NTNU
Norwegian University of
Science and Technology

Improved user-experience for control of ROVs

Torkil Eide Solstad

Marine Technology
Submission date: June 2016
Supervisor: Roger Skjetne, IMT
Co-supervisor: Mauro Candeloro, IMT

Norwegian University of Science and Technology
Department of Marine Technology

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

MSC THESIS DESCRIPTION SHEET

Name of the candidate: Torkil Eide Solstad
Field of study: Marine control engineering
Thesis title (Norwegian): Nye HMI løsninger for å forbedre brukeropplevelse ved styring av ROV
Thesis title (English): New HMI solutions to improve user experience for control of ROVs

Background
There has been an increased interest in research and development of technical solutions for underwater vehicles. We now possess the necessary knowledge and technology to perform complex underwater operations with high precision, such as seabed mapping, online underwater monitoring, subsea installations, and maintenance on pipes. There are many types of Unmanned Underwater Vehicles (UUVs) on the commercial market. Remotely Operated Vehicles (ROVs) and Autonomous Underwater Vehicles (AUVs) are most common. In most cases, underwater vehicles are expensive and not affordable to private consumers. Lately, a focus has been to develop low-cost solutions. In a student project by the candidate together with Stian S. Sandøy and Andreas. V. Henriksen, a fully actuated mini-ROV, named *ROV uDrone*, was assembled based on a "BlueROV Kit" from BlueRobotics. The control system hardware and software were designed and implemented based on the Robot Operating System (ROS), and successfully tested in MC-Lab. The candidate was in this project responsible for development of a touchscreen-based HMI solution based on a Windows Surface Pro tablet PC.

In this thesis the objective is new efficient human-machine interface (HMI) solutions for a low-cost ROV based on the Windows Surface Pro (WSP) and an Oculus Rift (OR) Head-Mounted Display (HMD). The HMI shall be interfaced with the ROS-based control system for ROV uDrone. The WSP and OR should be able to present the graphical information from the onboard camera(s) in an efficient manner, as well as control the ROV in different control modes. The main focus of the thesis is design, implementation, and testing of HMD- and touchscreen-based features for efficient user control of ROV positioning.

Work description
1. Perform a background and literature review to provide information and relevant references on:
   - The ROV uDrone.
   - Dynamic positioning of ROVs at NTNU AUR-Lab.
   - Low-cost ROVs and their HMI solutions.
   - Use of augmented reality information on HMI screens (Head Mounted Displays, e.g. fighter pilots helmets)
   - Relevant Oculus Rift applications.
   - Real-time video streaming technology in internet video conference programs – how is the video lag handled?

   Write a list with abbreviations and definitions of terms, explaining relevant concepts related to the background study and project tasks.

2. Implement an augmented reality platform for the WSP and the OR, with available videostream and monitoring information, considering:
   - Interface camera stream from ROV to WSP and OR with a minimum of lag in the stream.
   - How to specify camera configuration w.r.t. FOV, panning option, etc.?
   - What information should be shown on the WSP screen and on the OR screen?
   - Setting up the communication between the ROV control software and the WSP and OR.
   - Interfacing of sensor signals from the OR.
   - Converting touchscreen gestures and HMD head motions into a camera panning function (should be generic to allow both software and mechanic panning). This includes how to first rotate the camera reference frame in the body-frame, and next the body-frame in the inertial frame.

3. Suggest a set of touchscreen gestures and a set of head motions to generate relevant commands to the ROV. Different alternatives can be proposed, and their pros and cons should be discussed.

4. Develop guidance algorithms to convert HMD head-motions or touchscreen gestures into "Direct motion control" commands. This can be input to the existing thrust allocation in uDrone.

5. Develop guidance algorithms to convert HMD head-motions or touchscreen gestures into AutoHeading references.
   - This shall be input to the AutoHeading controller for uDrone.
   - It shall be possible to combine AutoHeading with direct motion control in surge/sway.
   - Develop a method for combining camera panning with ROV "dynamic panning".

6. Develop guidance algorithms to convert HMD head-motions or touchscreen gestures into AutoPos references for an ROV 3DOF DP controller.
   - This shall also include an HMI function on the WSP for setting desired depth to the AutoDepth controller.

7. Conduct testing in lab and discuss resulting experiences. What works well and less good?

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

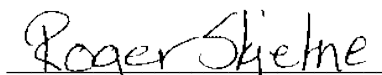
The candidate shall present personal contribution to the resolution of problems within the scope of work. Theories and conclusions should be based on mathematical derivations and logic reasoning identifying the various steps in the deduction.

The report shall be organized in a logical structure to give a clear exposition of background, results, assessments, and conclusions. The text should be brief and to the point, with a clear language. The report shall be written in English (preferably US) and contain the following elements: Title page, abstract, acknowledgements, thesis specification, list of symbols and acronyms, table of contents, introduction and background, problem formulations, scope, and delimitations, main body with derivations/developments and results, conclusions with recommendations for further work, references, and optional appendices. All figures, tables, and equations shall be numerated. The original contribution of the candidate and material taken from other sources shall be clearly identified. Work from other sources shall be properly acknowledged using quotations and a Harvard citation style (e.g. *natbib* Latex package). The work is expected to be conducted in an honest and ethical manner, without any sort of plagiarism and misconduct. Such practice is taken very seriously by the university and will have consequences. NTNU can use the results freely in research and teaching by proper referencing, unless otherwise agreed upon.

The thesis shall be submitted with a printed and electronic copy to the main supervisor, each copy signed by the candidate. The final revised version of this thesis description must be included. The report must be submitted according to NTNU procedures. Computer code, pictures, videos, data series, and a PDF version of the report shall be included electronically with all submitted versions.

Start date: 15 January, 2016 **Due date:** As specified by the administration.
Supervisor: Roger Skjetne
Co-advisor(s): Mauro Candeloro

Trondheim, 13.03.2016



Roger Skjetne (Supervisor)

Abstract

To make small size underwater Remotely Operated Vehicles(ROVs) more accessible for normal users efficient Human-Machine Interface(HMI) solutions are needed. In this theses two solutions are presented. One uses a Windows Surface Pro(WSP) touch pad to control the ROV using a touch based interface. The other solution uses a Oculus Rift(OR) Head Mounted Display(HMD) to control the ROV using head gestures.

The systems are made for a small ROV called uDrone. This is a drone with 6 thruster build by students the fall of 2015. In front of the ROV it has a 180 degree fisheye camera.

The solution presented for the HMD uses head movements and rotations to control the ROV. This can control the ROV both in direct motion control and in automatic control. In addition it uses the 180 degree fisheye camera to create a software panning feature. This way the user can rotate its head in yaw and pitch to see different parts of the video stream. Some form of augmented reality is also introduced by showing the user the position and orientation of the head and ROV in the HMD.

The solution presented for the touch pad uses touch gestures to control the ROV. The video stream is presented on the screen, with depth, heading and position information shown on top. To control the ROV in direct motion control two joysticks on the touch pad are used. In auto depth/heading/position control new desired positions are send to the controllers based on the moving of indicators done by the user. These indicators are shown together with the actual position making it easy to see both the desired and the actual position at one time.

Both solutions are tested the ROV uDrone in the Mc lab at NTNU Tyholt. The HMD control worked fine. The connection with a 180 degree camera and the software panning connected with the yaw control worked very well. The control of surge and sway using head gestures did not feel natural, and users preferred using game controllers.

The touch pad interface gave a better user experience in auto control modes, compared to writing numerical values to the controllers. In the direct motion control the use of joysticks on the screen did not work that good, and should be improved.

Sammendrag

For å gjøre små Fjernstyrte Undervannsfarkoster(ROV) mer tilgjengelig for vanlige brukere er nye effektive løsninger for menneske–maskin-interaksjon nødvendig. I denne oppgaven blir to løsninger presentert. En bruker en Windows Surface Pro(WSP) berøringsskjerm for å styre ROV ved hjelp av et berøring-basert grensesnitt. Den andre løsningen bruker en Oculus Rift(OR) hode montert skjerm(HMD) for å styre ROVen ved hjelp av hodebevegelser.

Systemene er laget for en liten ROV som heter uDrone. Dette er en drone med seks propeller bygget av studenter høsten 2015. I front av ROVen er det et 180 graders *fisheye* kamera.

Løsningen presentert for HMDen bruker hodebevegelser for å kontrollere ROV. På denne måten kan ROVen kontrollers både i direkte bevegelseskontroll og i automatisk kontroll. I tillegg bruker den 180 graders *fisheye* kameraet for å lage en panorering funksjon i programvaren. På denne måten kan brukeren rotere hodet i gir og stamp for å se forskjellige deler av videoen. Noe elementer av utvidet virkelighet er også innført ved å vise brukeren posisjon og orientering av hodet og ROVen i HMDen.

Løsningen presentert for berøringsskjermen bruker berørings-bevegelser for å styre ROVen. Videostrømmen er presentert på skjermen, med dybde-, kurs- og posisjonsinformasjon vist. For å kontrollere ROVen i direkte bevegelseskontroll brukes to styrespaker på berøringsskjermen. I automatisk dybde-, kurs- og posisjonskontroll blir nye ønskede posisjoner sendt til kontroll programvaren basert på flytting av indikatorer gjort av brukeren. Disse indikatorene er vist sammen med den faktiske posisjon slik at det er lett å se både den ønskede og den faktiske posisjon samtidig.

Begge løsninger er testet med ROVen uDrone i Mc laboratoriet ved NTNU Tyholt. HMD kontrollen fungerte fint. Forbindelsen mellom 180-graders kameraet med programvare panorering forbundet med gir kontroll fungerte veldig bra. Kontrollen av jag og svai bruker hodebevegelser som ikke føles naturlig, og brukerne foretrakk å bruke spillkontrollere.

Berøringsskjerm grensesnittet gav en bedre brukeropplevelse i auto kontroll modusene, i forhold til å skrive tallverdier til kontrollerne. Direkte bevegelseskontroll, med bruk av styrespaker på skjermen, fungerte ikke så bra, og bør forbedres.

Preface

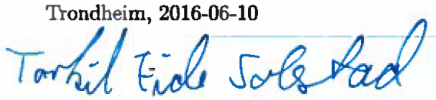
This master theses was written at Norwegian University of Science and Technology(NTNU) the spring of 2016. The main motivation of the project was to develop new solutions to improve the user experience when operating Unnmanned Underwater Vehicles(UUVs). All testing were performed at the Marine Cybernetics lab at NTNU, campus Tyholt.

The actual work load of the programming done in this project is not reflected in this report. Especially getting the Oculus Rift to work was harder than anticipated.

This project was done in cooperation with BluEye Robotics, and I want to thank them for lending me the computer to run the Oculus Rift software, and for feedback on the Human Machine Interface(HMI) solutions produced.

I want to thank my supervisor, professor Roger Skjetne, for letting me do an interesting project, and for guide along the way; Mauro Candelaro for answering questions and helping me when I was stuck; Stian and Andreas for all the hours spend in MC lab and the guys at James Bond office A1.007 for a nice last year in Trondheim.

Trondheim, 2016-06-10



Torkil Eide Solstad

Contents

List of Figures

List of Tables

Nomenclature

AUV Autonomous Underwater Vehicle.

DIY Do It Yourself.

DOF Degrees of Freedom.

HMD Head mounted Display.

HMI Human Machine Interface.

IR Infrared.

OpenCV Open Source Computer Vision.

OR Oculus Rift.

ROS Robot Operating System.

ROV Remotely Operated Vehicle.

UAV Unmanned Aerial Vehicle.

UUV Unnmanned Underwater Vehicle.

WSP Windows Surface Pro.

Chapter 1
Introduction

Unmanned Underwater Vehicles(UUVs) on the commercial marked are in most cases expensive and not affordable as consumer products. However, there has been focus lately on developing a low-cost solution. Follestad (2014) did contributions to this development with project theses the fall of 2014, and master theses the spring of 2015. The work resulted in a small low-cost Remotely Operated Vehicle(ROV) called Neptunus. The author, together with Stian S. Sandøy and Andreas V. Henriksen, continued the work on a low-cost ROV in project theses the fall of 2015, which resulted in the ROV uDrone. The contributions from the author resulted in a web application Human Machine Interface(MHI) build with HTML and JavaScript to use with a touch pad.

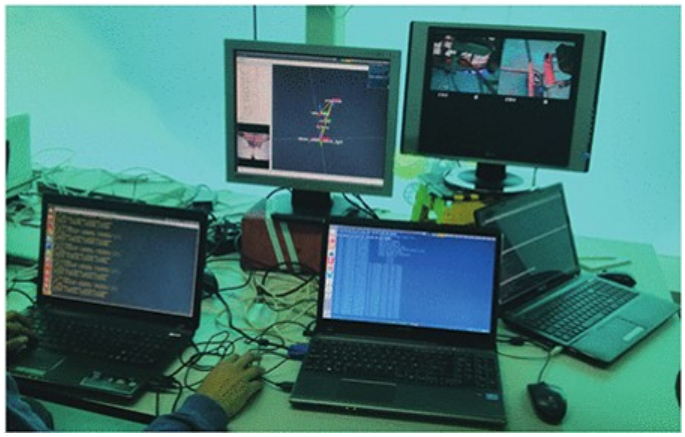


Figure 1.1: A figure of a normal Graphical User Interface(GUI) for controlling ROV/AUV systems. Notice the GUI complexity. (Juan & Sanz 2015).

To get the low-cost ROVs more accessible for recreational use improved user experiences are needed. The solutions used in the industry today to control ROVs are in most cases not usable for normal people because of complexity and prize. See

Figure 1.1 for a normal GUI for controlling a ROV system.

This theses have the objective to find new efficient Human-Machine Interface(HMI) solutions for low-cost ROVs. The HMI solutions in this project will be based on a Windows Surface Pro (WSP) touch pad and an Oculus Rift(OR) Head-Mounted Display(HMD).

1.1 Notation

This thesis uses the notation from (SNAME 1950) to define the positions and orientations a vehicle can move in. The Six Degrees of Freedom(DOF) can be seen in Figure 1.2 and in Table 1.1. The subscript v (vehicle) will be used when talking about the ROV motions, and the subscript h (head) will be used when talking about head motions.

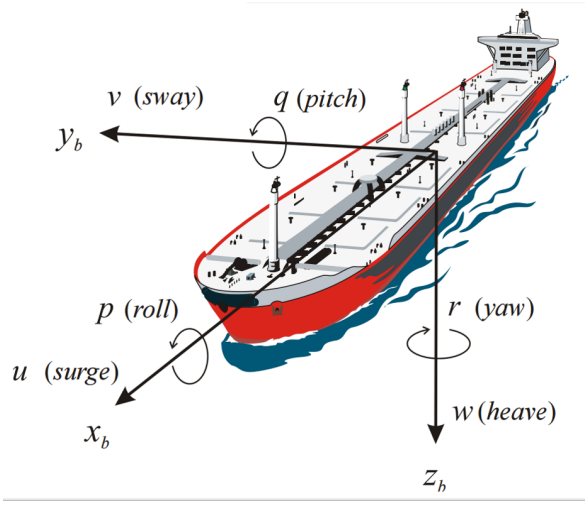


Figure 1.2: The used definition of the six degrees of freedom. (Fossen 2011).

Table 1.1: The notaion of (SNAME 1950) for marine vessels.

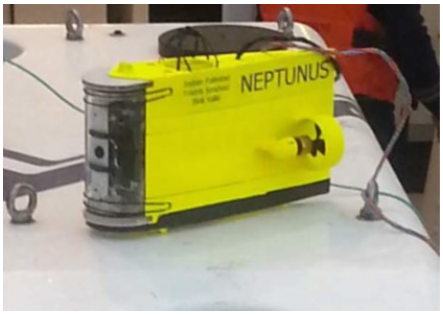
DOF		Forces and moments	Linear and angular velocities	Positions and Euler angles
1	motion in the x direction (surge)	X	u	x
2	motion in the y direction (sway)	Y	v	y
3	motion in the z direction (heave)	Z	w	z
4	rotation about the x axis (roll)	K	p	ϕ
5	rotation about the y axis (pitch)	M	q	θ
6	rotation about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

1.2 Low cost ROVs

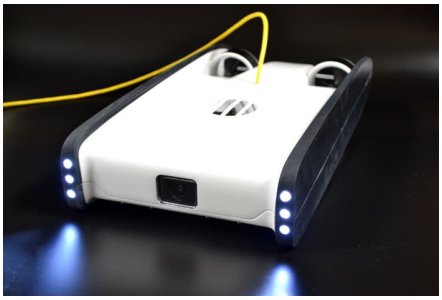
The marked for low cost ROVs is small, but quickly expanding. OpenROV was the first big producer of ROV costing less than 1000 US$. The OpenROV sells a Do It Yourself(DIY) kit where you gets a small ROV with three thrusters. The ROV is battery powered, and all the software is open source. Because of the open source it exist a big community making applications based on the OpenROV kit and software. More info about the OpenROV kit can be found in (OpenROV 2016), and a picture is seen in Figure 1.3a.

An example of an ROV based on the OpenROV kit is the work of (Follestad 2014) and (Munz 2015). This work resulted in a small low-cost ROV prototype called Neptunus. This ROV is based on the hardware from the OpenROV, but it has a different design and different software. A picture of the ROV can be seen in Figure 1.3b

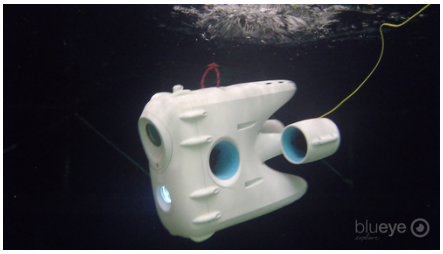
(a) The OpenROV DIY kit.



(b) The ROV Neptunus.



(c) The Trident ROV.



(d) The Blueye Robotics ROV.

Figure 1.3: Picture of different low cost ROVs.

OpenROV are also working on a small ready-to-dive ROV called Trident. This ROV is ready to use when delivered, and have a more optimized hydrodynamic shape making it able to move faster and smoother trough the water. (See Figure 1.3c.) The software on this ROV is the same as on the openROV DIY kit. The Trident is

said to begin shipment of the finished product in November of 2016. More info on the Trident can be found in (Trident 2016).

Also worth mentioning is the Blueye Robotics ROV. This is a company with a goal of making a low cost ROV for the consumer market. The candidate have worked for this company and also writes this thesis in cooperation with them. They are planning to start shipment of the first ROVs in august of 2016. More info about the ROV can be found in (BluEye 2016), and a picture of one of the prototypes can be seen in Figure 1.3d.

1.2.1 Low-cost ROVs HMI solutions.

The OpenROV have a open source software with a big community working with this software. This has resulted in several different HMI solutions working with the OpenROV. The current official version, in June 2016, can be seen in Figure 1.4. This version is a web browser HMI, made with HTML and JavaScript, served from a web server on the ROV. This HMI system needs a gaming controller to control the ROV. More info can be found in (Github 2016).

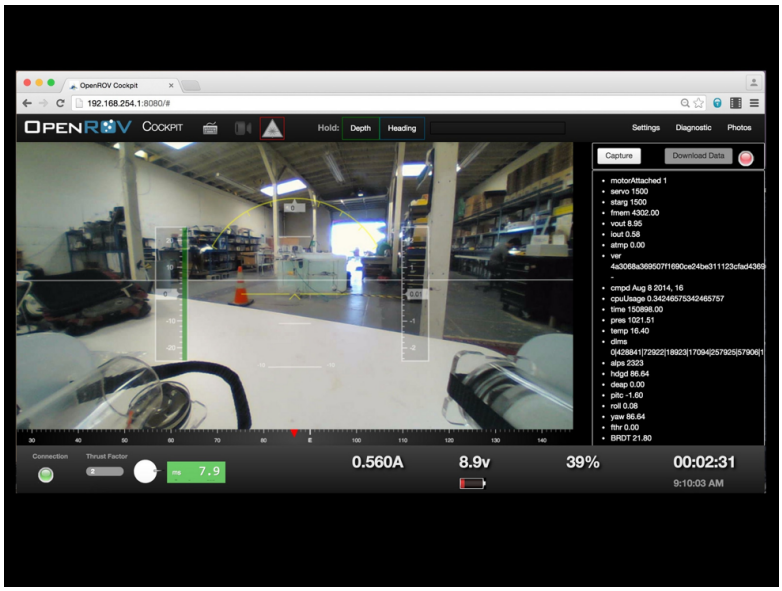


Figure 1.4: The user interface delivered with the OpenROV.

In the work of (Munz 2015) a web browser HMI is presented. It has a web-server is hosted from the ROV. The HMI is opened in a web browser by typing the IP adress of the web server. See Figure 1.5 for a picture of the HMI.

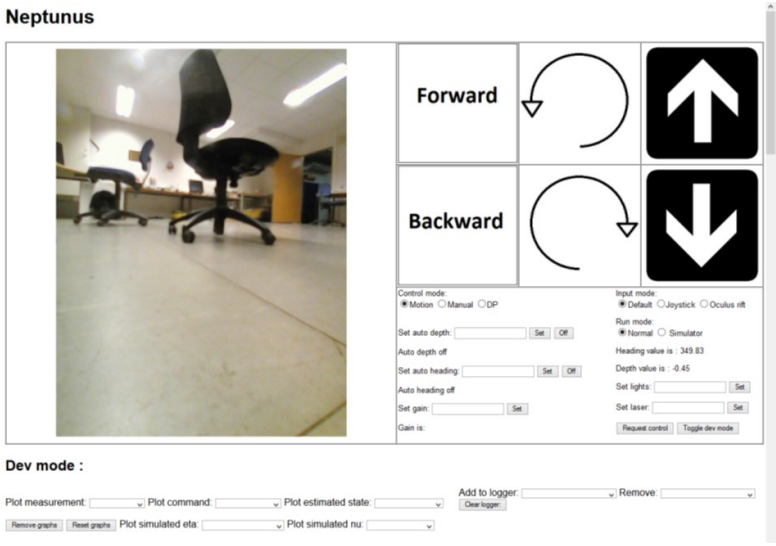


Figure 1.5: The user interface presented in (Munz 2015).

1.3 Dynamic Positioning of ROVs at NTNU AUR-Lab

Dynamic Positioning(DP) is a control system making it possible for a vehicle to automatically maintain a desired position and orientation using only its own propellers and thrusters. In (Dukan 2011) a DP control system for the Minerva ROV owned by NTNU is presented. The ROV is a SUB-fighter 7500 work class ROV. It is equipped with cameras and a manipulator arm. The ROV is used for research of the sea, and to testing new control algorithms. DP systems for the same ROV is also presented in (Dukan 2014) and (Candeloro 2012).

1.4 Head Mounted Displays

HMDs is, as the name says, a display device mounted to the users head. Normally HMDs uses one display for each eye, making it possible to achieve 3D vision. Most modern HMDs also have tracking of the head orientation, making them able to change the view on the screens based on head movements.

The use of HMDs is from the 60s, but it's first in the latest years they became popular in the commercial market. The reason for this is that now the price and available applications is at a level making them popular for normal consumers. See (Kiyokawa 2012) for a presentation of the current trends and future visions of HMDs. The use of HMDs can be divided into three main parts:

- Augmented reality: " technology that superimposes a computer-generated image on a user's view of the real world, thus providing a composite view." (Oxford 2016)
- Virtual reality: "The computer-generated simulation of a three-dimensional image or environment that can be interacted with in a seemingly real or physical way [..]."(Oxford 2016)
- Telepresence: "The use of virtual reality technology, especially for remote control of machinery or for apparent participation in distant events."(Oxford 2016)

In this theses it will focus mainly on telepretence with some levels of augmented reality to give the user of ROVs better user experiences.

1.4.1 HMDs Used in Control Applications

Using a HMD for controlling different types of drones have been tried several times before, but to the authors best knowledge (Candeloro 2015), is the only ones that have tried HMDs control on ROVs in full scale tests. Candeloro (2015) presents a way to use the head rotations with a HMD to control a ROV. This makes the hands of the operator free, which can be used to control manipulators on the ROV. Doing so, one person can do the ROV operations currently done by two persons.

Both (Juan & Sanz 2015) and (ChangSu Ha & Lee 2015) test how HMDs, combined with body movements, can be used to control ROVs. (Juan & Sanz 2015) tests how HMD can be used to get a better user experience, by using it to control the ROVs camera point of view. While this is concluded to be very helpful, the results for using head and body motions to control a ROV is more mixed. The test

participants used to gaming controllers preferred using it, while participants without game control experience found the use of head and body motions easier to learn. The article also looks at how to use augmented reality to give the user info on the state of the vehicle. See Figure 1.6. It is concluded with being very helpful for the ROV operator.

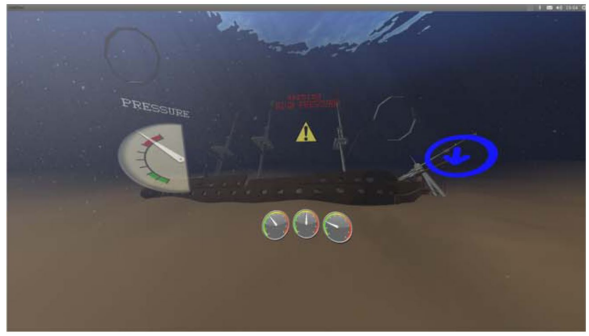


Figure 1.6: A proposed way to use augmented reality to help the user during ROV operations. (Juan & Sanz 2015).

In (ChangSu Ha & Lee 2015) the operators head, body and arms are used to control a ROV and an robotics arm. The conclusion is that the setup improved system performance, and helped the operator do the proposed tasks.

The use of HMD is also tested in control of other applications. (Pittman 2014) test different ways to control a flying drone using head tracking combined with a HMD. The article proposes four different ways of controlling the drone, compared to controlling it with a gaming controller. The conclusion is that the traditional game controller interface was superior to head tracking control. Of the head tracking control solutions, the best quantitative results came from head rotation control.

(Martins & Ventura 2015) do similar tests with a field robot. This article proposes a way of controlling the robots yaw and pitch movements with head movements, and do field tests with this setup. This is found to help the user navigate the robot.

1.4.2 Simulator sickness

Simulator sickness is a form of induced motion sickness. In normal motion sickness the discomfort comes from the actual motions, like the rolling of a ship, don't match the visual information. Simulator sickness occurs when the virtual information signals motions, but with the absence of any real movement. The symptoms include disorientation, nausea and eyestrain.

The occurrence differs from person to person, but for the users experiencing simulator sickness it can be very discomforting and destroy the desire to use the HMD. See (Oculus Rift 2016) for more info.

Since this application will include control of the moving object, the ROV, it is believed that this will reduce the risk for sickness. Still it is important to build the application with simulator sickness in mind, so the experience for the users are as good as possible.

1.5 Structure of Thesis

The thesis is organized as following: In chapter 2 the hardware used in this project is presented. This is the HMD, the touch pad and the ROV used for testing. Chapter 3 presents the software developed in this thesis. Chapter 4 presents the head motions and the touchscreen gestures used to control the ROV. Chapter 5 presents the guidance algorithms used in this thesis. The results from testing is presented in Chapter 6, and at last a conclusion and recommendations for further work is presented in chapter 7.

Chapter 2
Hardware

In this chapter all the hardware used in this project will be presented. This includes the HMD Oculus Rift(OR), the touch pad Windows Surface Pro(WSP), and the ROV uDrone.

2.1 Oculus Rift

In this project an *Oculus Rift developer kit 2* was used. The kit can be seen in figure 2.1. This is a HMD with one screen with a resolution of 960 x 1080 per eye. It has 100 degrees field of view.

The OR contains a gyroscope, an accelerometer and a magnetometer, giving an accurate estimate of the orientation of the HMD. In addition an infrared camera, seen in figure 2.1, measures the position to the OR. The combination of these sensors give full tracing in all six DOF. The definition of axes and rotations is given in figure 2.2. More info on the hardware can be found in (Rift 2016).

The OR needs a designated computer with good graphic card to run its software. The computer used is an ASUS X550JX running Windows 8.1. It has a GeForce GTX950M graphic card, a Core i7-4720HQ processor and 8GB RAM. More info can be found at (Asus 2016).

The OR have a Software Development Kit(SDK) that allows the creation of applications. For this project the SDK version 0.4.4 was used. This is a slightly outdated version. When the candidate started the work on this thesis in January of 2016, the latest version available was SDK 0.8.0. The reason for using a outdated version was that the available computer had to slow graphic card to use the new version.

Figure 2.1: *Oculus Rift Developer Kit 2.* The head mounted display used in this project. Next to the OR an infrared camera for tracking translations is seen.

Figure 2.2: The OR can measure all six degrees of freedom. This is the way the translations and rotations is defined in the OR software.

2.2 Windows Surface Pro

Windows Surface Pro 3 with Intel i7 processor is the touch pad used in this project. It is a full version Windows 10 notebook with touch screen. Because of it being a full version Windows it can run all the same programs as a normal laptop. This makes it a good testbed for implementing and testing the touch interface. For more references on WSP see (Windows 2016) and picture is shown in Figure 2.3.

Figure 2.3: Windows Surface Pro 3. The touch pad used in this project.

2.3 The ROV uDrone

The ROV uDrone is the ROV used for testing in this project. In this section the hardware and the software of the ROV will be presented.

2.3.1 Hardware Kit

The ROV uDrone is an underwater mini ROV. It was assembled by the author together with Stian S. Sandøy and Andreas V. Henriksen in a student project the fall of 2015. See (Henriksen 2015). It is based on the BlueROV Kit from BlueRobotics. This kit provided the electronic housing and 6 thrusters with motor controllers. The hardware makes the ROV fully actuated, mening it can move in all 6 degrees of freedom. More info on the kit can be found at (BlueRobotics 2015).

2.3.2 Hardware design

The hardware design in the ROV can be seen in figure 2.5. An Arduino Board run low-level programs and drivers at high frequencies. A Raspberry Pi 2 run more computationally heavy algorithms, as controllers, observers and thrust allocation at lower frequencies.

In addition to the ROV the system is set up with a topside computer. This is done to make it a easier testbed, and make communication with different hardware, like the positioning system Qualisys, easier.

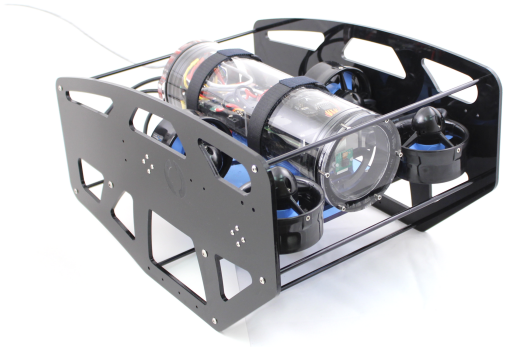


Figure 2.4: The BlueROV kit from BlueRobotics.

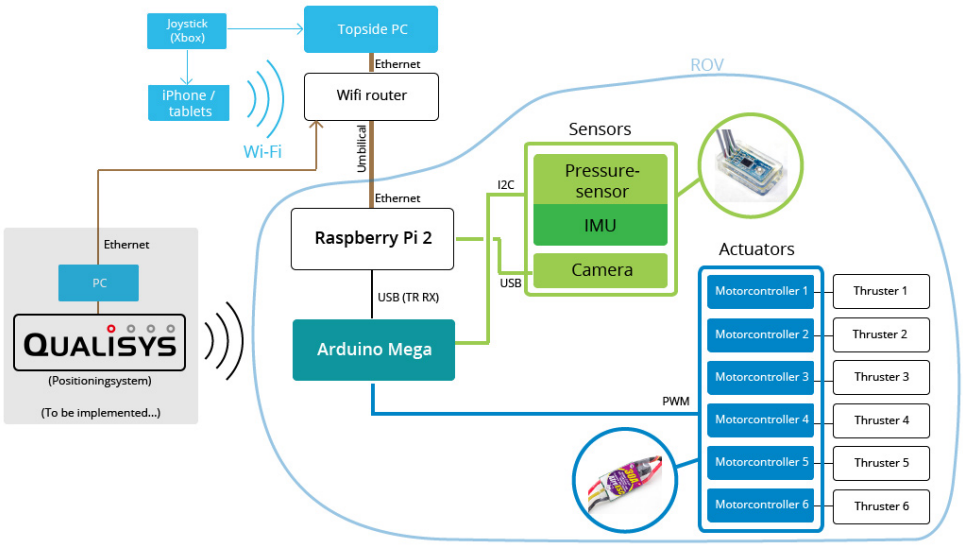


Figure 2.5: Hardware Map of the ROV uDrone. (Henriksen 2015)

2.3.3 Software

The software is written in C++ using Robot Operating System(ROS). ROS is a open source system platform designed for robotic systems. It gives a solutions to problems related to real time applications, running several processes and communication between different processes. Since ROS is open source it has a large number of

contributors, and a library consisting of hundreds of finished packages ready to use. See Cousins (2010) for a presentation on ROS and its capabilities.

In Figure 2.6 a the software structure in the ROV is presented. In this picture the different functions, and the main communication between them, are marked.

2.3.4 Camera

For video stream a 180 degree fisheye lens web-camera was chosen due to its small size and high quality video. The camera is connected and powered directly from the Raspberry Pi via USB. More info on the camera can be found in (Amazone 2016).

2.4 Hardware setup

The complete hardware setup can be seen in figure 2.7. As seen in the figure the video stream takes two different ways to the Oculus Rift and the touch pad. More info on this is found in chapter 3.2.3.

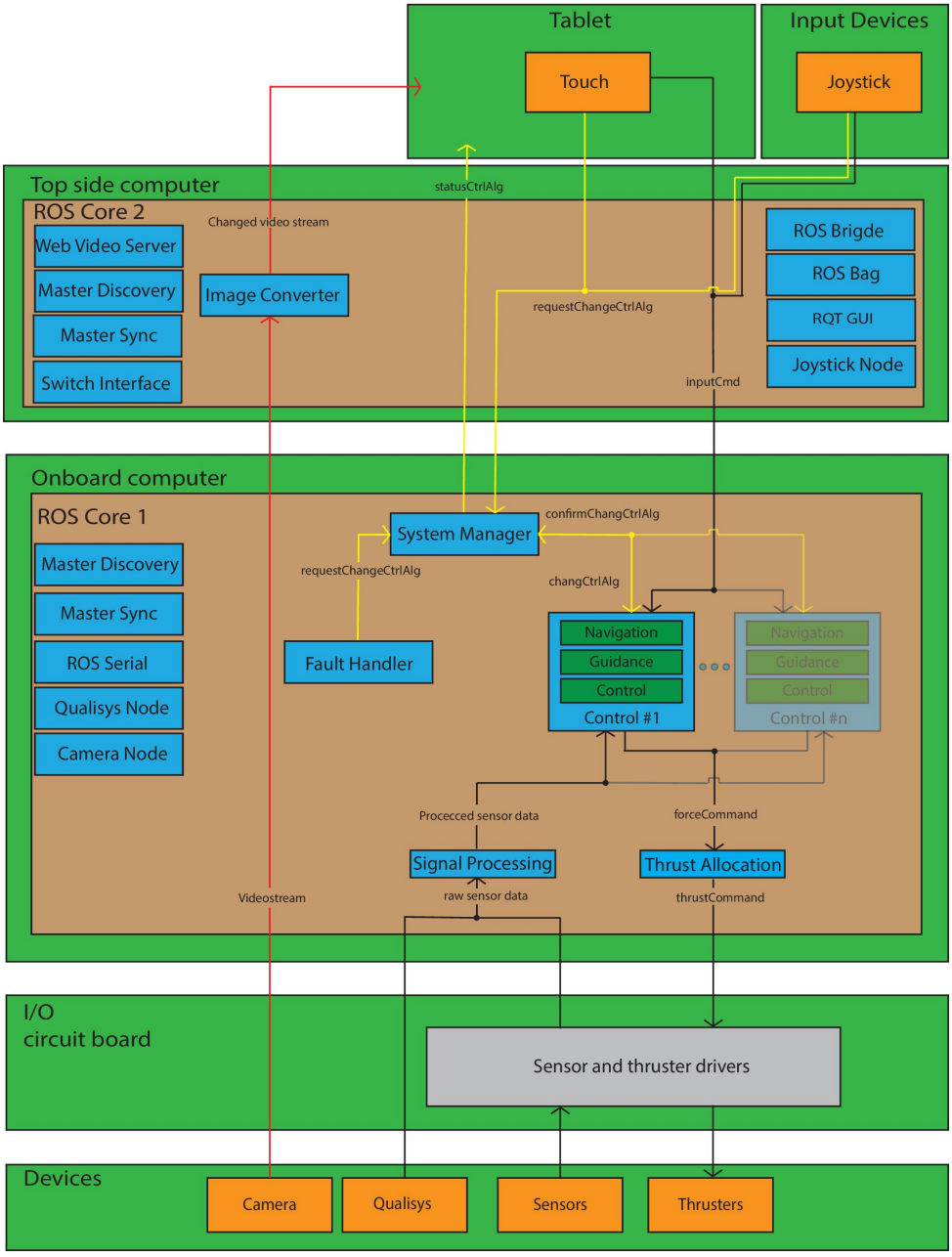


Figure 2.6: Software map of the ROV uDrone. (Henriksen 2015)

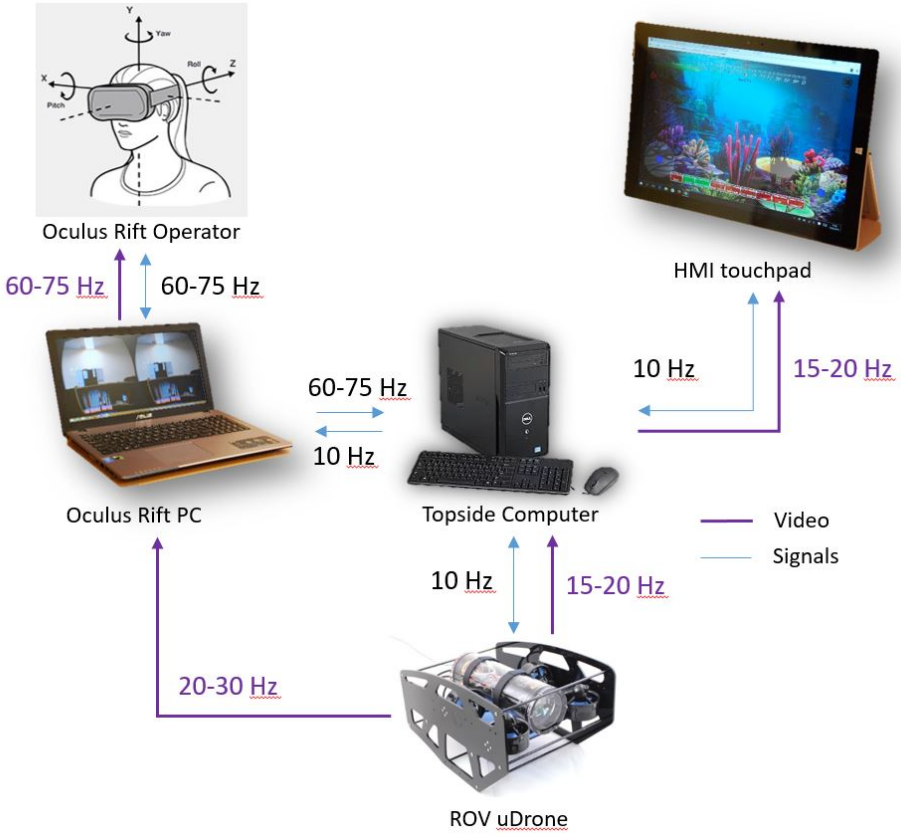


Figure 2.7: Hardware setup where communication between the different applications.

Chapter 3
Software

This chapter describes the software developed and used in this thesis.

3.1 uDrone

As seen in figure 2.6 the software on uDrone consist of several controllers, marked Control #1 and Control #n in the figure. The way the software is written makes it easy to add new controllers. When writing new controllers for this thesis, e.g. for the OR, a new controller #n was added.

3.1.1 Communication

The ROS package *ROS_bridge* is used to create a web server that non-ROS programs can communicate with. This is used as the server that receives and sends messages from/to the OR and the touch pad. The messages and commands are on the JSON object format. See (Protocol 2016) for info on the protocol used, or (RosBridge 2016) for more info on *ROS_bridge*.

3.2 Camera

The ROS program *USB_Cam*[1] creates the video stream from the camera on the ROV. To cast the video the program *web_video_server*[2] is used. It creates a web server that makes the video assessable over HTTP. The video to the Oculus Rift computer is casted straight from the ROV uDrone, with *video_web_server*. The video to the touch pad goes through the topside computer to go through the Image converter discussed later. See figure 3.1 for the software setup.

[1] Documentation can be found on http://wiki.ros.org/usb_cam
[2] Documentation can be found on http://wiki.ros.org/web_video_server

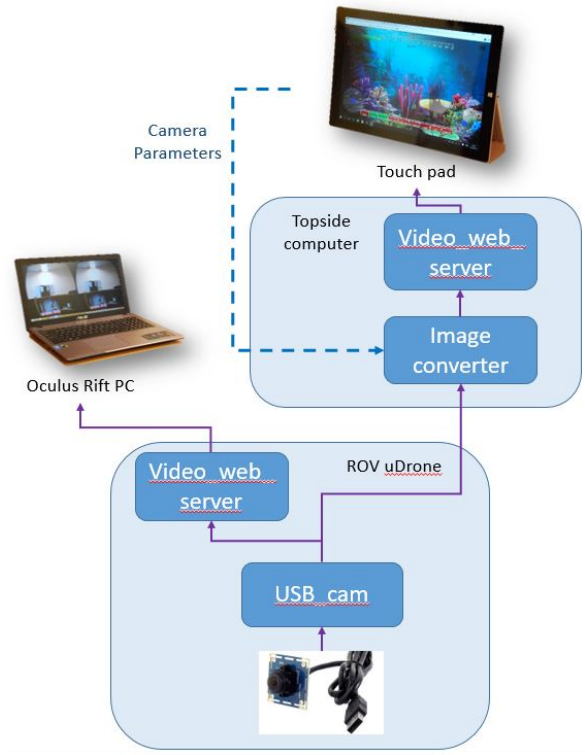


Figure 3.1: The software setup for the video stream.

3.2.1 OpenCV

OpenCV(Open source Computer Vision) is a library of programming functions aimed at real time computer vision. It saves pictures in matrices with BGR8 format. OpenCV includes several hundreds of computer vision algorithms, that easily can be used. This library is used both in the image converter software and in the OR software. Documentation on the OpenCV functions used can be found at (OpenCV 2016).

3.2.2 Image Converter

Image converter is a ROS node written to convert the video stream. This program uses openCV and is made to remove the fisheye effect from the video stream. It can also zoom, pan, tilt, change the color settings and change the brightness and contrast of the video. Because of the high computational cost to run the program it must run on the topside computer. The program receives signals, with what parameters that should be changed, from the touch pad.

Fisheye Compensation

To be able to compensate for the fish eye effect in the pictures, the ROS program *camera_calibration*[3] was used. This program uses pictures taken of a black and white grid to creates a distortion matrix and a camera matrix. See figure 3.2 for a example of a picture used for calibrating the camera. With the OpenCV function *undistort()* and the matrices a new undistorted picture is created. Also see 4.6 for a picture of the HMI with and without the fisheye effect.

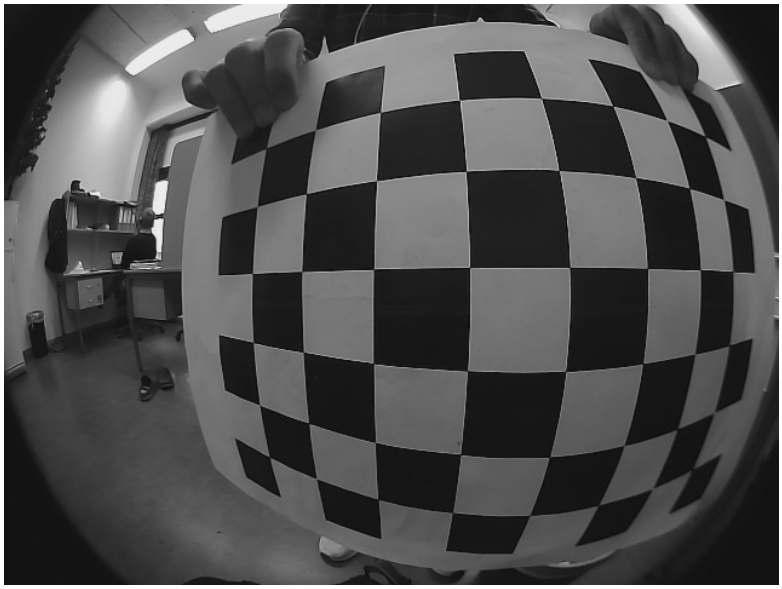


Figure 3.2: One of 35 pictures used by camera_calibration to create a distortion matrix and a camera matrix for the camera. These matrices was used to remove the fish eye effect on the picture.

Zoom/Tilt/Pan

Zoom, tilt and pan in the picture is done with the OpenCV function *wrapAffine()*. This function takes in a picture and a rotation matrix and give out the new picture. The translation matrix is created with the OpenCV function *getRotationMatrix2D(center, angle, scale)*. This function takes in the centre of the new picture, the rotation angle in degrees and the zoom scale.

3.2.3 Lag

Lag is the delay in the video. It is always a goal to have as little lag as possible, but parameters like resolution and framerate(how many pictures it is in the video each

[3]Documentation can be found on http://wiki.ros.org/camera_calibration

second) also plays a part in how much lag there will be. Because the resolution and framerate plays a part a lot of different setups was tested to see what was the most optimal. See Figure 3.3 to see a picture of how the lag was measured.

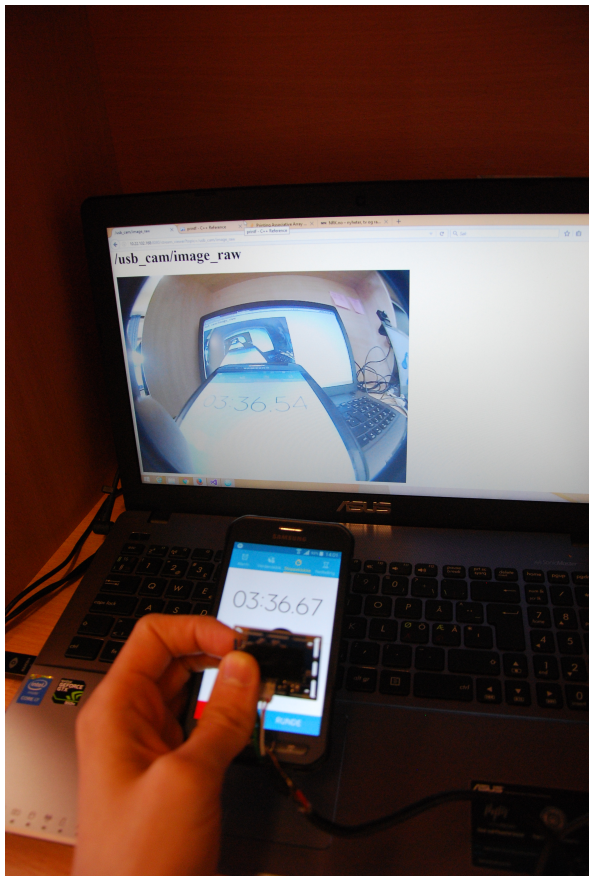


Figure 3.3: One of the setups to find out how much the lag in the video stream was. The lag is the difference between the two times, in this example 0.13 seconds.

After testing the ROV in the water several times, it became clear that as long as the lag was less than 0.2 seconds you would not notice it while controlling the ROV. The reason for this is believed to be that the ROV is slow moving. After this it was a goal to get as high framerate and resolution as possible as long as the lag was less than 0.2 seconds. The communication between the ROV and the topside computer added some latency and the same did the image converter, seen in figure 3.1. Because of this the video stream to the OR computer go past the topside computer, and the lag to the touch pad with the image converter was at minimum measured to 0.35 seconds.

The setup achieved was 20-30 hz video to the OR with 0.20 seconds lag, and a resolution of 480x640 pixels. To the touch pad the lag was a bit more and the framrate was a bit less because of the image converter. When the touch pad received the same picture as the OR, without removing the fish eye effect, it could get the same low lag and high framerate.

3.3 Oculur Rift Software

The OR software is written in C++, and is based on examples from (Davies & Benton 2015). The actual software is build on the stabilized webcam demo, combined with the photosphere example. See Figure 3.4 for a software map.

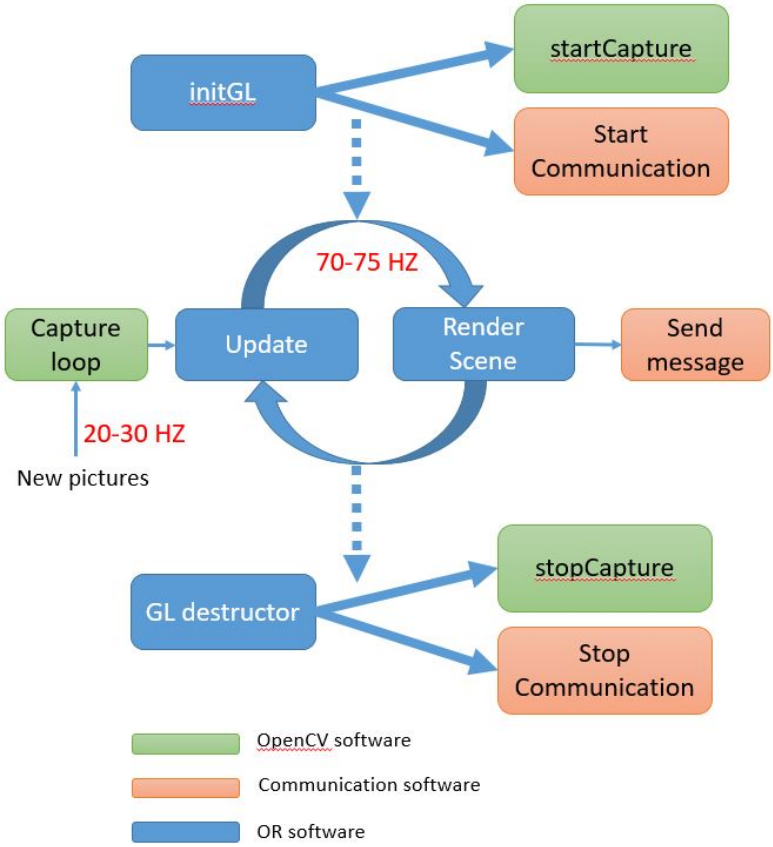


Figure 3.4: The software structure of the OR.

When the program is started, *initGL* is run. This program starts the OR, and runs *startCapture* and *startCommunication*. *startCapture* is a OpenCV program that starts reading the stream from the camera. *startCommunication* opens communication to the ROV, and sends messages about what info will be published and what info it wants to receive from the ROV.

The flow of the program is two main functions, *update* and *renderScene* which runs at 70-75 Hz. *update* calls the *capture loop*. The *capture loop* reads pictures from the ROV which arrives at 20-30 hz. Each time a new picture is avaible *update*

renders the picture to the OR. This function also recives messages with the current and desired position and write them to the screen.

renderScene updates the scene seen in the OR depending on the head rotations. This minimizes simulator sickness since the updating of the screen is at high frequencies and not dependent on the frequencies pictures are received. *renderScene* also calls the *send message* function that send the current head rotations and translations to the ROV control software.

When the program is stopped the *GL destructor* is called. This also calles *stopCapture* and *stopCommunication*, and make sure all parts of the program are stopped the right way, and all resources are released.

3.3.1 Communication between the OR and ROV

For C++ there is no library made for communicating with ROS-bridge. Therefor the open source C++ library *websocket++* was used for creating a client that communicates with the ROS software. See (Thorson 2016) for more info. The messages to/from ROS has a JSON format. The messages sent are made as text strings with JSON format, while the library *jsoncpp* (more info at (Lepilleur 2016)) was used to read the messages.

3.4 Touch Pad Software

The HMI for the touch pad is written as a web-browser interface using HTML, CSS and JavaScript. It is made for the Windows Surface Pro, for controlling the ROV using a touchscreen-based interface, while still showing the video stream from the ROV. The program is started by opening *index.html* in a web browser and this program run all the other files. See figure 3.5 for a software map showing the different parts of the program. This chapter will go through the different parts of the program.

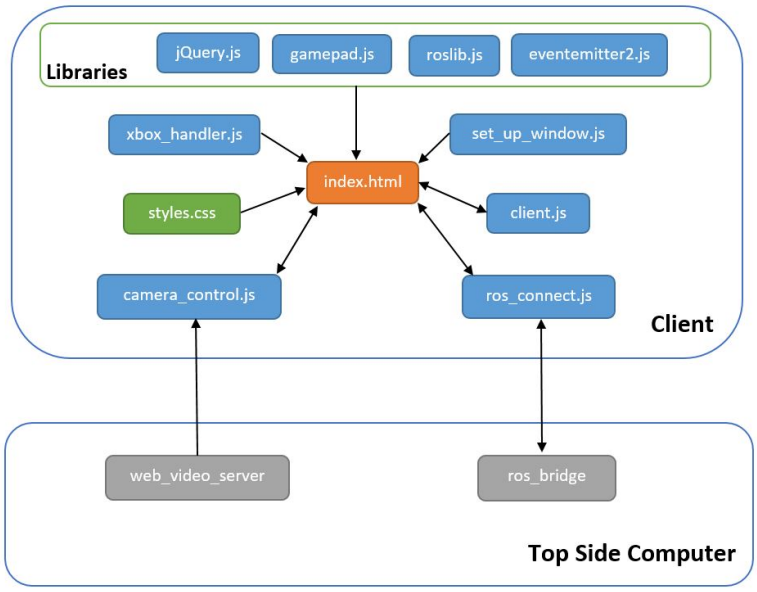


Figure 3.5: The software structure of the touch interface for the touch pad. The gray boxes are the ROS programs the HMI communicates with. The blue boxes are JavaScript files, the orange box is html-file and the green box is the Cascading Style Sheets(CSS) file.

Libraries

The HMI software depends on several libraries. These help the program in different ways. The libraries used are:

– *jQuery.js*[4] is a widely used library that enables a lot of event handlers. These enable dynamic changes on screen, like movement of boxes and images and color changes.

[4]More info can be found in the documentation on http://api.jquery.com/.

- *gamepad.js*[5] is a library that enable use of gamepads with JavaScript. This is used to control the ROV with a x-box controller.

- *roslib.js*[6] is the core JavaScript library for interacting with ROS from the browser. It uses WebSockets to enable subscribing and publishing to topics to communicate with *RosBridge*.

- *eventemitter2.js*[7] is a library that enables WebSockets communication and is used by *roslib.js*.

index.html

index.html is the main file that loads all the packages and run all the JavaScript files. All the elements that appear on screen is defined in this file.

styles.css

Styles.css is a Cascading Style Sheets(CSS) file. This is a file type where the styles used, like colors, founts, text sizes, button sizes, are set. All the elements in the interface have some properties defined in this file.

set_up_window.js

To make the HMI look nice on different platforms with different resolution and window sizes, most of the elements on screen is sized and placed depending on the size of the browser window. This ensure that the HMI fill the browser window, and that no scrolling is needed. The file *set_up_window.js* does this when the screen is loaded.

client.js

client.js fixes all movement on screen. Here the movement of the joysticks buttons and the movements of heading and depth info is handled.

camera_control.ho

The camera is controlled in the file *camera_control.js*. Here the streaming, and changing of camera parameters is managed. The video is shown using the video address in a picture element in the screen.

[5]More info can be found in the documentation on https://github.com/sgraham/gamepad.js/.
[6]More info can be found in the documentation on http://wiki.ros.org/roslibjs.
[7]More info can be found in the documentation on https://github.com/asyncly/EventEmitter2.

ros_connect.js

Communication with the ROS code on the ROV is done in *ros_connect.js*. This file handles all the inputs and outputs from the HMI except the camera stream. It also keeps track of the current control mode.

xbox_handler.js

The file *xbox_handelr.js* handles the x-box controller interface. This makes it possible to control the ROV with a x-box controller connected with USB to the touch pad.

Chapter 4
Touchscreen Gestures and Head Motions to Control the ROV

As mentioned in the introductions there are many different ways to control ROVs using HMDs and touch screens. In this chapter different methods will be discussed and the ones implemented will be explained, first for the HMD, then for the touch pad.

4.1 ROV control through HMD

The OR can measure the position and orientation of the users head in all six degrees of freedom. This gives six possible head movements to control the ROV. In the following section the camera control, the yaw control, the surge control and the sway control will be explained.

4.1.1 Camera control

The proposed video setup in the OR is to show only a part of the whole 180 degrees of the video. The video from the camera is put on the inside of a 3D ball, 180 degrees around the user. The OR user is free to rotate his head inside this ball, and by doing this changing the part of the video stream that is seen inside the HMD. Since the view from the OR is 100 degrees, the user can rotate his head 40 degrees in yaw or pitch before the user reach the edge of the video. E.g. rotating the head in yaw, the user will see a different part of the video stream. See Figure 4.1.

4.1.2 Yaw Control

To control the ROV in yaw the proposed setup is to move the head in yaw. See Figure 4.2. This is the same control method found in all the references.

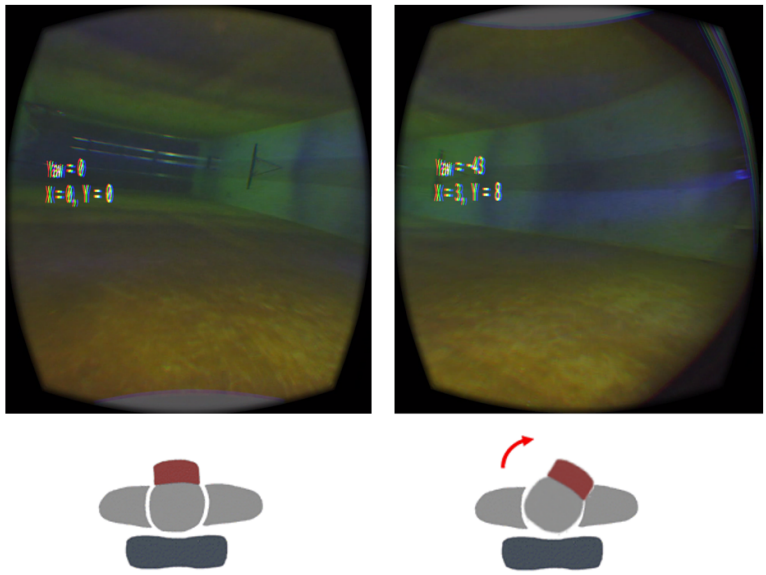


Figure 4.1: The picture inside the OR, when the user rotates his head in yaw. In this picture the head is rotated 43 degrees clockwise in yaw. In the right picture the user can see the edge of the video stream.

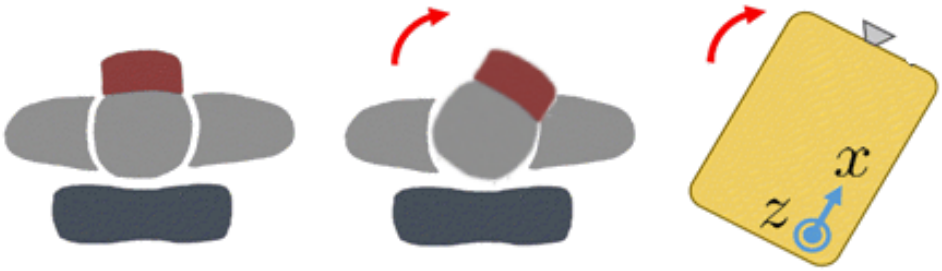


Figure 4.2: The proposed head movements giving yaw command to the ROV. In the figure a clockwise rotation of the head in yaw give a clockwise motion around the z axis for the ROV. Figure from (ChangSu Ha & Lee 2015).

4.1.3 Surge Control

The two different ways proposed to control the ROV in surge are head movement in surge and head movement in pitch. (Candeloro 2015) used pitch head movements, and (Pittman 2014) tried both and concluded that head movement in pitch was the best way. This was because it was harder for the user to find the zero position in translation than in rotation.

After initial testing it became clear that the camera setting in this project makes the head movement in pitch seem less appealing. Since the video is not locked in front of you, as explained in section 4.1.1, all rotations of the head have a impact on what parts of the video stream you see. If the user wants to go forward with pitch control the picture shown on the screen will be the bottom part of the video, and not what is right in front of the ROV. Because of this, surge movement was chosen to control the sway movements of the ROV. See Figure 4.3.

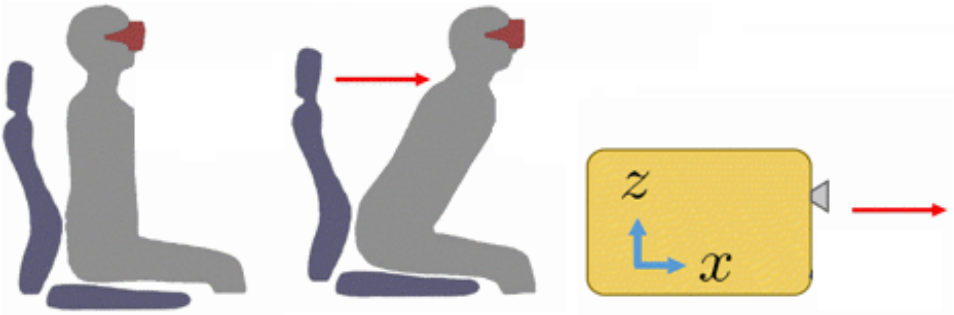


Figure 4.3: The proposed head movements giving surge command to the ROV. A movement with the head in positive x-direction will create a movement in positive x direction for the ROV. Figure from (ChangSu Ha & Lee 2015).

To make it easier for the user to find the zero point in translation, the position and yaw angle of the HMD is shown to the user. See Figure 6.2, where the yaw angle and the x and y position from the zero position is shown. This made it easier to control the ROV with translation and overcome one of the problems from (Pittman 2014), where this was not done.

4.1.4 Sway Control

In sway, as in surge, there are two ways proposed to control the the ROV. This is head movements in sway and head movement in roll. (Candeloro 2015) used roll head movements, and (Pittman 2014) also concluded that this was the best way, because of the difficulties of finding the zero point in sway. Some initial testing was done, and it was concluded that a sway head movements felt more natural. Because the translation is shown, some of the problems mentioned in (Pittman 2014) is avoided, and head movement in sway is chosen to control the ROV in sway. See Figure 4.4.

4.1.5 Heave Control

The scope of this theses does not include using HMD to control the heave motion of the ROV. Instead an auto depth controller keep the ROV at a constant depth. This depth can be changed with the touch pad HMI.

Figure 4.4: The proposed head movements giving sway command to the ROV. A movement in positive y-direction with the head gives a movement for the ROV in positive y-direction. Figure form (ChangSu Ha & Lee 2015).

4.2 Touchscreen Gestures to Control ROVs

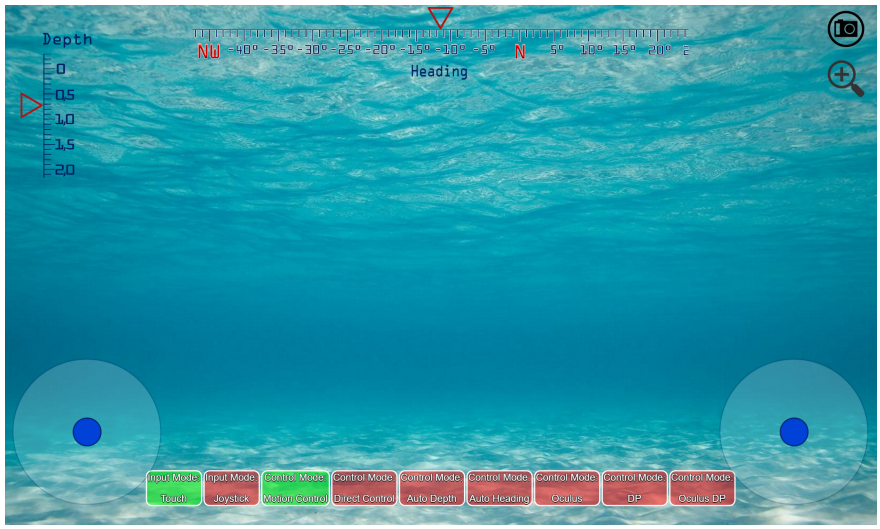


Figure 4.5: The touch pad interface in direct motion control. Notice the heading index at the top of the screen, the depth index on the left side of the screen and the zoom button on the right side of the screen. At the bottom of the screen, on each side, the joysticks, used to control the ROV in direct motion control are shown. In the middle the current control mode can be seen, and this is also where the control mode is changed.

Because of the amount of work required to implement new ways to control the ROV with touchscreen gestures, only the ways discussed in this chapter was tested.

In the following section the camera control, the direct motion control, the surge control and the sway control will be explained. Figure 4.5 show the touch pad HMI.

4.2.1 Camera Control

The camera on the ROV is a 180 degrees fisheye camera and without any software to change the video it has a black border and the picture is distorted at the edges. The fisheye effect can be removed with the use of software, as seen in Figure 4.6. This effect can be turned on and off by the user, but the standard is to remove the fisheye effect.

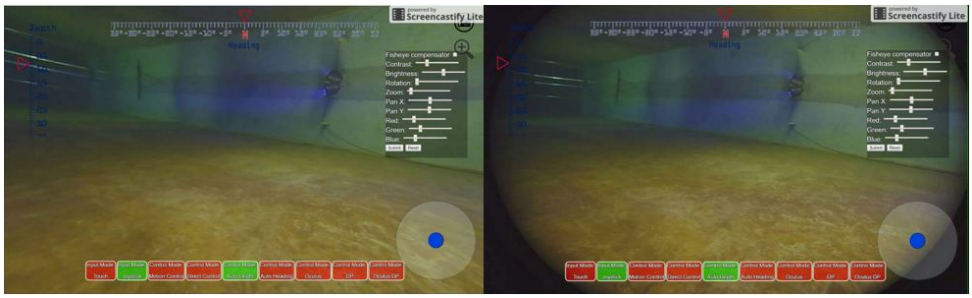


Figure 4.6: **Left**: Video with removed fisheye effect. **Right**: Video with fisheye effect straight from the camera.

If the user touches a zoom-button on the side of the screen the picture zooms in, first to double size then fourfold size. When this happens the user can pan the zoomed picture by dragging the finger over the screen. See Figure 4.7

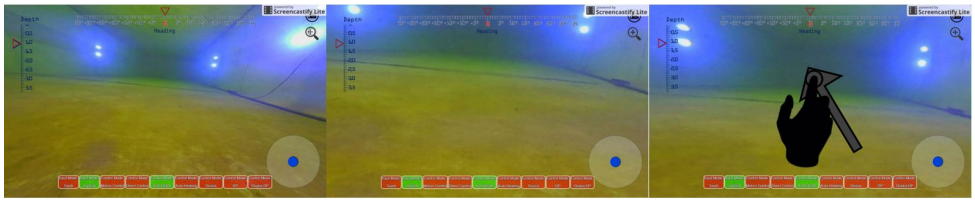


Figure 4.7: **Left**: Normal video. **Middle**: The video is zoomed in to double size. **Right**: The user drag his finger ut and to the left to pan in the video. The ROV is kept still in in the time between the pictures. Touch gesture icons from (Mobiletuxedo 2016).

4.2.2 Direct Motion Control

In direct motion control the ROV is controlled with two joysticks on the touch screen. All four degrees of freedom can be controlled this way. See Figure 4.8 where the direction controlled is marked.

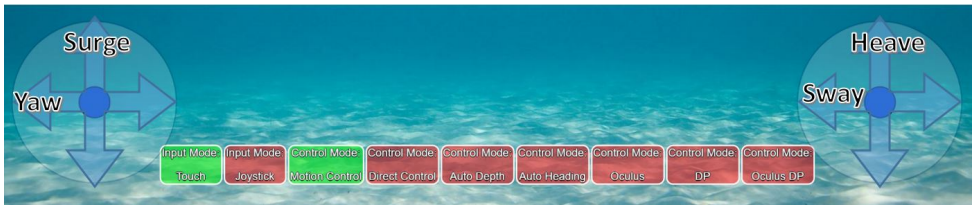


Figure 4.8: Direct motion control with touch screen. The two joysticks control four degrees of freedom on the ROV.

4.2.3 Auto Depth Control

A depth marker is always shown on the left side of the screen, see Figure 4.5. This is done with a red marker and a bar that moves up and down. If auto depth is turned on, a blue marker showing the desired depth appears. To change the desired depth the user drags the depth bar up or down. When the user releases the bar it goes back to showing the current depth, with the extra marker showing the desired depth. See Figure 4.9 where the desired depth is changed.

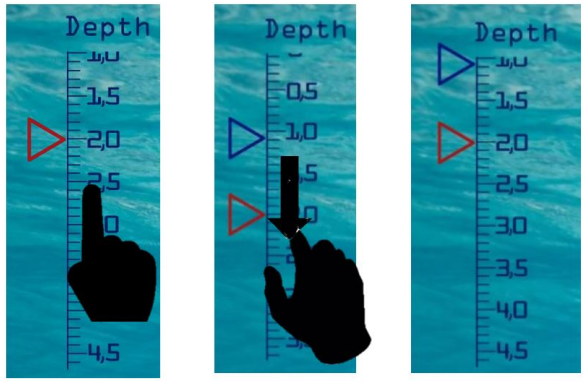


Figure 4.9: The proposed touchscreen gestures giving commands to the auto depth controller. The user drags the finger down while touching the depth bar. This set a new desired position at a lower depth. When the finger is lifted from the touch pad, the bar goes back to the current depth. The blue marker is desired depth and the red marker is current depth. Touch gesture icons from (Mobiletuxedo 2016).

4.2.4 Auto Heading Control

The desired heading is controlled in a similar way as the desired depth. The heading is shown at the top of the screen all the time, see Figure 4.5. When auto heading is enabled an auto heading reference marker appears. The same way as in auto depth control, by dragging the heading bar, a new desired heading is set.

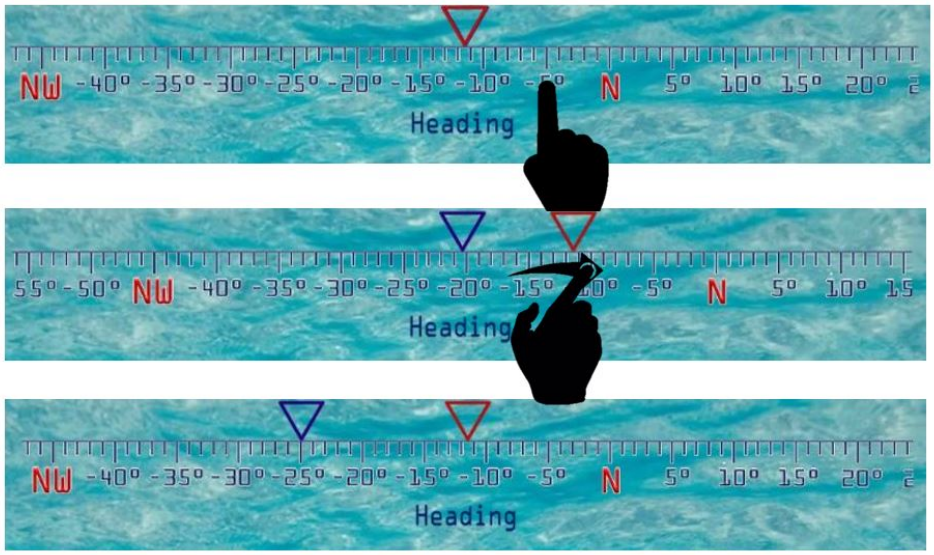


Figure 4.10: The proposed touchscreen gestures giving commands to the auto heading controller. The user drags the heading bar to set a new desired heading. The blue marker is desired heading and the red marker is current heading. In the figure the user moves his finger to the right, setting a desired heading to the left of the current heading. Touch gesture icons from (Mobiletuxedo 2016).

4.2.5 Auto Position Control

This mode is used to set the desired X and Y position of the ROV in a 4 DOFs DP controller. The 4 DOFs are surge, sway, heave and yaw. The proposed way to do this is to see a "map" of the area the ROV moves in, where a marker can be moved in the X and Y direction. This gives new desired position relative to the current position of the vehicle. After the user release the blue marker, the blue marker will move to show the difference between the current and the desired position. See Figure 4.11.

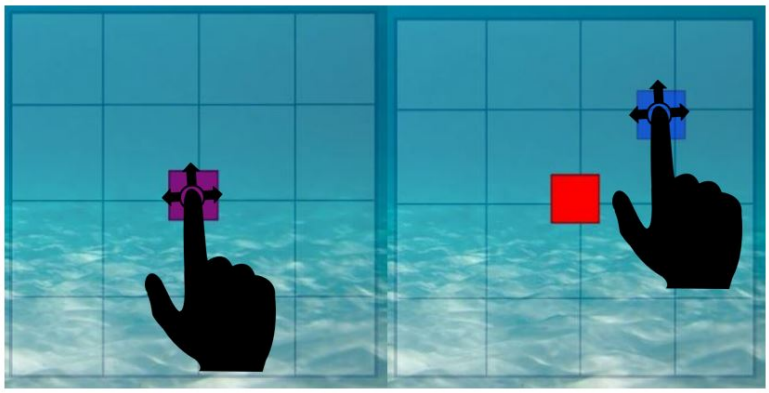


Figure 4.11: The proposed touchscreen gestures giving commands to the auto position controller. The blue marker shows the desired position. In the Figure the user moves the desired position marker to the top right, giving a positive desired position in X and Y. Touch gesture icons from (Mobiletuxedo 2016).

Chapter 5
Guidance Algorithms

In this chapter the different control modes; the guidance algorithms with the HMD; the guidance algorithms making direct motion commands; and the guidance algorithms making position/depth/heading references will be explained.

5.1 Control Modes

uDrone has three different auto control modes, in addition to direct motion control and direct thrust control. These are auto depth control, auto heading control and auto position control. These modes can be combined in different ways. In the scope of this thesis control of heave with HMD will not be discussed. See table 5.1 and 5.2 for the different control modes and what DOFs that can be controlled.

Control modes for touch pad		
Control mode	**Controlled directly**	**Controlled by setpoint**
Direct Motion Control	X + Y + Z + ψ	
Auto Depth Control	X + Y + ψ	Z
Auto Heading Control	X + Y + Z	ψ
Auto Depth and Heading Control	X + Y	ψ + Z
Auto Position and Heading Control (Full DP)		X + Y + Z + ψ

Table 5.1: Different control modes for touch pad with which DOFs can be controlled in each.

5.2 HMD

From initial testing and several sources, e.g. (Candeloro 2015) and (Valle 2015) it became clear that using a HMD for input into the control system would be easier if

Control modes for HMD			
Control mode	**Controlled directly**	**Controlled by setpoint**	**Controlled automatically**
Direct Motion Control	X + Y + ψ		Z
Auto Heading Control	X + Y	ψ	Z
Auto Position and Heading Control (Full DP)		X + Y + ψ	Z

Table 5.2: Different control modes for HMD with which DOFs can be controlled in each.

including a dead band to the controller. This makes it easier for the user to find the zero head position. In direct motion control the deadband is 5 cm movement and 25 degrees rotation in yaw. In auto heading the deadband is 5 degrees.

The inputs from the HMD, Θ, is the angles and translations rotated from the zero point. This zero point for yaw, x, y and z is set when the program starts, and can be reset later. This is important so the user can find a good/natural position to have as zero point when controlling the ROV. Θ is given as:

$$\Theta_h = [\phi, \theta, \psi, x, y, z]^T \in [-1, 1] \tag{5.1}$$

where ϕ is roll, θ is pitch and ψ is yaw angles of the HMD, and the translation and orientation directions is given as in figure 2.2. The values are normalized between -1 and 1, and saturated at 40 cm translation and 100 degrees rotation in yaw.

5.3 Direct Motion Control

5.3.1 HMD

In the direct motion control the HMD control is a feed-forward force control. The commands from the HMD is multiplied with a gain vector and a transformation vector to give the force commands to the thrust allocation, given as in 5.2. The thrust allocation is given in (Sandøy 2016).

$$\tau_v = KT\Theta_h \tag{5.2}$$

τ_v is the four element thrust vector, with the controllable degrees of freedom, given as:

$$\tau_v = [X, Y, Z, N]^T \tag{5.3}$$

K is a gain matrix giving the max thrust in each degree of freedom given as:

$$K = diag[X_{max}, Y_{max}, 0, N_{max}]. \tag{5.4}$$

T is the translation matrix that translate from the HMD frame to the ROV frame given as in 5.5. Notice the negative numbers in x and yaw, this is to give the right translation and rotation for controlling the ROV, as explained in chapter 5.

$$T = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \end{bmatrix} \tag{5.5}$$

5.3.2 Touch Pad Interface

For the touch pad the control input is the desired force in each DOF from the joysticks. The commands is given between [-1,1], and multiplied with max force before sent to the thrust allocation. See (5.6). K is a gain matrix given in 5.7, τ_v is the thrust vector given in 5.3, and the input from the joystick τ_{in} is given in 5.8.

$$\tau_v = K\tau_{in} \tag{5.6}$$

$$K = diag[X_{max}, Y_{max}, Z_{max}, N_{max}] \tag{5.7}$$

$$\tau_{in} = [X_{in}, Y_{in}, Z_{in}, N_{in}]^T \tag{5.8}$$

5.4 Auto Control

The auto control modes on the ROV uDrone are auto depth control, auto heading control and auto position control. The control algorithms are developed by Andreas V. Henriksen and Stian S. Sandøy in their project and master theses. For reference on

auto depth and auto heading see (Henriksen 2015) and for referance on auto position control see (Sandøy 2016). The controllers have a reference position/orientation as input.

5.4.1 HMD

To compute the desired positions and heading equation 5.9 is used. This integrates the position and orientation of the HMD multiplied by a gain matrix and a transformation matrix. η_d is the desired positions and rotation given in 5.10. The transformation matrix is given in 5.5.

$$\eta_d = \int_0^t KT\Theta_h d\tau \tag{5.9}$$

$$\eta_d = [x_d, y_d, z_d, \psi_d] \tag{5.10}$$

K is a gain constant, tuned to make the control feel natural. This was made so that a HMD movement of 0.1 m forward in 1 seconds move the desired position with 0.1 m forward.

5.4.2 Touch Pad Interface

The touch pad sends reference points directly to the auto controller algorithms. The touch pad transforms the finger movements to match the values seen on the screen. The reference value can be seen in the figures while the user is making the finger movements. See e.g. figure 4.9 where the desired depth sent to the controllers is 1 meter.

Chapter 6
Results

In this chapter the results from testing will be presented. All testing was done in the MC lab at NTNU Tyholt with the ROV uDrone. The MC lab is a 1.5 meter deep wave basin, with a underwater positioning system, available of give the position and orientation of ROVs. Because of limited time in the MC-lab and problems with getting a good position measurement for the ROV, most of the tests where done in direct motion control. Early in the project there was also a goal to do a high number of tests, with different operators, but this was only partly completed due to lack of time. This chapter will go though the testing of the different parts of the OR and the touch pad interface, and discuss what worked fine, and what that not work so good.

6.1 OR

A picture from a test with the OR is shown in Figure 6.1. A movie of what the user see in the OR from the testing can be found at: https://youtu.be/JBZvZk8ulno .

6.1.1 Camera Control

The dynamic panning in the video feed explained in chapter 4.1.1 worked very well. Some test subjects mentioned this felt natural and greatly improved the user experience. The view could remind of the view when driving a car, where the ROV moves one direction even if the head is turned.

6.1.2 Yaw Control

The yaw control in combination with the software panning felt natural to use. The deadband while turning the head, makes the turning first done in the software and if the head is tuning is big enough the ROV starts turning. The effect makes it easy for the user to decide what to look at.

Figure 6.1: Picture from a test with the OR in the MC lab.

6.1.3 Direct Motion Control

The direct motion control did not feel natural. The control worked, and the user can see the current position of the head, so was easy to find the position giving zero thrust command. See Figure 6.2. The problem was that the head movements felt unnatural and to move the ROV over longer distances the user needed to keep the head in unatural positions over a long time. This did not feel good for the neck. Maybe some tuning of the saturation could help, so the max force was achieved with less than 0.40 m movements. Most people preferred using a joystick to control the

ROV, while using the OR on the head just to keep the camera control.

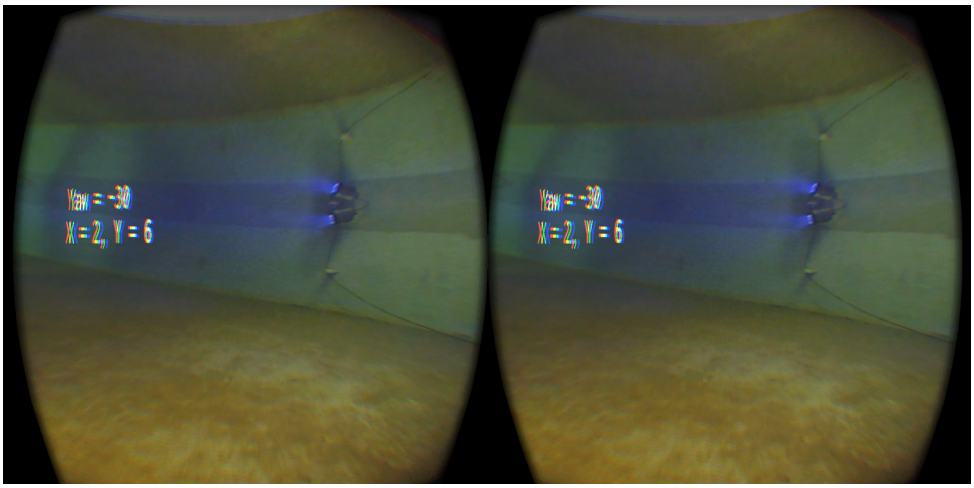


Figure 6.2: A picture of the video stream shown in the Oculus Rift. This picture is form testing the direct motion control. The two parts is divided to each eye. The text seem out of focus but it makes a 3D effect seen by the user.

6.1.4 Auto Position control

The controlling using auto position control was easier that direct motion control. In this mode the current position and desired position can be seen on the screen. See Figure 6.3. The ROV also kept the position better when no commands was given. This made control easier, and made it easier to get the ROV to a desired position.

6.1.5 Simulator Sickness

The author did not feel any signs of simulator sickness, but this is not a good estimate for the risk of getting sick. When getting used to head mounted displays, the risk of getting sick is greatly reduced.

Some other test subjects mentioned that the become a bit dizzy when asked, but no subjects reported any big problems.

6.2 Touch Pad Interface

A video from testing the HMI on the touch pad can be found here: https://youtu.be/hv2t0JuLY2A . All the control modes discussed in chapter 4.2 worked and made the operator able to control the ROV. Figure 6.4 show a picture from the testing.

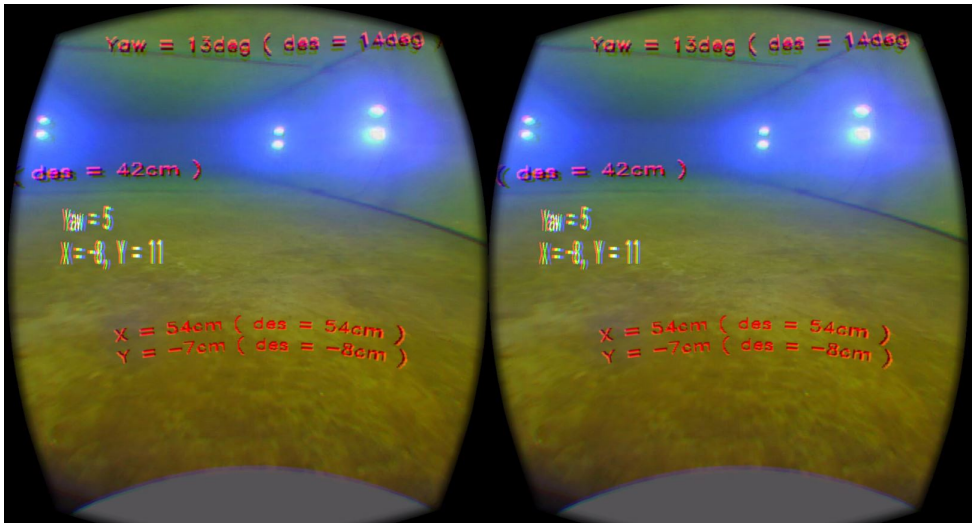


Figure 6.3: The view in the OR while in DP mode. The current and desired positions is showed. This makes control easier. The two parts of the picture is for each eye.

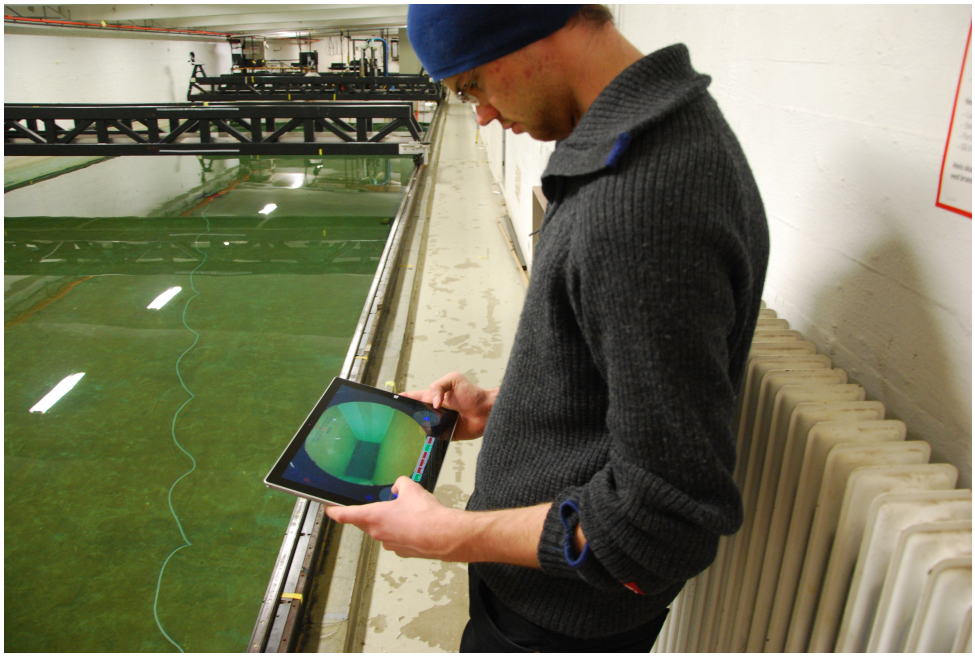


Figure 6.4: Picture from a test with the touch pad interface in the MC Lab.

6.2.1 Direct Motion Control

The use of joysticks on the screen to control the ROV in direct motion control was not easy. Since you get no response from the joysticks, the user can not feel in what position the control is in. This made it necessary to look down at the joysticks while using them, to see what position they where in. It was much easier to control the ROV with a game controller, while looking at the touch pad for video feed and graphical references.

6.2.2 Auto Heading/Depth

The auto position mode works well. It it easy to set a new desired depth/heading. The current position is also shown at all time, and as long as the controllers work well, this is a better way to set the desired position than to write the numerical values into the controller. In the presented way the risk for miss touch is limited, since the user have to drag the bars to the value wanted. It is also easier to get a sense of the distance from the current value the new desired value is because the current position is shown at all time.

6.2.3 Auto Position Control

The auto position mode is easy to control. For testing in the lab it works very well. The interface is limited to setting a new desired position +/- 1.25 meters from the current position. This is a suitable value in the test tank, but could be less viable in the ocean. One idea can be to be able to zoom in the window, to be able to change the max value.

Chapter 7
Conclusion

In this theses two new HMIs was developed and tested to control a small ROV. Both methods worked and is believed to give the user an improved user experience.

The OR interface gave the option to control the ROV in direct motion control or different levels of automatic control. The use of a 180 degree fisheye camera give the user a better view compered to a normal camera, and make it possible to move around in the picture only using software. The interface also used some levels of augmented reality by showing the current head orientations and positions, in addition to the current and desired ROV positions on the screen. This is believed to help the user controlling the ROV, compared to previous projects done with controlling ROVs with head mounted displays.

The touch screen interface works good to control desired positions in auto control modes, but is hard to use in direct motion control. This will need more work before it become preferable, compared to using a game controller to control the ROV. The way the desired positions is set greatly increase the user experience, and make is easier for the operator while controlling ROVs.

Chapter 8
Further Work

For later projects there are several things that can be looked more into to make the results of the project better. The further work will be divided into the two platforms used in this work, the HMD and the touch pad.

8.1 HMD

More User Tests

To get more quantitative data on the user experience some tests with several user should be conducted. This way a conclusion on the risk for simulator sickness, and which control mode that is most popular or easiest to use could be achieved. One example could be to do the some ROV operations, like controlling it on a track, with several test participants and several control modes, and find out what control mode had the lowest average completion time. This way it would be much easier to conclude on the best way to control a ROV.

Stereo Vision

Stereo Vision uses two cameras to give the user a 3D vision of what is happening. Whit this it would be easier to see distances under water. This can enhance the user experience, but may be at the expense of the current setup with the 180 degree camera.

Working on a ship

Many ROV operations are done with the operator being on a ship. Depending on the weather, the movements on the ship can be substantial. This can disturb the measurements of the orientation of the head, and make it much harder to control the ROV with head movements.

8.2 Touch Pad

Video Stream

As discussed in section 3.2.3 the lag in the video when removing the fish eye effect is to high. This comes from the test setup run now, with one computer running the camera, one computer removing the fish eye effect, and the touch pad viewing the video. See Figure 3.1. This could be changed ether by making the bottom computer stronger, and then make it able to run the image converter algorithms, or to include the algorithm in the touch pad.

Direct Motion Control with Joysticks

As discussed in the results section the joystick control with the touch screen is not that good. A way to smarter control the ROV in direct motion control, without joysticks, or a better implementation of the joysticks, should be considered.

References

Amazone (2016), 'Wide angle camera'. Viewed 08.06.2016.
URL: *http://www.amazon.com/180degree-Fisheye-1080p-Angle-Camera/dp/B00LQ854AG*

Asus (2016), 'Asus x550jx product info'. Viewed 20.02.2016.
URL: *https://www.asus.com/Notebooks/X550JX/*

BlueRobotics (2015), 'Bluerov'. Viewed 08.06.2016.
URL: *https://www.bluerobotics.com/store/rov/bluerov-r1/*

BluEye (2016), 'Blueye-what we do'. Viewed 08.06.2016.
URL: *http://www.blueye.no/what-we-do/*

Candeloro, M., S. A. J. L. S. D. F. (2012), 'Observers for dynamic positioning of rovs with experimental results', *9th IFAC Conference on Manoeuvring and Control of Marine Craft* volume 1,, pages 85–90.

Candeloro, M., V. E. M. M. R. S. R. L. M. . S. A. J. (2015), Hmd as a new tool for telepresence in underwater operations and closed-loop control of rovs.

ChangSu Ha, Sangyul Park, J. H. I. J. Y. L. G. R. C. H. I. S. & Lee, D. (2015), 'Whole-body multi-modal semi-autonomous teleoperation of mobile manipulator systems'.

Cousins, S. (2010), 'Welcome to ros topics'.

Davies, B. A., B. K. & Benton, A. (2015), *Oculus Rift in Action*, Manning Publications Co., Shelter Island, USA.

Dukan, F. (2014), ROV Motion Control Systems, Thesis, Norwegian University of Science and Technology.

Dukan, F., L. M. S. A. J. (2011), 'Dynamic positioning system for a small size rov with experimental results'.

Follestad, J., S. F. V. E. (2014), Low cost ROV design, based on testing, simulations and analysis of OpenROV., Thesis, Norwegian University of Science and Technology.

Fossen, T. I. (2011), *Handbook of Marine Craft Hydrodynamics and Motion Control*, John Wiley & Sons Ltd.

Github (2016), 'Openrov software'. Viewed 08.06.2016.
URL: *https://github.com/OpenROV/openrov-software*

Henriksen, A. V., S. S. S. (2015), Hardware and Software Design of uDrone, Thesis, Norwegian University of Science and Technology.

Juan, C. G., B. P. J. P. J. S. P. M. J. D. & Sanz, P. J. (2015), 'Towards an immersive and natural gesture controlled interface for intervention underwater robots'.

Kiyokawa, K. (2012), 'Trends and vision of head mounted display in augmented reality', *2012 International Symposium on Ubiquitous Virtual Reality* pp. p. 14–17.

Lepilleur, B., D. C. (2016), 'Jsoncpp documentation'. Viewed 08.06.2016.
URL: *http://open-source-parsers.github.io/jsoncpp-docs/doxygen/index.html*

Martins, H., O. I. & Ventura, R. (2015), 'Design and evaluation of a head-mounted display for immersive 3d teleoperation of field robots', *Robotica* 33(10), 20.

Mobiletuxedo (2016), 'Touch gesture icons'. Viewed 02.06.2016.
URL: *http://www.mobiletuxedo.com/touch-gesture-icons/*

Munz, J. (2015), Design and Implementation of Software for the ROV Neptunus, Thesis, Norwegian University of Science and Technology.

Oculus Rift, D. (2016), 'Simulator sickness'. Viewed 08.06.2016.
URL: *https://developer.oculus.com/documentation/intro-vr/latest/concepts/bp_app_simulator_sickness/*

OpenCV (2016), 'Geometric image transformations'. Viewed 08.06.2016.
URL: *http://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html*

OpenROV (2016), 'Openrov 2.8 mini observation class rov'. Viewed 08.06.2016.
URL: *http://www.openrov.com/products/2-8.html*

Oxford, U. P. (2016), 'Oxford dictionaries, online dictionary: Augmented reality, telepresence and virtual reality'. Viewed 08.06.2016.
URL: *http://www.oxforddictionaries.com/definition/english/augmented-reality*
http://www.oxforddictionaries.com/definition/english/telepresenceq=Telepresence
http://www.oxforddictionaries.com/definition/english/virtual-reality?q=Virtual+reality

Pittman, C., L. J. J. J. (2014), 'Exploring head tracked head mounted displays for first person robot teleoperation'.

Protocol, R. (2016), 'rosbridge v2.0 protocol specification'. Viewed 08.06.2016.
URL: *https://github.com/RobotWebTools/rosbridge_suite/blob/groovy-devel/ROSBRIDGE_PROTOCOL.md*

Rift, O. (2016), 'Oculus rift dk2 specifications'. Viewed 02.02.2016.
URL: *https://www.oculus.com/en-us/dk2/*

RosBridge (2016), 'Package summary'. Viewed 08.06.2016.
URL: *http://wiki.ros.org/rosbridge_suite*

Sandøy, S. (2016), System identification and state estimation for the ROV uDrone, Master thesis, Norwegian University of Science and Technology.

SNAME (1950), 'Nomenclature for treating the motion of a submerged body through a fluid', *The Society of Naval Architects and Marine Engineers, Technical and Reserach Bulletin* No. 1-5(April 1950), 15.

Thorson, P. (2016), 'Websocket++'. Viewed 08.06.2016.
URL: *https://www.zaphoyd.com/websocketpp/*

Trident (2016), 'Openrov trident'. Viewed 08.06.2016.
URL: *http://www.openrov.com/products/1-trident.html*

Valle, E. (2015), Marine Telepresence System, Thesis, Norwegian University of Science and Technology.

Windows (2016), 'Surface pro 3 specifications'. Viewed 02.02.2016.
URL: *https://www.microsoft.com/surface/en-us/devices/surface-pro-3*