
  </joint>
  <!-- Sensor post arm to camera joint -->
  <joint name="stereo_left_camera_post_arm_to_stereo_left_camera_joint"
type="fixed">
    <origin rpy="0 0.261799387799 0" xyz="0.02 0 0.027"/>
    <parent link="stereo_left_camera_post_arm_link"/>
    <child link="stereo_left_camera_link"/>
  </joint>
  <!-- Camera to optical frame joint.  This is oriented to convert
between ENU
         and camera conventions -->
  <joint
name="stereo_left_camera_to_stereo_left_camera_link_optical_joint"
type="fixed">
    <origin rpy="-1.57079632679 0 -1.57079632679" xyz="0 0 0"/>
    <parent link="stereo_left_camera_link"/>
    <child link="stereo_left_camera_link_optical"/>
  </joint>
  <!-- Gazebo camera setup -->
  <gazebo reference="stereo_right_camera_link">
    <sensor name="stereo_right_camera_sensor" type="multicamera">
      <update_rate>30.0</update_rate>
      <camera name="stereo_right_camera_camera">
        <horizontal_fov>1.3962634</horizontal_fov>
        <image>
          <width>1280</width>
          <height>720</height>
          <format>R8G8B8</format>
        </image>
        <clip>
          <near>0.05</near>
          <far>300</far>
        </clip>
        <noise>
          <type>gaussian</type>
          <!-- Noise is sampled independently per pixel on each frame.
                 That pixel's noise value is added to each of its color
                 channels, which at that point lie in the range [0,1].
-->
          <mean>0.0</mean>
```

```
          <stddev>0.007</stddev>
        </noise>
      </camera>
      <camera name="stereo_left_camera_sensor">
        <pose>0 -0.07 0 0 0 0</pose>
        <horizontal_fov>1.3962634</horizontal_fov>
        <image>
          <width>1280</width>
          <height>720</height>
          <format>R8G8B8</format>
        </image>
        <clip>
          <near>0.05</near>
          <far>300</far>
        </clip>
        <noise>
          <type>gaussian</type>
          <!-- Noise is sampled independently per pixel on each frame.
                 That pixel's noise value is added to each of its color
                 channels, which at that point lie in the range [0,1].
-->
          <mean>0.0</mean>
          <stddev>0.007</stddev>
        </noise>
      </camera>
      <plugin name="stereo_camera_controller"
filename="libgazebo_ros_multicamera.so">
        <alwaysOn>true</alwaysOn>
        <updateRate>0.0</updateRate>
        <cameraName>wamv/sensors/cameras/stereo_camera</cameraName>
        <imageTopicName>image_raw</imageTopicName>
        <cameraInfoTopicName>camera_info</cameraInfoTopicName>
        <frameName>wamv/stereo_left_camera_link_optical</frameName>
        <!--
<rightFrameName>right_camera_optical_frame</rightFrameName>-->
        <hackBaseline>0.07</hackBaseline>
        <distortionK1>0.0</distortionK1>
        <distortionK2>0.0</distortionK2>
        <distortionK3>0.0</distortionK3>
        <distortionT1>0.0</distortionT1>
        <distortionT2>0.0</distortionT2>
      </plugin>
    </sensor>
  </gazebo>
  <!-- Set color of post -->
  <gazebo reference="stereo_left_camera_post_link">
    <material>Gazebo/Black</material>
  </gazebo>
  <!-- Monocular Camera -->
  <link name="stereo_right_camera_link">
    <visual name="stereo_right_camera_visual">
      <origin rpy="0 0 1.57079632679" xyz="-0.033 0 0"/>
      <geometry>
        <mesh
filename="package://vrx_gazebo/models/mono_camera/mesh/mono_camera.dae"
/>
      </geometry>
    </visual>
```

```
    <collision name="stereo_right_camera_lens_collision">
      <origin rpy="0 1.57079632679 0" xyz="0.018 0 0"/>
      <geometry>
        <cylinder length="0.042" radius="0.015"/>
      </geometry>
    </collision>
    <collision name="stereo_right_camera_box_collision">
      <origin rpy="0 0 0" xyz="-0.0205 0 0"/>
      <geometry>
        <box size="0.036 0.03 0.03"/>
      </geometry>
    </collision>
    <!-- Model inertia as box with <size>0.078 0.03 0.03</size> -->
    <inertial>
      <mass value="0.3"/>
      <inertia ixx="0.0000450" ixy="0" ixz="0" iyy="0.0001746" iyz="0"
izz="0.0001746"/>
    </inertial>
  </link>
  <!-- Define a link for the optical frame.  This will match the sensor
frame and
         be oriented in the camera convention. -->
  <link name="stereo_right_camera_link_optical">
    </link>
  <!-- Sensor post -->
  <link name="stereo_right_camera_post_link">
    <visual>
      <geometry>
        <cylinder length="0.1765" radius="0.0076"/>
      </geometry>
      <material name="stereo_right_camera_post_material">
        <color rgba="0.0 0.0 0.0 1.0"/>
      </material>
    </visual>
    <collision name="stereo_right_camera_post_collision">
      <geometry>
        <cylinder length="0.1765" radius="0.0076"/>
      </geometry>
    </collision>
    <inertial>
      <mass value="0.15885"/>
      <inertia ixx="0.000414671703375" ixy="0" ixz="0"
iyy="0.000414671703375" iyz="0" izz="4.587588e-06"/>
    </inertial>
  </link>
  <!-- Sensor post arm -->
  <link name="stereo_right_camera_post_arm_link">
    <visual name="stereo_right_camera_post_arm_visual">
      <origin rpy="-1.0471975512 0 -1.57079632679" xyz="-0.038 0 -
0.003"/>
      <geometry>
        <mesh
filename="package://vrx_gazebo/models/sensor_post/mesh/sensor_post_arm.
dae"/>
      </geometry>
    </visual>
    <collision name="stereo_right_camera_post_arm_collision">
      <origin rpy="-1.20427718388 0 -1.57079632679" xyz="0 0 0"/>
```

```
      <geometry>
        <cylinder length="0.065" radius="0.008"/>
      </geometry>
    </collision>
    <inertial>
      <origin rpy="-1.20427718388 0 -1.57079632679" xyz="0 0 0"/>
      <mass value="0.1"/>
      <inertia ixx="0.00003680833" ixy="0" ixz="0" iyy="0.00003680833"
iyz="0" izz="0.00000320000"/>
    </inertial>
  </link>
  <joint name="base_to_stereo_right_camera_post_joint" type="fixed">
    <origin rpy="0 0 0" xyz="0.7 -0.1 1.38475"/>
    <parent link="base_link"/>
    <child link="stereo_right_camera_post_link"/>
  </joint>
  <!-- Sensor post to sensor post arm joint -->
  <joint
name="stereo_right_camera_post_to_stereo_right_camera_post_arm_joint"
type="fixed">
    <origin rpy="0 0 0" xyz="0.03 0 0.08825"/>
    <parent link="stereo_right_camera_post_link"/>
    <child link="stereo_right_camera_post_arm_link"/>
  </joint>
  <!-- Sensor post arm to camera joint -->
  <joint
name="stereo_right_camera_post_arm_to_stereo_right_camera_joint"
type="fixed">
    <origin rpy="0 0.261799387799 0" xyz="0.02 0 0.027"/>
    <parent link="stereo_right_camera_post_arm_link"/>
    <child link="stereo_right_camera_link"/>
  </joint>
  <!-- Camera to optical frame joint.  This is oriented to convert
between ENU
         and camera conventions -->
  <joint
name="stereo_right_camera_to_stereo_right_camera_link_optical_joint"
type="fixed">
    <origin rpy="-1.57079632679 0 -1.57079632679" xyz="0 0 0"/>
    <parent link="stereo_right_camera_link"/>
    <child link="stereo_right_camera_link_optical"/>
  </joint>
  <!-- Gazebo camera setup -->
  <gazebo reference="stereo_right_camera_link">
    <sensor name="stereo_right_camera_sensor" type="camera">
      <update_rate>30.0</update_rate>
      <camera name="stereo_right_camera_camera">
        <horizontal_fov>1.3962634</horizontal_fov>
        <image>
          <width>1280</width>
          <height>720</height>
          <format>R8G8B8</format>
        </image>
        <clip>
          <near>0.05</near>
          <far>300</far>
        </clip>
        <noise>
```

```
          <type>gaussian</type>
          <!-- Noise is sampled independently per pixel on each frame.
                 That pixel's noise value is added to each of its color
                 channels, which at that point lie in the range [0,1].
-->
          <mean>0.0</mean>
          <stddev>0.007</stddev>
        </noise>
      </camera>
      <plugin filename="libgazebo_ros_camera.so"
name="camera_plugin_stereo_right_camera">
        <alwaysOn>true</alwaysOn>
        <updateRate>0.0</updateRate>

<cameraName>wamv/sensors/cameras/stereo_right_camera</cameraName>
        <imageTopicName>image_raw</imageTopicName>
        <cameraInfoTopicName>camera_info</cameraInfoTopicName>
        <frameName>wamv/stereo_right_camera_link_optical</frameName>
        <hackBaseline>0.07</hackBaseline>
        <distortionK1>0.0</distortionK1>
        <distortionK2>0.0</distortionK2>
        <distortionK3>0.0</distortionK3>
        <distortionT1>0.0</distortionT1>
        <distortionT2>0.0</distortionT2>
      </plugin>
    </sensor>
  </gazebo>
  <!-- Set color of post -->
  <gazebo reference="stereo_right_camera_post_link">
    <material>Gazebo/Black</material>
  </gazebo>
  <!-- 3d lidar sensor -->
  <link name="lidar_wamv_link">
    <visual name="lidar_wamv_visual">
      <origin rpy="0 0 0" xyz="0 0 -0.035"/>
      <geometry>
        <mesh
filename="package://vrx_gazebo/models/3d_lidar/mesh/3d_lidar.dae"/>
      </geometry>
    </visual>
    <collision name="lidar_wamv_collision">
      <geometry>
        <cylinder length="0.075" radius="0.055"/>
      </geometry>
    </collision>
    <inertial>
      <mass value="1"/>
      <inertia ixx="0.00109375" ixy="0" ixz="0" iyy="0.00109375"
iyz="0" izz="0.00125"/>
    </inertial>
  </link>
  <!-- Sensor post -->
  <link name="lidar_wamv_post_link">
    <visual>
      <geometry>
        <cylinder length="0.4535" radius="0.0076"/>
      </geometry>
      <material name="lidar_wamv_post_material">
```

```
        <color rgba="0.0 0.0 0.0 1.0"/>
      </material>
    </visual>
    <collision name="lidar_wamv_post_collision">
      <geometry>
        <cylinder length="0.4535" radius="0.0076"/>
      </geometry>
    </collision>
    <inertial>
      <mass value="0.40815"/>
      <inertia ixx="0.00700098096413" ixy="0" ixz="0"
iyy="0.00700098096413" iyz="0" izz="1.1787372e-05"/>
    </inertial>
  </link>
  <!-- Sensor post arm -->
  <link name="lidar_wamv_post_arm_link">
    <visual name="lidar_wamv_post_arm_visual">
      <origin rpy="-1.0471975512 0 -1.57079632679" xyz="-0.038 0 -
0.003"/>
      <geometry>
        <mesh
filename="package://vrx_gazebo/models/sensor_post/mesh/sensor_post_arm.
dae"/>
      </geometry>
    </visual>
    <collision name="lidar_wamv_post_arm_collision">
      <origin rpy="-1.20427718388 0 -1.57079632679" xyz="0 0 0"/>
      <geometry>
        <cylinder length="0.065" radius="0.008"/>
      </geometry>
    </collision>
    <inertial>
      <origin rpy="-1.20427718388 0 -1.57079632679" xyz="0 0 0"/>
      <mass value="0.1"/>
      <inertia ixx="0.00003680833" ixy="0" ixz="0" iyy="0.00003680833"
iyz="0" izz="0.00000320000"/>
    </inertial>
  </link>
  <joint name="base_to_lidar_wamv_post_joint" type="fixed">
    <origin rpy="0 0 0" xyz="0.63 0.0 1.52325"/>
    <parent link="base_link"/>
    <child link="lidar_wamv_post_link"/>
  </joint>
  <!-- Sensor post to sensor post arm joint -->
  <joint name="lidar_wamv_post_to_lidar_wamv_post_arm_joint"
type="fixed">
    <origin rpy="0 0 0" xyz="0.03 0 0.22675"/>
    <parent link="lidar_wamv_post_link"/>
    <child link="lidar_wamv_post_arm_link"/>
  </joint>
  <!-- Sensor post arm to lidar joint -->
  <joint name="lidar_wamv_post_arm_to_lidar_wamv_joint" type="fixed">
    <origin rpy="0 0.13962634016 0" xyz="0.04 0 0.05"/>
    <parent link="lidar_wamv_post_arm_link"/>
    <child link="lidar_wamv_link"/>
  </joint>
  <gazebo reference="lidar_wamv_link">
    <sensor name="lidar_wamv_sensor" type="gpu_ray">
```

```
      <update_rate>10</update_rate>
      <ray>
        <scan>
          <horizontal>
            <samples>18750.0</samples>
            <resolution>0.1</resolution>
            <min_angle>-3.14159265359</min_angle>
            <max_angle>3.14159265359</max_angle>
          </horizontal>
          <vertical>
            <samples>16</samples>
            <resolution>1</resolution>
            <min_angle>-0.261799387799</min_angle>
            <max_angle>0.261799387799</max_angle>
          </vertical>
        </scan>
        <range>
          <min>0.1</min>
          <max>130</max>
        </range>
        <noise>
          <type>gaussian</type>
          <mean>0.0</mean>
          <stddev>0.01</stddev>
        </noise>
      </ray>
      <plugin filename="libgazebo_ros_velodyne_gpu_laser.so"
name="lidar_wamv_plugin">
        <topicName>wamv/sensors/lidars/lidar_wamv/points</topicName>
        <frameName>wamv/lidar_wamv_link</frameName>
        <min_intensity>0</min_intensity>
      </plugin>
    </sensor>
  </gazebo>
  <!-- Set color of post -->
  <gazebo reference="lidar_wamv_post_link">
    <material>Gazebo/Black</material>
  </gazebo>
  <!--specs for 16 beam lidar are implemented by default.
          See wamv_3d_lidar for those-->
  <link name="cpu_cases_link">
    <visual name="cpu_cases_visual">
      <origin rpy="0 0 0" xyz="0.073 0 -1.53"/>
      <geometry>
        <mesh
filename="package://vrx_gazebo/models/cpu_cases/mesh/cpu_cases.dae"/>
      </geometry>
    </visual>
    <collision name="cpu_case_1_collision">
      <origin rpy="0 0 0" xyz="0.185 0 0"/>
      <geometry>
        <box size="0.595 0.83 0.47"/>
      </geometry>
    </collision>
    <collision name="cpu_case_2_collision">
      <origin rpy="0 0 0" xyz="-0.3 0 -0.092"/>
      <geometry>
        <box size="0.375 0.64 0.28"/>
```

```
      </geometry>
    </collision>
    <inertial>
      <mass value="20"/>
      <inertia ixx="1.51633333333333" ixy="0" ixz="0"
iyy="1.93633333333333" iyz="0" izz="2.71633333333333"/>
    </inertial>
  </link>
  <joint name="chasis_cpu_cases_joint" type="fixed">
    <origin rpy="0 0 0" xyz="-0.15 0 1.53"/>
    <parent link="base_link"/>
    <child link="cpu_cases_link"/>
  </joint>
</robot>
```

Appendix Q

Setup guide simulation

Setup and usage of simulation

1. Follow instalation of ROS Melodic on ros wiki homepage
2. Install osrf/vrx simulation following the tutorial in the wiki
3. Install ros stereo_image_proc
4. Generate own wamv urdf following the tutorial in the osrf/vrx wiki
5. Change the partion in my_wamv.urdf:

```
  <!-- Gazebo camera setup -->
  <gazebo reference="stereo_left_camera_link">
    <sensor name="stereo_left_camera_sensor" type="camera">
      <update_rate>30.0</update_rate>
      <camera name="stereo_left_camera_camera">
        <horizontal_fov>1.3962634</horizontal_fov>
        <image>
          <width>1280</width>
          <height>720</height>
          <format>R8G8B8</format>
        </image>
        <clip>
          <near>0.05</near>
          <far>300</far>
        </clip>
        <noise>
          <type>gaussian</type>
          <!-- Noise is sampled independently per pixel on each frame.
               That pixel's noise value is added to each of its color
               channels, which at that point lie in the range [0,1]. -->
          <mean>0.0</mean>
          <stddev>0.007</stddev>
        </noise>
      </camera>
      <plugin filename="libgazebo_ros_camera.so"
name="camera_plugin_stereo_left_camera">
        <alwaysOn>true</alwaysOn>
        <updateRate>0.0</updateRate>
        <cameraName>wamv/sensors/cameras/stereo_left_camera</cameraName>
        <imageTopicName>image_raw</imageTopicName>
        <cameraInfoTopicName>camera_info</cameraInfoTopicName>
        <frameName>wamv/stereo_left_camera_link_optical</frameName>
        <hackBaseline>0.07</hackBaseline>
        <distortionK1>0.0</distortionK1>
        <distortionK2>0.0</distortionK2>
        <distortionK3>0.0</distortionK3>
        <distortionT1>0.0</distortionT1>
        <distortionT2>0.0</distortionT2>
      </plugin>
```

```
  </sensor>
 </gazebo>
```

to:

```
 <!-- Gazebo camera setup -->
 <gazebo reference="stereo_right_camera_link">
  <sensor name="stereo_right_camera_sensor" type="multicamera">
   <update_rate>30.0</update_rate>
   <camera name="stereo_right_camera_camera">
    <horizontal_fov>1.3962634</horizontal_fov>
    <image>
     <width>1280</width>
     <height>720</height>
     <format>R8G8B8</format>
    </image>
    <clip>
     <near>0.05</near>
     <far>300</far>
    </clip>
    <noise>
     <type>gaussian</type>
     <!-- Noise is sampled independently per pixel on each frame.
          That pixel's noise value is added to each of its color
          channels, which at that point lie in the range [0,1]. -->
     <mean>0.0</mean>
     <stddev>0.007</stddev>
    </noise>
   </camera>
   <camera name="stereo_left_camera_sensor">
    <pose>0 -0.07 0 0 0 0</pose>
    <horizontal_fov>1.3962634</horizontal_fov>
    <image>
     <width>1280</width>
     <height>720</height>
     <format>R8G8B8</format>
    </image>
    <clip>
     <near>0.05</near>
     <far>300</far>
    </clip>
    <noise>
     <type>gaussian</type>
     <!-- Noise is sampled independently per pixel on each frame.
          That pixel's noise value is added to each of its color
          channels, which at that point lie in the range [0,1]. -->
```

```
        <mean>0.0</mean>
        <stddev>0.007</stddev>
      </noise>
    </camera>
    <plugin name="stereo_camera_controller" filename="libgazebo_ros_multicamera.so">
      <alwaysOn>true</alwaysOn>
      <updateRate>0.0</updateRate>
      <cameraName>wamv/sensors/cameras/stereo_camera</cameraName>
      <imageTopicName>image_raw</imageTopicName>
      <cameraInfoTopicName>camera_info</cameraInfoTopicName>
      <frameName>wamv/stereo_left_camera_link_optical</frameName>
      <!--<rightFrameName>right_camera_optical_frame</rightFrameName>-->
      <hackBaseline>0.07</hackBaseline>
      <distortionK1>0.0</distortionK1>
      <distortionK2>0.0</distortionK2>
      <distortionK3>0.0</distortionK3>
      <distortionT1>0.0</distortionT1>
      <distortionT2>0.0</distortionT2>
    </plugin>
  </sensor>
</gazebo>
```

6. Then run the mapping by:

Launching Sansisland world and spawning wamv model

```
$ roslaunch vrx_gazebo sandisland.launch urdf:=$HOME/my_wamv/my_wamv_2.urdf
```

Running the stereo_image_proc node

```
$ ROS_NAMESPACE=/wamv/sensors/cameras/stereo_camera rosrun stereo_image_proc stereo_image_proc approximate_sync:=true
```

Running rgbd mapping with lidar visualizing in rviz

```
$ roslaunch wamv_gazebo rgbd_mapping.launch rviz:=true
```

Teleop using keyboard

```
$ roslaunch vrx_gazebo usv_keydrive.launch
```

Appendix R

Meeting minutes

The date for remaining meetings can be found in Appendix C Hour List. The topics for remaining meetings can be found in the progress reports for the weeks before the meeting.

Minutes of kick-off meeting

Date: 10.01.2020
Place: Lab 160
Particcipants: Markus Holt, Yrian Hovde Øksne, Mikal Aanning, Sigurd Odden, Ottar L. Osen, Girts Strazdins, Anete Vagale Elias Hasle
Abscent: Robin T. Bye

Kick-off meeting for bachelor thesis conducted at the Electronics lab L160 at NTNU campus Ålesund the 10.01.2020. Ottar L. Osen began the meeting with presenting the origin of the project. Ottar have been working on establishing a competition between colleges and universities in Norway, based on the international competition RoboBoat arranged by RoboNation, where a team of students design autonomous surface vehicles which are tested in manoeuvrability, navigation and design. The rules of the competition was not completed, but the group were told to base development of a solution on the RoboBoat competition. The competition is

The scope of the project will be to design and build a hull with necessary sensors, propulsion and power to solve each task of the competition and develop necessary software to solve each task. Ottar also mentioned that the group should partner with a group consisting of graduate students from POD and ship design, which are going to compete in the RoboBoat competition. The group was told to focus on further development of the pre-project report in the following period.

Minutes of progress meeting

Date: 28.01.2020
Place: Rundskue
Particcipants: Markus Holt, Yrian Hovde Øksne, Mikal Aanning, Sigurd Odden, Ottar L. Osen, Robin T. Bye, Girts Strazdins and Paul Steffen Kleppe
Abscent: none

The meeting took place in the meeting room Rundskue at NTNU campus Ålesund the 28.01.2020.

The meeting was started with the group presenting their current version of the Gantt diagram. Feedback from the supervisors where to split up the processes and create more specific tasks in the diagram. The groups MVP was also presented. The group got pleasant feedback about the pre-project report and about the software design. The group was told to put more focus on the hardware and development of the boat. It should be created a weight budget and order necessary components in the near future.

The group wanted counselling about autonomous navigation algorithms. The feedback was to contact PhD. candidate Anete Vagale as she recently had finished a thesis about all the popular path planning algorithms in autonomous vessels and was currently working with collision avoidance.

Minutes of progress meeting

Date: 11.02.2020
Place: Rundskue
Particcipants: Markus Holt, Yrian Hovde Øksne, Mikal Aanning, Sigurd Odden, Ottar L. Osen, Robin T. Bye, Girts Strazdins
Abscent: none

The meeting took place in the meeting room Rundskue at NTNU campus Ålesund the 11.02.2020.

The meeting was started with the group presenting their current version of the Gantt diagram. Feedback from the supervisors where to split up the processes and create more specific tasks in the diagram. The groups MVP was also presented. The group got pleasant feedback about the pre-project report and about the software design. The group was told to put more focus on the hardware and development of the boat. It should be created a weight budget and necessary components should be ordered in the near future.

The group wanted counselling about autonomous navigation algorithms. The feedback was to contact PhD. candidate Anete Vagale as she recently had finished a thesis about all the path planning algorithms used in autonomous vessels and was currently working with collision avoidance.

Minutes of progress meeting

Date: 05.05.2020
Place: Zoom
Particcipants: Markus Holt, Yrian Hovde Øksne, Mikal Aanning, Sigurd Odden, Ottar L. Osen, Robin T. Bye, Girts Strazdins
Abscent: none

The meeting took place at Zoom the 05.05.2020. Feedback about progress in simulation and report writing. A draft of the report should be sent to supervisors in order to get feedback.

Appendix S

Progress Reports

IE303612 Bachelor thesis	Project: ASV	Number of meetings this period 1): 3	Company: NTNU Ålesund	Page: 1 of 2
Progress report	Period/week(s): 3	Number of hours this period. (from log): Approx. 200 in total	Members: Sigurd Odden, Mikal Aanning, Markus Holt and Yrian Hovde Øksne	Date: 24.01.2020

Main goal/purpose for this periods work

- Develop a progress plan for the entire project
- Get an overview of the projects scope
- Acquire knowledge of existing solutions and applicable technologies
- Establish partnership for development of hull

Planned activities this period

- Research existing solutions (ASV/USV)
- Meeting with potential partners
- Complete pre-project
- Create Gantt
- Develop MVP and deployment diagram
- Start filling Scrum backlog

Actual conducted activities this period

- The pre project was completed
- Researched existing solutions (ASV/USV):

Looked at ROS and its capabilities, considering we need object- and colour detection, odometry, navigation, room mapping, remote control and motor control.
Looked at hardware such as: microcontroller, lidar, camera, thrusters, batteries, battery management and IMU.
Researched software testing and case testing.

- Met with potential partners

A partnership was established. We can trade our software solution for a copy their hull.

- Completed pre-project
- Created Gantt, developed drafts of MVP and deployment diagram and started filling Scrum backlog

We are aware that the current state of these progression tools will change during the project.

Description of/ justification for potential deviation between planned and real activities

- We did not expect to complete the pre project and description of MVP that fast. Thus, we started filling the Scrum backlog and development of deployment diagram.

Description of/ justification for changes that is desired in the projects content or in the further plan of action – or progress report

- As we are in the planning phase, no major changes are necessary. We have created a deviation management plan; therefore, we are prepared if any changes are needed.

Main experience from this period

- We have a better understanding of the scope of the project.
- We are experiencing some issues with breaking down a user story regarding autonomous navigation into subtasks. The user story is called: The vehicle must be able to determine its own position in its frame of reference and then plan the most effective path to some goal location and execute on it. If the occasion arises, the vehicle must avoid collisions.

We had introductory meeting with supervisors, meeting with partners, regular stand-up meetings and progress meeting.

IE303612 Bachelor thesis	Project: ASV	Number of meetings this period 1): 3	Company: NTNU Ålesund	Page: 2 of 2
Progress report	Period/week(s): 3	Number of hours this period. (from log): Approx. 200 in total	Members: Sigurd Odden, Mikal Aanning, Markus Holt and Yrian Hovde Øksne	Date: 24.01.2020

Main purpose/focus next period

- A more rich and complete backlog
- Get a better understanding of how autonomous navigation works and how to implement it.
- Learn ROS-, Python-, OpenCV- and Linux basics.

Planned activities next period

- Complete the ROS short course and OpenCV course
- Research autonomous navigation
- Update backlog

Other

Wish/need for counseling

- Autonomous navigation
- Hardware components
- Software architecture

Approval/signature group leader	Signature other group participants
N.N.	N.N.

We had introductory meeting with supervisors, meeting with partners, regular stand-up meetings and progress meeting.

IE303612 Bachelor thesis	Project: ASV	Number of meetings this period 1): 2	Company: NTNU Ålesund	Page: 1 of 1
Progress report	Period/week(s): 1	Number of hours this period. (from log): Approx. 80	Members: Sigurd Odden, Mikal Aanning, Markus Holt and Yrian Hovde Øksne	Date: 31.01.2020

Main goal/purpose for this periods work

- A more rich and complete backlog
- Get a better understanding of how autonomous navigation works and how to implement it
- Learn ROS-, Python-, OpenCV- and Linux basics

Planned activities this period

- Complete the ROS short course and OpenCV course
- Research autonomous navigation
- Update backlog

Actual conducted activities this period

- Attended meeting with supervisors and partners

Got feedback on how to split backlog into smaller subtasks. We should not fit the syllabus into the project, we should use the syllabus to solve the project. We got advised to set up a weight budget. We got insight in which path planning algorithm to use.

- Started short courses about ROS and OpenCV
- Restructured MVP and backlog

Description of/ justification for potential deviation between planned and real activities

- We did not finish restructuring the new backlog due to lack insight. We plan to complete it after completing the short courses.

Description of/ justification for changes that is desired in the projects content or in the further plan of action – or progress report

- We did not finish the courses because of lectures and project in other subjects.

Main experience from this period

- Insight gained from the courses

Basic insight of Linux (Ubuntu), OpenCV, ROS and Python

Main purpose/focus next period

- A more rich and complete backlog
- Get a better understanding of how autonomous navigation works and how to implement it.
- Learn ROS-, Python-, OpenCV- and Linux basics.
- Get more insight in hardware selection and document reasoning

Planned activities next period

- Complete the ROS short course and OpenCV course
- Research autonomous navigation
- Update backlog
- Create a weight budget
- Select hardware

Other

Wish/need for counseling

- Autonomous navigation
- Hardware components

Approval/signature group leader	Signature other group participants
N.N.	N.N.

IE303612 Bachelor thesis	Project: ASV	Number of meetings this period 1): 1.	Company: NTNU Ålesund	Page: 1 of 2
Progress report	Period/week(s): 1	Number of hours this period. (from log): Approx. 100	Members: Sigurd Odden, Mikal Aanning, Markus Holt and Yrian Hovde Øksne	Date: 07.02.2020

Main goal/purpose for this periods work

- A more rich and complete backlog
- Get a better understanding of how autonomous navigation works and how to implement it
- Learn ROS-, Python-, OpenCV- and Linux basics
- Get more insight in hardware selection and document reasoning

Planned activities this period

- Complete the ROS short course and OpenCV course
- Research autonomous navigation
- Update backlog
- Create a weight budget
- Select hardware

Actual conducted activities this period

- We are well into both ROS- and OpenCV courses
- Researched autonomous navigation through Python with ROS and its addon packages like: gmapping, hector_slam, navigation etc
- We updated MVP, split up user stories into smaller more cohesive parts. We also updated some user stories in the backlog
- We have started creating weight- and cost budget for components
- We have decided on type of sensors

Description of/ justification for potential deviation between planned and real activities

- Due to technical issues in the form of internet loss, some delays resulted in not meeting our intended progression goals in ROS courses

Description of/ justification for changes that is desired in the projects content or in the further plan of action – or progress report

- We and our partners decided on using the T-200 thrusters instead of outboard motor
- Gmapping seems to be a better choice than hector-slam for naval navigation

Main experience from this period

- Insight gained from the courses

Basic insight of Linux (Ubuntu), OpenCV, ROS and Python

- Insight gained on how to improve our backlog
- Capabilities and possibilities of ROS and OpenCV

IE303612 Bachelor thesis	Project: ASV	Number of meetings this period 1): 1.	Company: NTNU Ålesund	Page: 2 of 2
Progress report	Period/week(s): 1	Number of hours this period. (from log): Approx. 100	Members: Sigurd Odden, Mikal Aanning, Markus Holt and Yrian Hovde Øksne	Date: 07.02.2020

Main purpose/focus next period

- A more rich and complete backlog
- Get a better understanding of autonomous navigation algorithms
- Learn ROS and OpenCV in depth
- Get more insight in choice of power system and document reasoning
- Get started on the Bachelor report

Planned activities next period

- Complete the ROS- and OpenCV course
- Order parts
- Research autonomous navigation
- Update backlog
- Create a weight- and cost budget
- Select hardware
- Get started on the Bachelor report

Other

Wish/need for counseling

- Autonomous navigation design

Approval/signature group leader	Signature other group participants
N.N.	N.N.

IE303612 Bachelorthesis	Project ASV	Number of meeting this period 1). 2 planned	Firma - Oppdragsgiver Norges Tekniske-Naturvitenskaplige Universitet	Side 1 av 2
Progress report	Period/week(s) 2	Number of hours this period. (from log) Approx. 100	Prosjektgruppe (navn)	Dato 14.02.20

Main goal/purpose for this periods work

- Expand scrum backlog and divide user stories into smaller pieces
- Get a more complete understanding of autonomous navigation
- Learn ROS-, Python-, Linux- and OpenCV methods and principles
- Decide on MVP (Hopefully - requires competition rules to be completed.)
- Hardware selection and ordering

Planned activities this period

- Complete ROS- and OpenCV courses
- Order parts
- Research autonomous navigation
- Update backlog
- Create weight- and cost budget
- Select hardware
- Get started on bachelors report

Actually conducted activites this period

- Completed ROS- and OpenCV courses
- Decided on some hardware parts
- Researched autonomous navigation
- Updated some of the backlog
- Began budgeting of hardware weight and cost
- Arranged meeting with purchasing manager
- Began writing bachelors report

Description of/ justification for potential deviation between planned and real activities

- Due to lectures in other courses we did not meet the satisfactory progression goals. However, the difference in conducted- and planned activities is minimal and can be added onto the next period with ease. This will not be a problem.

Description of/ justification for changes that is desired in the projects content or in the further plan of action – or progress report

- Due to the fact that this period was mostly used for getting a deeper understanding of the components the project solution requires, we did not change anything about the MVP.

Main experience from this period

- Insight gained from courses.
- Insight gained on how to improve backlog and MVP
- Greater understanding of capabilities and utilities of ROS, Linux and OpenCV
- Greater understanding of practical report writing

Main purpose/focus next period

- Get a more rounded overview of the solutions scope
- Get greater understanding of practical report writing

Planned activities next period

- Continue writing bachelors report
- Complete all diagrams – software (UML) and hardware (Electrical schematics)
- Order most of the hardware parts, especially the ones with the longest arrival time.
- Continuous research
- Update backlog

1) Noter her kort tilbakemelding om antall møter – fordelt på typer (interne, styringsgruppe, møte med veileder) - i denne rapportperioden
Partnermøte og veiledermøte

IE303612 Bachelorthesis	Project ASV	Number of meeting this period 1). 2 planned	Firma - Oppdragsgiver Norges Tekniske-Naturvitenskaplige Universitet	Side 2 av 2
Progress report	Period/week(s) 2	Number of hours this period. (from log) Approx. 100	Prosjektgruppe (navn)	Dato 14.02.20

- Get ready for development	
Other	
Wish/need for counceling - Nothing in particular	
Approval/signature group leader N.N.	Signature other group participants N.N.

1) Noter her kort tilbakemelding om antall møter – fordelt på typer (interne, styringsgruppe, møte med veileder) - i denne rapportperioden
Partnermøte og veiledermøte

IE303612 Bachelor thesis	Project: ASV	Number of meetings this period 1): 2	Company: NTNU Ålesund	Page: 1 of 2
Progress report	Period/week(s): 1	Number of hours this period. (from log): Approx. 100	Members: Sigurd Odden, Mikal Aanning, Markus Holt and Yrian Hovde Øksne	Date: 21.02.2020

Main goal/purpose for this periods work

- Get a more rounded overview of the solutions scope
- Get greater understanding of practical report writing

Planned activities this period

- Continue writing bachelors report
- Complete all diagrams
- Software (UML) and hardware (Electrical schematics)
- Order most of the hardware parts, especially the ones with the longest arrival time
- Continuous research
- Update backlog
- Get ready for development

Actual conducted activities this period

- We have updated the MVP and backlog
- We have begun designing electrical circuits
- We have begun designing AI model
- We have begun designing UML
- Microcontroller, thrusters and ESC are ordered

Description of/ justification for potential deviation between planned and real activities

- No deviation this week

Description of/ justification for changes that is desired in the projects content or in the further plan of action – or progress report

- No changes

Main experience from this period

- We gained insight of pattern planning algorithms
- Design of power system
- We gained insight in how to write a good report

Main purpose/focus next period

- Development

IE303612 Bachelor thesis	Project: ASV	Number of meetings this period 1): 2	Company: NTNU Ålesund	Page: 2 of 2
Progress report	Period/week(s): 1	Number of hours this period. (from log): Approx. 100	Members: Sigurd Odden, Mikal Aanning, Markus Holt and Yrian Hovde Øksne	Date: 21.02.2020

Planned activities next period	
- Begin development of AI, sensor fusion and ROS implementation	
Other	
Wish/need for counseling	
Approval/signature group leader N.N.	Signature other group participants N.N.

IE303612 Bachelor thesis	Project: ASV	Number of meetings this period 1): 1	Company: NTNU Ålesund	Page: 1 of 2
Progress report	Period/week(s): 1	Number of hours this period. (from log): Approx. 100	Members: Sigurd Odden, Mikal Aanning, Markus Holt and Yrian Hovde Øksne	Date: 16.03.2020

Main goal/purpose for this periods work

- Continue development
- Finish remote control service
- Finish AI model

Planned activities this period

- Continue writing bachelors report
- Finish sprint
- Order the last set of hardware components.
- Continuous research
- Update backlog
- Update Gantt

Actual conducted activities this period

- Updated Gantt diagram with milestones in software development and more details generally
- Enabled remote control with PS4 controller
- Enabled communication between Jetson and Pixhawk
- Finished YOLO-model with example dataset
- Updated backlog with more relevant subtasks

Description of/ justification for potential deviation between planned and real activities

- It was not conducted as much work as planned on Thursday and Friday because of corona.

Description of/ justification for changes that is desired in the projects content or in the further plan of action – or progress report

- A meeting was held on Monday 16.03 in order to plan the further work during this pandemic. We are still going to have workhours from 09-16, Monday to Friday and we will have a daily stand-up meeting at 0900 in order to have good communication within the group, and we will have a status meeting each Friday.

Main experience from this period

- It is impossible to evaluate every risk that can occur during a project.

IE303612 Bachelor thesis	Project: ASV	Number of meetings this period 1): 1	Company: NTNU Ålesund	Page: 2 of 2
Progress report	Period/week(s): 1	Number of hours this period. (from log): Approx. 100	Members: Sigurd Odden, Mikal Aanning, Markus Holt and Yrian Hovde Øksne	Date: 16.03.2020

Main purpose/focus next period - Begin working on navigation - Feed sensor data from Pixhawk to ROS	
Planned activities next period - Develop nodes to feed imu data, gnss and compass to ROS - Mapping of obstacles in coordinate frame	
Other	
Wish/need for counseling	
Approval/signature group leader N.N.	Signature other group participants N.N.

IE303612 Bachelor thesis	Project: ASV	Number of meetings this period 1): 1	Company: NTNU Ålesund	Page: 1 of 2
Progress report	Period/week(s): 1	Number of hours this period. (from log): Approx. 100	Members: Sigurd Odden, Mikal Aanning, Markus Holt and Yrian Hovde Øksne	Date: 23.03.2020

Main goal/purpose for this periods work

- Begin working on navigation
- Feed sensor data from Pixhawk to ROS
- Begin working on gmapping and localization
- Optical obstacle mapping

Planned activities this period

- Develop nodes to feed imu data, gnss and compass to ROS
- Mapping of obstacles in coordinate frame
- Create map with gmapping
- Locate the robot in the map
- Map obstacles using camera

Actual conducted activities this period

- Started to develop a model in order to represent, test and tune the vehicle in Gazebo
- Implemented mapping and localization for test-robot in Gazebo simulation using ROS and Navigation stack

Description of/ justification for potential deviation between planned and real activities

- Ros nodes for sensor data feed has not yet been created as it was prioritised to create a simulation of the vehicle
- Had to simulate the map with ROS Development Studio, since the ZED camera has not been ordered yet.
- Did not map obstacles using camera due to lack of knowledge, however, a fair amount of research was conducted. This will most likely be done within the next week

Description of/ justification for changes that is desired in the projects content or in the further plan of action – or progress report

- Creating a simulation was prioritised because it will get difficult to develop a physical model of the boat in the nearest future. We would still have a programmable and representable boat if the competition were to be cancelled, or the creation of the boat would turn up to be impossible to complete.
-

Main experience from this period

IE303612 Bachelor thesis	Project: ASV	Number of meetings this period 1): 1	Company: NTNU Ålesund	Page: 2 of 2
Progress report	Period/week(s): 1	Number of hours this period. (from log): Approx. 100	Members: Sigurd Odden, Mikal Aanning, Markus Holt and Yrian Hovde Øksne	Date: 23.03.2020

Main purpose/focus next period - Simulation - Path planning and autonomous navigation - Optical obstacle mapping	
Planned activities next period - Complete modelling of the boat. - Implement the boat in Gazebo. - Use mapping and localization to implement path planning - Implement autonomous navigation using path execution on planned path	
Other	
Wish/need for counseling	
Approval/signature group leader N.N.	Signature other group participants N.N.

IE303612 Bachelor thesis	Project: ASV	Number of meetings this period 1): 1	Company: NTNU Ålesund	Page: 1 of 2
Progress report	Period/week(s): 1	Number of hours this period. (from log): Approx. 120	Members: Sigurd Odden, Mikal Aanning, Markus Holt and Yrian Hovde Øksne	Date: 30.03.2020

Main goal/purpose for this periods work

- Have objects mapped in a Gazebo environment, using the coordinates that have been sent from a python script
- Have a Gazebo environment with water, waves, wind and current
- Autonomous navigation and orientation

Planned activities this period

- Mapping objects in Gazebo using ROS
- Make an environment in Gazebo for simulation.
- Create a map with camera in ROS Development Studio
- Implement localizer using map as input
- Implement move_base for navigation
- Implement Python Script for navigating a planned path
- Complete modelling of the boat
- Implement the boat in Gazebo

Actual conducted activities this period

- Mapped a room with camera in ROS development Studio
- Implemented simulation from the RobotX competition, containing wind, drag, water current and the WAM-V boat for testing. Looked at how to calculate thrust vector, use sensors and driving the wam-v
- Implemented localizer for turtlebot in test simulation
- Implemented move_base with goal-oriented control for turtlebot in test simulation
- Started implementing python script for navigating a planned path

Description of/ justification for potential deviation between planned and real activities

- Focused on making an environment with waves, wind and premade models of objects(buoys etc.) instead of focusing on implementation of objects in an empty workspace.
- Created a map with camera in ROS Development Studio, but the robot had some problems localizing in RTAB-map
- Due to sickness, one group member took the Friday of and did not fully implement the python script for navigating a planned path
- Simulation is ready to development and testing but will need modifications in order to fit to our specifications. Have not finished implemented a model of our boat, as the time was spent to learn how to use the wam-v and how to create our own boat in the simulation.

Description of/ justification for changes that is desired in the projects content or in the further plan of action – or progress report

- Creating an environment was prioritised because it will get difficult to implement objects in an empty workstation, compared to an environment package with models already implemented.
- The scope of out project change drastically due to the COVID-19 crisis. The new scope will be explained at length in our bachelor’s report

Main experience from this period

IE303612 Bachelor thesis	Project: ASV	Number of meetings this period 1): 1	Company: NTNU Ålesund	Page: 2 of 2
Progress report	Period/week(s): 1	Number of hours this period. (from log): Approx. 120	Members: Sigurd Odden, Mikal Aanning, Markus Holt and Yrian Hovde Øksne	Date: 30.03.2020

Progression was made during this "home office" period that we are currently stated in. We look to make more progress in the coming weeks with a common goal of writing a good bachelors report by spending more time with the report than before.

Main purpose/focus next period

- Establish connection between ZED-camera and python backend
- Data-set collection using the ZED-camera
- Autonomous navigation for turtlebot
- Transfer implemented solutions from turtlebot to our model
- Digital twin

Planned activities next period

- Research regarding ZED software
- Implementation of ZED in the python code revolving OpenCV
- Data-set collection of buoys for training and testing
- Labelling the dataset
- Further develop our own boat and implement it in the simulation
- (hopefully) Implement solutions from test simulation using turtlebot to simulation with our boat
- Further develop navigation

Other

Wish/need for counseling

Approval/signature group leader	Signature other group participants
N.N.	N.N.

IE303612 Bachelor thesis	Project: ASV	Number of meetings this period 1): 1	Company: NTNU Ålesund	Page: 1 of 2
Progress report	Period/week(s): 2	Number of hours this period. (from log): Approx. 100	Members: Sigurd Odden, Mikal Aanning, Markus Holt and Yrian Hovde Øksne	Date: 14.04.2020

Main goal/purpose for this periods work

- Digital twin
- Fuse RTAB-mapping and autonomous navigation system
- Prepare for AI-training
- Autonomous navigation with python

Planned activities this period

- Further develop our own boat and implement it in the simulation
- Write in theory and method chapter in the report
- Building a course in Gazebo
- Gather and label data set of buoys for AI training
- Write continuously on the bachelor's report

Actual conducted activities this period

- Finished writing about reference frames and transformations in report.
- Further developed the model of our boat, began working with thruster placement.
- began writing about RTAB-map in the report
- Fused RTAB-mapping and autonomous navigation system in ROS Development Studio
- Implemented the four courses in gazebo simulation
- Gathered and labelled data set for AI training
- Made python script for controlling a test robot with the navigation stack autonomously

Description of/ justification for potential deviation between planned and real activities

- It was focused more on writing in the report than the simulation during the Easter week

Description of/ justification for changes that is desired in the projects content or in the further plan of action – or progress report

-

Main experience from this period

Main purpose/focus next period

- Simulation
- Report
- Buoy detection

IE303612 Bachelor thesis	Project: ASV	Number of meetings this period 1): 1	Company: NTNU Ålesund	Page: 2 of 2
Progress report	Period/week(s): 2	Number of hours this period. (from log): Approx. 100	Members: Sigurd Odden, Mikal Aanning, Markus Holt and Yrian Hovde Øksne	Date: 14.04.2020

Planned activities next period - Complete digital twin of boat - Report writing - Complete training the AI model - Implement and transfer scripts from test robot to the original boat robot in gazebo	
Other	
Wish/need for counseling	
Approval/signature group leader N.N.	Signature other group participants N.N.

IE303612 Bachelor thesis	Project: ASV	Number of meetings this period 1): 1	Company: NTNU Ålesund	Page: 1 of 1
Progress report	Period/week(s): 2	Number of hours this period. (from log): Approx. 100	Members: Sigurd Odden, Mikal Aanning, Markus Holt and Yrian Hovde Øksne	Date: 14.04.2020

Main goal/purpose for this periods work

- Train buoy detection
- Get simulation up and going on all workstations
- Complete digital twin of boat
- Implement and transfer scripts from test robot to simulation robot

Planned activities this period

- Report writing
- Training the buoy detection algorithm
- Implement mean Average Precision
- Install VRX simulation on all workstations(Jetson TX2)
- Complete digital twin of boat

Actual conducted activities this period

- Finished training the buoy detection algorithm
- Implemented MAP for the buoy detection
- Report writing
- Installed nvidia docker, but cannot install VRX simulation
- Created our own version of the Wam-v usv from the RobotX simulation with X/holonomic thruster configuration.

Description of/ justification for potential deviation between planned and real activities

- Could not access jetsons GPU with nvidia docker, therefor, work on simulation was postponed. We will not be able to have a simulation for each person in the group, therefor, we shall work over a screensharing program with the group member who has a powerful enough computer to run the simulation.

Description of/ justification for changes that is desired in the projects content or in the further plan of action – or progress report

-

Main experience from this period

Main purpose/focus next period

- Report
- Test run in simulation

Planned activities next period

- Report writing
- Merge and adjust Wam-v with navigation and path-planning scripts.

Other

Wish/need for counseling

Approval/signature group leader	Signature other group participants
N.N.	N.N.

IE303612 Bachelor thesis	Project: ASV	Number of meetings this period 1): 2	Company: NTNU Ålesund	Page: 1 of 1
Progress report	Period/week(s): 2	Number of hours this period. (from log): Approx. 100	Members: Sigurd Odden, Mikal Aanning, Markus Holt and Yrian Hovde Øksne	Date: 04.05.2020

Main goal/purpose for this periods work

- Simulation
- Report writing
- Test run in simulation

Planned activities this period

- Merge wamv with navigation and path planning scripts.
- Report writing

Actual conducted activities this period

- Finished writing about project organisation and transition from scrum to gantt diagram.
- Finished writing the chapters revolving around thruster configuration and hull design.
- Finished writing the chapters revolving around computer vision, data-collection and annotation.
- Implemented the rmAP score in result chapter.
- Began writing on the discussion part of the object detection
- Began implementing rtabmap for mapping and navigation on Wam-v

Description of/ justification for potential deviation between planned and real activities

- Navigation and path-planning scripts have not yet been merged on Wam-v because of difficulties with feeding sensor data to Rtabmap. Different solutions in both Gmapping and Rtabmap have been tried without luck. The problem has been located in a package that should rectify the stereo images, but the node does not give any output.

Description of/ justification for changes that is desired in the projects content or in the further plan of action – or progress report

-

Main experience from this period

Main purpose/focus next period

- Get close to a finished product and report

Planned activities next period

- Report writing
- Test and document run in simulation

Other

Wish/need for counseling

Approval/signature group leader	Signature other group participants
N.N.	N.N.