Master's thesis

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

Henning Ødeby Karlsen

Towards Autonomous ROVs in Aquaculture:

Implementation of Autonomous Mission Control for Unmanned Operations

Master's thesis in Engineering & ICT
Supervisor: Martin Ludvigsen
Co-supervisor: Walter Caharija, Herman Biørn Amundsen
June 2021

NTNU
Norwegian University of
Science and Technology

Henning Ødeby Karlsen

Towards Autonomous ROVs in Aquaculture:

Implementation of Autonomous Mission Control for Unmanned Operations

Master's thesis in Engineering & ICT
Supervisor: Martin Ludvigsen
Co-supervisor: Walter Caharija, Herman Biørn Amundsen
June 2021

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

NTNU
Norwegian University of
Science and Technology

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

PROJECT THESIS IN MARINE CYBERNETICS

SPRING 2021

FOR

STUD. TECH. Henning Ødeby Karlsen

Autonomous Aquaculture:

Implementation of autonomous mission control for unmanned operations

Work Description

The current state-of-the-art technologies and operations for sea-based aquaculture farms are highly dependent on manual labour and close human interaction. In the future the fish farming industry will have to monitor and manage the production process by employing autonomous and remotely controlled robots for inspection and intervention tasks. This will reduce risk and costs, as well as improve sustainability.

For an autonomous robot to successfully accomplish a given operation it must be able to handle different tasks within the operation, as well as handle different situations that could occur during the operation. A common approach to this is using hybrid control and switching between different controllers for the different tasks and the different situations. The different controllers can be linked to different states, which can have different representations such as finite-state-machine (FSM) or behaviour trees.

The project thesis will utilize an Argus Mini ROV that SINTEF Ocean is currently deploying for testing of autonomous functionalities. It has a back-end application (FhSim C++ ACS and simulation environment) and a front-end application (Aqueous JavaScript GUI), and is extensively used for net inspection and fish monitoring purposes. The Argus Mini ROV have some autonomous capabilities but is not able to conduct autonomous operations in aquaculture.

The first objective of this project thesis is to investigate how the different states and the relations between them can be best represented and how this representation can be linked to the implementation of the different controllers and switching algorithms.

The second objective of this project thesis is to implement a mission control system for conducting autonomous operations in aquaculture.

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

Scope of work

1. Literature study on the different approaches to mission control, focusing on:
    a. Guidance, navigation and control.
    b. Reactive, deliberative, hybrid and behavior based systems
    c. Layered control architectures
2. Literature study on the use of finite state machines and behaviour trees in hybrid control, focusing on:
    a. How they can represent choices made by an autonomous vessel, such that humans can understand the reason behind the choice.
    b. How they can be linked to implementation of different states and switching logics
3. Building finite state machines or behavior trees of autonomous inspection and intervention operations executed with ROVs at fish farms.
    a. Look into how these autonomous operations are implemented.
4. Interview personnel working with ROVs at fish farms.
    a. Gain understanding of how ROV operations are currently done, and how they could be optimized.
5. Implement mission control system.
    a. Examine robustness as well as optimality.
6. Experimental work (If possible);
    a. Test the implementation in a real-life scenario.

The report shall be written in English and edited as a research report including literature survey, description of mathematical models, description of control algorithms, simulation results, model test results, discussion and a conclusion including a proposal for further work. Source code should be provided on a memory stick or similar. It is supposed that the Department of Marine Technology, NTNU, can use the results freely in its research work, unless otherwise agreed upon, by referring to the student's work. The thesis should be submitted in two copies within June 21.

Co-supervisors: Walter Caharija, SINTEF and Herman Biørn Amundsen, SINTEF

Supervisor: Martin Ludvigsen

Abstract

This thesis designs and implements an autonomous mission control system for an autonomous remotly operated vehicle (ROV) such that it is capable of performing autonomous underwater inspection, maintenance and repair (IMR) operations within aquaculture. Autonomous ROVs capable of conducting autonomous IMR operations could reduce the different risks associated with subsea operations within aquaculture as well as enable the move towards exposed locations.

For designing the autonomous mission control system, the classical Guidance, Navigation, and Control-control architecture is combined with the idea of a hybrid mission control architecture. A layered approach for implementing the different components in the hybrid mission control architecture in an existing manual system is proposed. For assisting with creating and visualizing the logic flow through an autonomous mission control system, a finite state machine (FSM) and a behavior tree (BT) is proposed for an autonomous mission control system.

The different underwater IMR operations in aquaculture and what would be required for their implementation in an autonomous mission control system is established. The different risks connected to the use of autonomous ROVs are also established.

An autonomous mission control system is then proposed for SINTEF Ocean's Argus Mini ROV, and elements of the autonomous mission control system are implemented. A previously created controller for autonomously traversing the net by utilizing a Doppler velocity log (DVL) is used as the basis for an autonomous net inspection mode. A smaller FSM was utilized for visualizing the logic flow within the net inspection mode. This system is capable of performing an autonomous net inspection in simulations but requires improved filtering and better simultaneous localization and mapping (SLAM) before it is viable in a real-world aquaculture application. The implemented elements are a reactive behavior for handling the loss of DVL measurements and a deliberative behavior for tracking the inspection progress and creating a pattern for net inspection. The reactive behavior for handling the loss of DVL measurements performs as intended both in simulation and field tests conducted in a full-scale, real-world fish farm. The deliberative behavior for tracking the inspection progress and creating an inspection pattern performed as intended in simulation; however, although showing promise, the performance was unsatisfactory in the real-world tests. The unsatisfactory performance was due to a large amount of wave and current-induced noise, a flawed filter, and a poor SLAM algorithm.

Sammendrag

I denne oppgaven utformes og implementeres et autonomt system for oppdragskontroll av en autonom ROV, slik at den er i stand til å utføre autonome undervanns- inspeksjons-, vedlikeholds- og reparasjonsoppdrag innenfor akvakultur. Autonome ROV'er med mulighet for å gjennomføre autonome inspeksjons-, vedlikeholds- og reparasjons-oppdrag kan bidra til å redusere risikoene knyttet til undervannsoperasjoner i havbruk, og muliggjøre forflytningen til mer eksponerte farvann.

For å designe systemet for autonom oppdragskontroll ble den klassiske guiding-, navigasjons- og kontrollsystem arkitekturen kombinert med ideen om en hybrid oppdragskontroll-arkitektur. En lagdelt tilnærming for implementering av de forskjellige komponentene i hybrid oppdragskontroll-arkitekturen i et eksisterende manuelt kontrollsystem blir presentert. En endelig tilstands-maskin og et atferdstre blir presentert for å gjøre det enklere å lage og visualisere logikkflyten igjennom det autonome systemet for oppdragskontroll.

De forskjellige undervanns- inspeksjons-, vedlikeholds- og reparasjons-oppdragene i havbruk, og hva som vil kreves for å implementere de i et system for autonom oppdragskontroll blir etablert. De forskjellige risikoene knyttet til bruken av autonome ROV'er i havbruk blir også etablert.

Et system for autonom oppdragskontroll blir presentert for SINTEF Oceans Argus Mini ROV, og elementer av dette blir implementert i et system for oppdragskontroll, slik at det er i stand til å utføre autonom notinspeksjon. En tidligere implementert kontroller for autonom traversering av nota ved hjelp av en Doppler velocity log (DVL) blir benyttet som basis for en autonom notinspeksjonsmodus. Systemet klarer å gjennomføre autonom notinspeksjon i simuleringer, men krever forbedret filtrering og en bedre algoritme for samtidig lokalisering og kartlegging (SLAM) før det vil være gunstig å bruke i fullskala akvakultur. De to implementerte elementene er en reaktiv atferd for å håndtere tap av DVL målinger og en overveiende atferd for å ha kontroll på fremgangen i inspeksjonen og generere et mønster for autonom notinspeksjon. Atferden for å håndtere tap av DVL målinger fungerer som tiltenkt både i simuleringer og i fullskala tester utført i et ekte oppdrettsanlegg. Atferden for å ha kontroll på fremgangen i inspeksjonen og generere et mønster for autonom notinspeksjon fungerer som tiltenkt i simuleringer, men yter ikke godt nok i fullskala tester. Årsaken til den noe svake ytelsen var mye støy grunnet bølger og strøm, et filter som ikke fungerte og en svak SLAM algoritme.

Preface

This master thesis is carried out at the Department of Marine Technology at the Norwegian University of Science and Technology (NTNU) in collaboration with SINTEF Ocean. It is the final part of the 5-year MSc program, Engineering & ICT, with specialization within Marine Technology. A pre-project laying the foundation for this thesis was conducted during the fall of 2020, and significant parts of this thesis were initially written as a part of the pre-project. This thesis looks into an autonomous mission control system for connecting different automatic functions and administering decisions for an ROV such that it is capable of performing autonomous underwater IMR operations in aquaculture.

The thesis has been based on SINTEF Ocean's framework for control of the Argus Mini ROV. The framework consists of the simulation software FhSim, which is developed by SINTEF Ocean, and the GUI Aqueous, which was initially developed by students in the course TDT4290 before the development was taken over by SINTEF Ocean. The simulation setup in this thesis is based on the setup created by Herman Biørn Amundsen for his Master thesis. The script for plotting the results in 3D was also provided by Amundsen.

The paper cited in Sections 2.1 and 2.6 was written for the specialization subject *TMR06 - Autonomous Systems* by Vilde Xiu Drønen, Hwang Jae Hyeong, and me. The paper is being considered for publication. I was the primary author of the sections *"Risk Related to Autonomy in Aquaculture"* and *"Knowledge Base"*. The paper can be found in Appendix A.

An abstract for a technical paper submitted to the OCEANS conference was accepted and a technical paper summarizing this thesis is in the making. The abstract was written in collaboration with Amundsen, Caharija, and Prof. Ludvigsen and can be found in Appendix B.

I want to thank my supervisor Professor Martin Ludvigsen for providing me with guidance and valuable feedback throughout the year. Furthermore, I would like to thank my co-supervisor Walter Caharija for sharing his knowledge of ROV operations in aquaculture and guiding me through the thesis, as well as inviting me to conduct full-scale tests at SINTEF Oceans full-scale laboratory, SINTEF ACE Tristeinen. I also owe a great deal to my co-supervisor, Herman Biørn Amundsen, for helping me with SINTEF Ocean's framework for ROV control, answering all my questions, and for reading my drafts and giving me feedback throughout the writing process; thank you! I would also like to thank SINTEF Ocean for giving me the opportunity to work with this exciting problem and for allowing me to test my implementation in a real aquaculture setting.

Due to the pandemic, the field trials were delayed, but I was lucky to get some valuable data and experience shortly before the due date. These field trials were shortened after my co-supervisor and I were quarantined. I have therefore had some tough days quarantined at a hotel in the final days of my thesis. This year has been different for most of us, and the last two semesters have been very challenging for me. I am very grateful for all the lunches and dinners I have had at Tyholt with my classmates, as they have been vital in getting me through this last year.

Contents

List of Figures

Chapter 1

Introduction

This section introduces the background for the project, presents some of the important literature studied for this project, states the objective of the project, and finally presents the structure of the report.

1.1 Background

Goal 2 of the UNs sustainable development goals is to end hunger, achieve food security, improve nutrition, and promote sustainable agriculture [47]. An important contributor to a food-secure future for all is the fisheries and aquaculture sector due to its significant and growing role in providing food, nutrition, and employment [14]. Globally, it is estimated that in 2018 about 156 million tonnes of fish were produced for direct human consumption [14]. While capture fisheries are stabilizing as they are reaching the limit of sustainable harvest, the aquaculture sector continues to rise and since 2016 aquaculture has been the main source of fish available for human consumption [14]. In Norway, the aquaculture industry has grown substantially in the last 30 years, and in 2019 Norway produced 1 440 358 tonnes of salmon and trout [46].

Figure 1.1: A typical fish farm using net cages, here SINTEF ACE Rataren located near the island of Frøya, Trøndelag, Norway. (M.O. Pedersen, SINTEF Ocean AS) [3]

The lack of available sheltered locations for new aquaculture production facilities along the Norwegian coast and the necessity to preserve some of the natural landscape is pushing the Norwegian aquaculture industry to move towards more exposed and remote sites. An example of a fish farm in moderately sheltered waters can be seen in Figure 1.1. Moreover, further offshore, the production conditions are potentially better since the farms experience less negative consequences from sea lice [19]. The environmental impact is also reduced due to a greater distance to wild salmon [19]. For the personnel working in aquaculture, the move from sheltered waters to more exposed areas also means a higher risk of injuries and fatalities. The aquaculture industry is already one of the most dangerous occupations in Norway [21], meaning that these risks should be reduced as much as possible, as soon as possible. The current technologies and operations in fish farms are highly dependent on manual labor for tasks such as cleaning and maintenance. This leads to close human interaction with tools and fish cage structures. Typical operations that are currently performed manually for each net cage are the removal of dead fish as well as inspection, maintenance, and repair (IMR) operations. [50]. Increased use of Remotely Operated Vehicles (ROVs) for these purposes will create a positive chain reaction, reducing risk and increasing efficiency. ROVs can, to some extent, replace diving operations, which are among the operations in aquaculture with the highest risk [41].

Large actors in the oil and gas industry have been looking into the use of ROVs for IMR operations for over a decade [38]. In the last decade, the use of ROVs in aquaculture has increased, and it seems to continue increasing in the future. The use of ROVs in aquaculture can be more challenging compared to the oil and gas industry, and there are several reasons for this. The aquaculture net pens are flexible structures, and knowing their exact position and movement is challenging. Collisions between the ROV and the net must be avoided as it may weaken the net and ultimately cause a hole, hence causing farmed salmon to escape the cage. Controlling the ROV is also more challenging because the aquaculture net pens are situated in the wave zone, causing larger forces from waves than those experienced by ROVs deep under the surface. Lastly, the large, dense biomass causes challenges for acoustic signals and visual references that the ROV uses for communication and navigation. [3] Remotely Operated Vehicles in the context of marine application usually refer to a box-shaped, tethered, unmanned vehicle used for different underwater operations such as detailed inspection, subsea installation, and maintenance. In [44], NORSOK defines three different classes of ROVs which are presented in Table 1.1.

Class I	Pure observation	Small vehicles with cameras
Class II	Observation class	Vehicles able to carry additional sensors
Class III	Work class vehicles	Vehicles large enough to carry sensors and tooling

Table 1.1: Classes of ROVs from [44]

All of these classes can, to some extent, be found in aquaculture [3]. Class I ROVs are only used for simple visual inspection; in aquaculture, this could be an inspection of the net, examining fish behavior, etc. A Class II ROV differs from Class I by having additional sensors as payload, and some could also have lights and simple tools. As they should still be relatively small, they have limited abilities, but due to the ability to carry additional sensors, they could perform advanced inspections. They could also possibly be used for tasks

such as net repair, retrieval of mortalities, and other tasks that only require simple tools. Class I and Class II ROVs can be found quite commonly in aquaculture. Class III ROVs are not as commonly found in aquaculture due to their size and cost. They do, however, have some more advanced uses, such as washing net cleaning and subsea drilling for anchoring bolts.

During the aforementioned operations, the ROV is usually manually controlled by an operator on board the service vessel that launched the ROV. While the aquaculture environment is complex, several of these tasks themselves are fairly simple, and could with some technological innovation, be performed by an autonomous ROV. An example of this is presented in [4], where an ROV is outfitted with a Doppler velocity log (DVL) to estimate the position of the net relative to the ROV, which then traverses the net autonomously. Figure 1.2 depicts the mentioned ROV performing autonomous net traversal. This is an important step in the direction of autonomous ROVs in aquaculture.

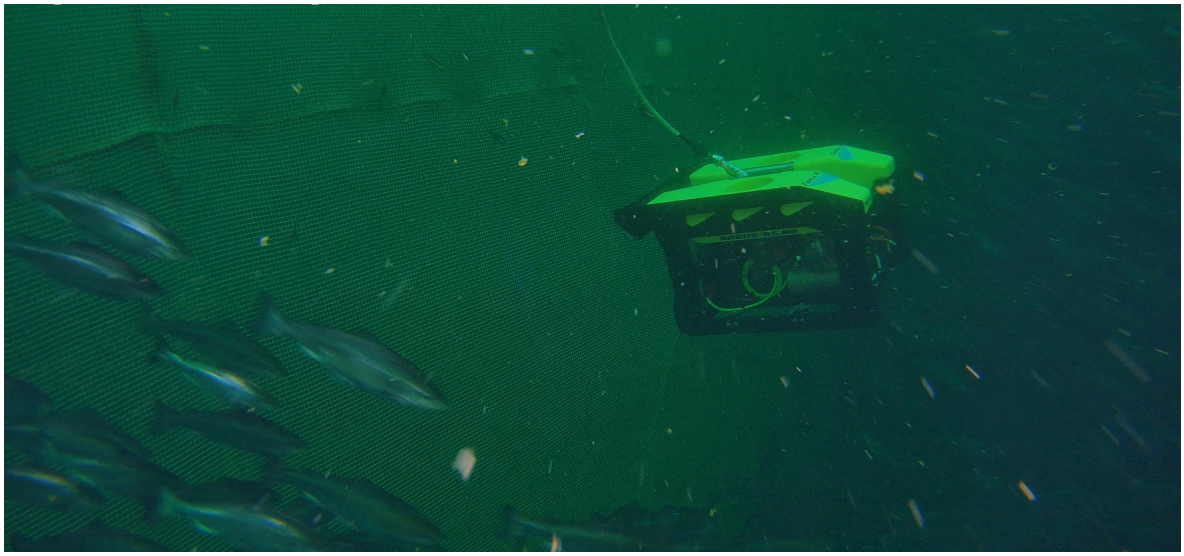

Figure 1.2: SINTEF Ocean's Argus Mini ROV inspecting the net cage. [3]

The objectives for automating different functions in ROV operations are related to safety, performance, consistency, time, and cost [13]. Today, it is common for ROVs in aquaculture to have some automatic functions such as automatic depth control and hovering, which allows for limited autonomy. A few more advanced ROVs have capabilities such as station keeping, which allows for some more advanced autonomous functions. Further research and technological innovation are required before ROVs in aquaculture can be fully autonomous.

1.2 Objective

This project looks into the mission control architecture for connecting the different automatic functions and administering decisions for performing autonomous IMR operations. Therefore, the objective is to design and implement a mission control system for an ROV such that it can perform selected IMR tasks within aquaculture autonomously.

1.3 Report Structure

- Chapter 1 describes the background and objective of this project.
- Chapter 2 introduces the theory required for designing the mission control architecture of an autonomous ROV.
- Chapter 3 describes where and how different IMR operations could be implemented for the Argus Mini ROV owned by SINTEF Ocean.
- Chapter 4 describes the iterative implementation process of selected IMR operations.
- Chapter 5 documents the results from field trials using the final implementation of the mission control architecture.
- Chapter 6 discusses the results and challenges that occurred during implementation and field tests and how these could be solved.
- Chapter 7 concludes the project and presents opportunities for further work.

Chapter 2

Theory

This chapter will introduce the theory required for designing the mission control architecture of an autonomous ROV. Most of the sections in this chapter were researched and written during the pre-project, and forms the theoretical foundation for the developed control architecture.

2.1 Underwater IMR in Aquaculture

To determine what tasks and goals should be included in the mission control architecture for an autonomous ROV in aquaculture, one must first have good knowledge of the real-world tasks it should be able to complete. This section will therefore take a look at what IMR-operations an autonomous ROV could perform. This section is largely based on a paper written for the specialization subject "TMR06 - Autonomous Systems", talks with personnel from SINTEF Ocean working closely with ROVs in aquaculture [3] and an analysis of current ROV operations in the norwegian aquaculture, [41]. The paper was written by the author in collaboration with fellow students at the Department of Marine Technology at NTNU, Vilde Xiu Drønen and Hwang Jae Hyeong, and can be found in Appendix A.

2.1.1 Inspection

The net cage containing the fish is among the most crucial structures in aquaculture. The risk with the most significant consequences in aquaculture, for both the environment and for the asset, is the risk of fish escapes. The most frequent reason for fish escapes is structural failures, such as mooring failure, breakdown of net cage structures, abrasion, and tearing of nets [23]. An important tool in avoiding these structural failures is regular inspection. Several of the structural parts which need regular inspection can be found underwater. In the last couple of decades, the use of manually controlled ROVs for structural inspection operations has gradually replaced the divers, lowering the risk of injuries and fatalities [41]. Inspection can also be an important tool for monitoring fish welfare. During delousing or other operations where the fish is forcefully moved (crowding), the fish is monitored for determining the effects on the fish and if the operation affects the welfare of the fish. This monitoring is usually conducted using a manually controlled ROV. Onward in the future, it can be expected that the manually controlled ROVs could be replaced with autonomous ROVs. Three underwater inspection operations where manual ROVs are currently used are net inspection, mooring

inspection, and fish monitoring.

Net inspection

The main component of a net cage is the net itself, and as mentioned above, holes and breakages in the net are the most common reason for fish escapes. Holes are typically caused by predators or caged fish biting the net, abrasion, collisions with boats, and handling procedures such as lifting the net. To avoid fish escapes the nets must be inspected such that the holes can be found and repaired. These inspections are scheduled regularly and are also usually conducted after handling procedures. When the inspection is completed with an ROV, the standard procedure is to pilot the ROV around in the net cage while pointing the ROVs camera at the net. The distance between the ROV and the net is usually around 2-3 meters from the net, but could be as low as 0.5 meters in low visibility conditions [41]. As parts of the inspection will take place in the wave zone and the net cage contains other structures and moorings, there is a risk of the ROV colliding with the net or getting the tether stuck in some other structure. Avoiding collisions and maneuvering in a way that keeps the tether free requires careful control of the ROV. While the ROV is piloted around in the net, the operator monitors the video feed showing the net and identifies and reports deviations such as holes, gnawing, misplacement, and other defects. The video feed is prone to disturbances, such as low visibility, fast-moving currents, and fish coming between the camera and the net. For this operation to be fully autonomous, it would require autonomous piloting of the ROV and automatic detection of deviations. The use of a Doppler velocity log (DVL) to autonomously track the net was first proposed in [36]. A method for guiding an ROV to autonomously traverse the net by the use of a DVL is presented in [4]. Automatic detection of holes has been researched and developed by SINTEF, but the algorithm is in its infancy [20]. Based on the results in [4], autonomous traversal of the net pen is feasible with the use of a DVL, although some further research is needed to achieve better control of the ROV and better filtering of noise and errors in the DVL distance measurements. This could be combined with further development of the automatic hole detection algorithm for an autonomous net inspection. However, the autonomous net inspection will not be fully autonomous until the traversal, and especially the detection system, can be completely independent of humans.

Mooring inspection

The moorings are related to the structure keeping the net cage in place. They consist of anchor points, mooring chains, mooring ropes, and the joints between them. The different components can have different failures, leading to the complete failure of a mooring which again could lead to a complete breakdown of the net cage. While a complete breakdown of the net cage is uncommon, it leads to the largest amount of escaped fish [23]. Inspections of the moorings can help discover possible faults and prevent complete failure, and regulations state that inspections should be conducted at least twice a year. ROVs usually conduct these inspections as the mooring inspection is a time-consuming operation, and some of the anchors could be located in deeper waters, making the inspection operation unsuitable for divers [41]. When the inspection is conducted with an ROV, the standard procedure is to place the ROV in the water, outside of the net cage, and then follow a mooring line from top to bottom. It can be challenging to control the ROV as it will be located in the wave zone for a part of the operation. The location of the tether must also be considered to avoid wedging or damaging

it. While the ROV is following the mooring line, the video feed is monitored for deviations in the mooring line that could lead to a failure. As mentioned in the net-inspection section above, this video feed is prone to disturbances, making it harder to detect deviations. The most common deviations are ruptures or overlaps. Other deviations can be ropes being in contact with other ropes or the bottom, rocks, and other hard surfaces that could rub or tear the rope. These deviations will be reported and then patched up. General wear and tear of the chain, rope, and joints can also be detected and used for planning future inspections or reparations. A method for localizing a flexible anchor line was proposed and tested in [20,37], and while it, in combination with an autonomous ROV, could be used for performing mooring inspections autonomously, further research is required for this to be feasible. More research has been conducted regarding detecting faults in ropes and chains, but again further research is required as most of the research has been for ropes outside of water. For the mooring inspection to become fully autonomous, the ROV must be able to perform as well as the human operators do today or better. Hence, the detection system must be able to recognize more than signs of rope and chain failure.

Fish Monitoring

The fish in Norwegian fish farms is protected by the Animal Welfare Act [27]. To avoid high-stress levels in the fish during handling operations that utilize crowding, such as delousing, the operation and the movement of the fish is monitored. In most handling operations utilizing crowding, a crowding net is placed inside the net cage and shrunk in order to force the fish into a desired area. ROVs are often used for inspecting fish behavior during these operations. The ROV is placed inside the net cage but outside of the crowding net. It then follows the movement of the fish inside the crowding net, and a video feed is used to monitor the behavior of the fish. The ROV operators relay the information from the video feed to the personnel conducting the handling operation, which then adapts the procedure according to the given information. For this operation to be made autonomous, it would require a control system that can follow the shrinking of the net, as well as the movement of the fish. Secondly, it would require a computer vision system able to determine how stressed the fish are. To the best of the author's knowledge, both of these requirements are within an area where little research has been conducted, meaning that a large amount of research must be conducted before this operation can be made autonomous.

2.1.2 Maintenance

Maintenance work is performed to minimize the possibility of something going wrong or no longer functioning as desired. It is also essential to maintain the primary function of several components. The various pieces of the net cage require different maintenance operations. The tension of the moorings must be checked and adjusted, and the moorings and net must be cleaned from biofouling. The only maintenance task commonly performed with ROVs is the operation of keeping the net clean from biofouling. This can be conducted with special cleaning rigs, either manually operated by cranes, ropes, and humans, or by an ROV. The benefit of using an ROV for this operation is reduced risk for workers and increased efficiency. While other underwater maintenance operations may be required, these are rarely conducted with ROVs as these operations often have precision requirements that are hard to satisfy with ROV grippers. The primary focus of this subsection will therefore be net cleaning.

Net cleaning

There are several reasons why the net should be kept clean, most of them related to the welfare of the fish [7]. First, the biofouling reduces the oxygen level and waste flushing in the net cage due to reduced water circulation in the cage. Second, the biofouling could contain parasites such as louse, which could damage the fish or introduce diseases. Lastly, the weight of the net can be greatly increased with biofouling, increasing the risk of structural failures. Nets are often covered with a layer of antifoulants to reduce biofouling, but this does not last forever, and biofouling will occur. Hence, cleaning is needed to remove the biofouling. There are different ways of cleaning the net; currently, the most common method is spraying the net with high-pressure water from a minimal distance. Cleaning methods utilizing brushes can also be found. While more abrasive methods can be more efficient in cleaning the net, they can cause more wear and tear of the net, as well as the release of chemicals from the antifoulants, such as copper. As a result, they are not as common. During cleaning operations with an ROV, the ROV is placed inside the net cage and then maneuvered around, touching the net, while a cleaning tool cleans the biofouling from the net. There exist both purpose-built ROVs for net cleaning and specialized cleaning rigs that can be mounted to work class ROVs. An example of a purpose-built ROV for net cleaning can be seen in Figure 2.1, with eight rotating discs spraying high-pressure water at the net.

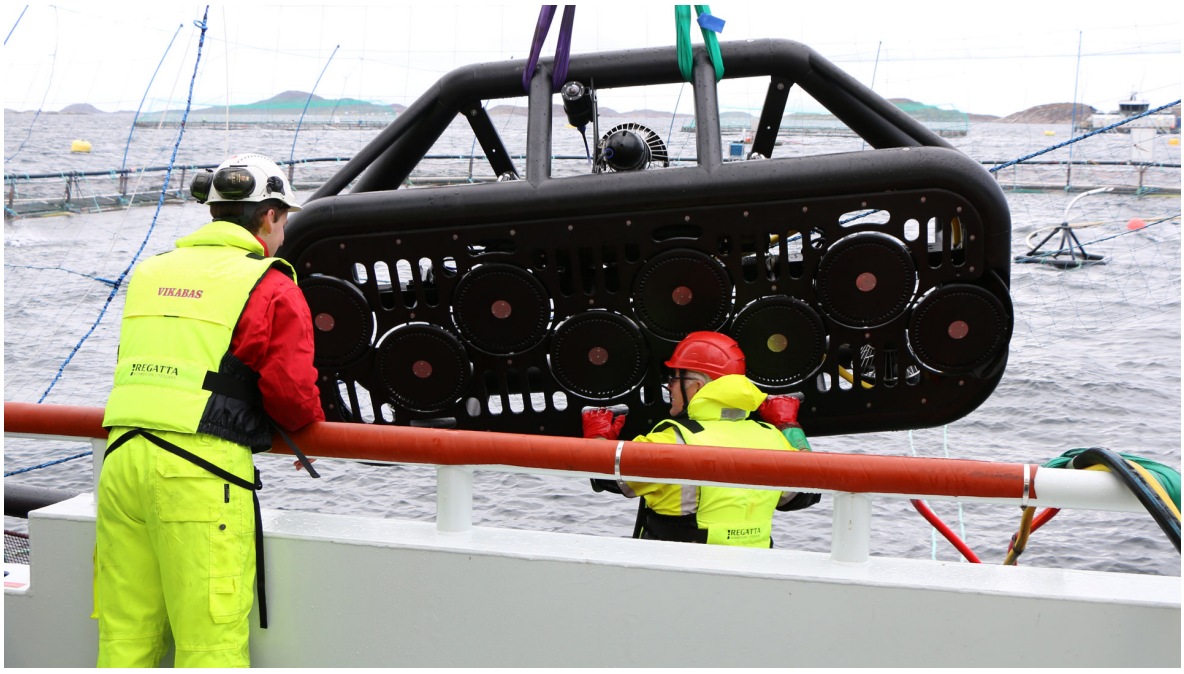

Figure 2.1: The purpose-built net-cleaning ROV *FNC8 2.0* from AKVA group, produced by Sperre AS [1]

The *FNC8 2.0* is also "*prepared for autonomous operations*" [1]. It could also be feasible for general work class ROVs with cleaning rigs to be made autonomous. As the ROV will be interacting with the net while cleaning, it runs a high risk of damaging the net should something go wrong. Therefore, it is crucial that the ROV does not apply too much pressure to the net and that any sharp edges or protruding objects are removed. For autonomous operations,

an ROV would need a positioning system relative to the net, a precise control system, and some system for determining whether the desired area of the net has been sufficiently cleaned. Researchers at SINTEF Ocean have conducted research on the specifications of a permanent resident, autonomous, and tetherless subsea robot for cleaning and inspection of net cages in [29]. They state that no existing solution is able to fully address the challenges they found for cleaning and inspection of net cages. Akva group and Sperre AS state that their *FNC8 2.0* is prepared for autonomous operations, but the author could not find any examples of this being tested or in use. Therefore, it can be assumed that they have a capable system that requires more research and work before being ready for implementation.

2.1.3 Repair

Repair operations are essential for avoiding massive consequences, such as large-scale fish escapes. Several of the components of a net cage are large and expensive, and they are therefore easier and more economical to repair instead of replacing them. The net is one of the components in aquaculture that is most exposed to damages and hence needs the most repairs. It is also one of the most crucial components of aquaculture. Therefore, the most important underwater repair operation which can be conducted by an ROV is the operation of repairing a hole in the net.

Net Repair

As earlier stated, the formation of holes is the most common reason for fish escapes in Norwegian aquaculture [23]. Even a small hole could lead to a larger rip, meaning that it is of great importance to repair any holes that might occur in the net. The holes should also be repaired as quickly as possible after being discovered. Holes are usually sewn back together by divers. This is to achieve a repair of sufficient quality. There have been proposed some solutions for repairing holes in the net with ROVs, but they are not widely used [41]. An example is the sewing-machine tool produced by Sperre AS [42]. Using ROVs would greatly reduce the risk for the divers; however, the quality of the repair must be satisfying for it to be feasible. The use of ROVs could also increase the risk of expanding the hole, as the ROV needs to be placed close to the hole with a repair tool such as a sewing machine protruding. As holes should be repaired as quickly as possible, it could be beneficial if an ROV was able to both conduct an inspection and perform a repair within the same operation. Before this operation could be made autonomous, there would need to be a feasible solution for repairing holes with an ROV. This could be a sewing-machine tool like the one from Sperre AS, but to the best of the author's knowledge, it lacks proper use in real-world application. Assuming a solution exists, an autonomous ROV with systems for performing net repairs would require several advanced systems. A precise controller for keeping stationary while conducting repairs would be required. It should also be able to locate and move to the location of the hole autonomously. The ROV would also need an autonomous repair system, which must be able to identify the edges of the hole and how it can be repaired and then perform the reparation.

2.2 Control Architecture

A mission control system for achieving the tasks presented in section 2.1 will have different systems interacting to achieve the desired motion. These systems will be combined in some control architecture. In regards to the control architecture for marine craft, *Handbook of Marine Craft Hydrodynamics and Motion Control* (Fossen, 2011) [15] proposes a "Guidance, Navigation and Control"-architecture, often referred to as "GNC". This architecture is among the most common approaches to vessel control, and it can be found implemented in a large number of practical applications. The approach in (Fossen, 2011) [15] is based on the classical approach to a control problem, but expanded with a guidance system for providing an improved reference to the control system, and a navigation system for determining the position of the vessel. A figure showing the three interconnected systems can be seen in Figure 2.2. These will be further explained in the following sections.

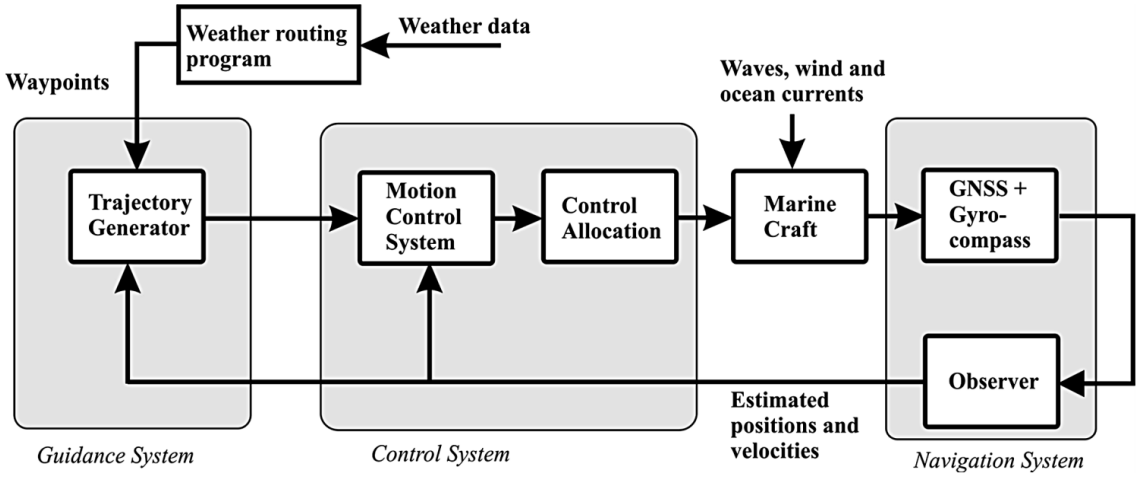

Figure 2.2: The three interconnected subsystems of the GNC-architecture, reprinted from (Fossen, 2011, p. 233) [15]

2.2.1 Guidance

The goal of the guidance system is to calculate reference signals for the controllers in a control system. In the simplest form, the reference could be the goal position, but by improving the reference signal, a more efficient and precise result could be achieved from the controller. Path following is an example of this, where the guidance system calculates a path to the end goal and sets a point along this path as the controller set point instead of setting the end goal as the setpoint.

2.2.2 Navigation

The navigation system estimates the current state and position of the system, usually from a combination of data provided by different sensors and observers. The resulting estimate is used both by the guidance system for calculating the reference, and by the control system for tracking of the reference signals.

2.2.3 Control

The control system determines the necessary forces to achieve a goal. This desired goal is calculated by the guidance system. The control system will also include the allocation problem if a system has several ways of providing the necessary forces.

2.2.4 Expanding GNC for autonomy

The approach in (Fossen, 2011) [15] covers the solution of classical control problems in the area of marine control. To introduce autonomy into this approach, the guidance, navigation, and control systems must be expanded with more advanced and deliberative functions compared to what is commonly found in marine control. When looking into a mission control architecture for an autonomous ROV, the guidance system will be the most relevant part of the GNC-architecture. The other systems could also be made to include more advanced and deliberative components, but this will not be covered in this thesis.

2.3 Mission Control Architecture

Controlling an autonomous vessel is a complex task, and a well-conceived control architecture can help manage this complexity. The autonomous vessel must be able to perform different tasks and at the same time react to unexpected situations [40]. A mission control architecture contains control laws, error detections, recovering, path planning, task planning, and monitoring of events during mission execution [16]. The design of a control architecture is one of the most important elements when developing the autonomy of a vehicle [10]. The design of a mission control architecture is no exact science, there is no right or wrong answer, but various control architectures have different advantages. Four different classes of methods for robot control are presented in the *Springer Handbook of Robotics* [40]. These represent different ideas and does not directly relate to the control architecture itself, however, the different ideas lay a foundation for building the mission control architecture.

- Deliberative - Think, Then Act
- Reactive - Don't Think, (Re)Act
- Hybrid - Think and Act Concurrently
- Behavior-Based Control - Think the Way You Act

These will be further explained in the following subsections.

2.3.1 Deliberative

The idea of deliberative control is that the robot uses all the available information from sensors combined with previous knowledge to reason about what the next action should be. This reasoning is typically a form of planning, where the system searches for the possible action sequences and their outcomes. The system must then evaluate all the outcomes and choose the one that achieves the goal or solves the task optimally. Planning is a significant component in artificial intelligence and can be a complex process. An example of planning can be path planning for a robot that is navigating through some environment. To generate

the optimal plans the system must be able to look into the future and predict the outcomes of possible actions. This means that the system must have an internal model of the environment, often called a world model, that must be as accurate and up-to-date as possible for enabling the system to choose the best action in any given situation. If the environment is a dynamic, rapidly changing, and noisy environment, it can be challenging to keep the world model accurate and up-to-date, especially in real-time. This gives purely deliberative robots a slow reaction time, and as a result, it is rare to find purely deliberative robot systems today. [39,40]

2.3.2 Reactive

To allow the system to quickly respond to a dynamic, rapidly changing, and noisy environment, one could apply the idea of reactive control. In reactive control, there is no intervening reasoning, and instead, there are simple rule-based methods preprogrammed into the system, enabling rapid real-time responses. These rule-based methods can often be represented as finite state machines, where the rules represent the state transitions. This allows the system to quickly react to events as they happen. Examples of this can be object avoidance for a robot with limited vision or reacting to sensor errors. Reactive control trades the complexity of reasoning for a faster reaction time. For purely reactive systems, no information about the state of the world is stored, and the overall system is limited to simple tasks with low complexity. This makes it very challenging to design a robot that is capable of accomplishing long-term goals or acting optimally. As a result, most robot systems today have some internal model of the environment and plan for completing long term goals. [39, 40]

2.3.3 Hybrid

Hybrid control aims to combine the best aspects of the reactive- and deliberative control ideas. This allows the system to utilize the rationality and optimality of deliberative control, as well as the quick reaction time of reactive control. A common method for achieving this combination is to let different subsystems manage the different control ideas. A reactive subsystem will use preprogrammed rules to enable reactive actions, i.e., evasive maneuvers for collision avoidance, while a deliberative subsystem will manage the overall plan, goals, and challenges. The control system must then use some algorithm to decide which of the actions provided by the subsystems it should use. In the simplest form, this algorithm could be to always give the reactive control priority and then have a "no-action" reactive action to let the deliberative system through. However, this would highly rely on well-created reactive actions, and a more advanced algorithm could be beneficial for more efficiently reaching the goals. One example where a more advanced algorithm would be required is if a situation evolves from a situation that would generally be managed by the reactive subsystem into a situation that must be managed by the deliberative subsystem. Another example could be where the deliberative subsystem must influence the reactive subsystem such that it does not entirely digress from the overall goals. I.e., the system must make an informed decision of what to do. The design of a controller using a hybrid mission control architecture requires good knowledge of both the main goals and the main pitfalls for the controlled system, whether it is a robot trying to find its way through some environment or an ROV performing IMR operations in aquaculture. [39, 40]

2.3.4 Behavior-Based

The idea of behavior-based control divides the mission control problem into different low-level control behaviors. These behaviors are implemented as control laws and can all take input from the sensors and other modules, as well as send outputs to the actuators and other modules. The behaviors range from the most straightforward reactive behaviors to more advanced behaviors capable of modeling the environment. All the behaviors are run concurrently and interact mainly in the environment instead of internally in the system. The combined output from all the behaviors should lead to the system reaching its overall goals. A significant challenge for behavior-based control is the coordination algorithm needed to achieve this. Designing this algorithm could be more challenging in behavior-based control compared to hybrid control. It could also result in reduced efficiency or analyzability of the resulting control system. However, it could be more suited to the application of machine-learning-based methods such as reinforcement learning [39]. Another challenge with behavior-based control is that it rarely fits into pre-existing systems but instead requires the entire control system to be designed around it. [39, 40]

2.3.5 Choosing a Suitable Mission Control Architecture

The idea of reasoning and deliberative control is one of the key components for making a system into an autonomous system. However, it is not as simple as just using the idea of deliberative control to design a mission control architecture. An autonomous ROV in aquaculture is subject to a complex environment. It has to cope with large moving biomass, relatively large environmental forces from waves and current, and flexible structures moving with the aforementioned environmental forces. Hence, a reactive component will be needed to react to unplanned events and avoid unwanted situations. When choosing a suitable mission control architecture for control of an autonomous ROV in aquaculture, a combination of these would therefore be appropriate.

As ROVs are rarely purpose-built to be autonomous, they already have a control system for manual control with some automatic capabilities. This control system will usually have an architecture similar to what is presented in (Fossen, 2011) [15], and this is a well-known approach for marine control systems. Hence, it could be beneficial to expand on this system instead of re-doing the entire control system from the bottom up. This will fit more within the idea of a hybrid mission control architecture compared to the idea of behavior-based control. Another reason for choosing a hybrid architecture is that it is easier to design and understand. A mission control architecture based on the idea of behavior-based control is more suitable for multi-robot systems and other systems where scalability is more important. For the case with an autonomous ROV for aquaculture, there is only one ROV and no need for scalability.

Further in this project, the mission control architecture will be based on the idea of a hybrid mission control architecture.

2.4 Architectural layers

A common approach for dividing a mission control system into various subsystems is to let different layers handle different parts of the control problem. One layer would handle planning

and control calculations, while an execution layer would typically interact with the hardware and execute the desired control actions. In the layered approach, the reactive ideas are included in the execution layer, while the deliberative ideas are implemented into the overhead control layer. There could also be a layer solely for communication and coordination between the other layers. The actual layers used in different mission control systems can widely differ and are usually both software and hardware dependent.

Hybrid control can significantly benefit from being implemented in a well-structured, layered control architecture. A layered control architecture has the benefit of being easier to understand, more reusable, and easier to test and validate. Several layered architectures for robot control are presented in [40]. One of them is a three-layered architecture, with a planning layer, an executive layer, and a behavior control level.

2.4.1 Planning Layer

As the name suggests, the planning layer takes care of planning the long-range goals of the mission. It should be looking towards the future and determine the optimal plan for achieving all the goals. If the system is unable to achieve the desired goals, it should be able to replan to accomplish the goals. The planning layer can also include specialized planners, such as path planners and resource optimizers, for efficiently and optimally solving particular planning problems.

2.4.2 Executive Layer

The executive layer is the interface between the planning layer and the behavior control layer. It translates high-level plans into lower-level behaviors and instructs the behavior control layer on when the different behaviors should be executed and which behavior should be prioritized over the others. The executive layer will monitor the execution of the low-level behaviors, and if the execution fails, it will handle the exception. Some of these exceptions could require the planning layer to replan the mission, while others could be solved by calling another behavior. It can also manage constraints between different behaviors.

2.4.3 Behavioral Control Layer

The behavioral control layer represents the lowest level of control. While the name matches the mission control architecture idea of behavioral control mentioned in Subsection 2.3.4, the behavioral control layer is not entirely based on the idea of behavioral control. The behavior control layer refers to behaviors as the different modes sensors and actuators can be in and the tasks they can perform. Hence, the behavioral control layer is the layer that interacts with the sensors and actuators of the system. It relays the information detected by the sensors up to the higher layers and receives back instructions from the higher layers to execute with the actuators. This layer primarily consists of classical methods from control engineering, with each behavior solving one specific task.

2.4.4 Implementation in Practical Systems

Up to this point in this section, there has been little consideration for how these layers are to be implemented in a practical system. In a practical system, one needs to consider the different

hardware and software architectures and how they interact. All practical systems that rely on electronics for control will require some hardware interfacing software for interacting with the hardware. Usually, several different software could interact with each other before feeding the resulting instructions to the hardware interfacing software. The hardware interfacing software then ensures that the hardware performs the tasks it is given. It could seem like this would suit the layers approach well, but for practicality, this software is rarely built with these architectural layers in mind. Hence, the layers in the practical system could vary significantly from what is presented in the theory. Usually, the theory would be altered to fit the system when being implemented. However, some of the main components in the theory could still be found in the practical system. One separation that would usually follow the theory is the separation between the executive/planning layers and the behavior control level. As stated above, the behavior control level will be similar to a classical control system and can often run on the hardware of the system to be controlled, while the executive layer and the planning layer will run on some software that the human operators could monitor and interact with.

2.5 Dynamic Mission Management Specification

Missions in underwater robotics are often quite advanced and require several different behaviors in the behavioral control layer. The logic for deciding what behavior to run and switching between these behaviors can often be hard to keep track of and challenging to create in the first place. Therefore, mathematical models of computation can be a valuable tool for creating and visualizing the logic behind the control flow [11,22]. Mathematical models are well suited to serve as the executional layer, but they could also be utilized within the other layers. Two such mathematical models are Finite State Machines (FSM) and Behavior Trees (BT).

Finite State Machines originated within computational mathematics in the 1940s, and research on FSMs was conducted into the 1980s. Today, most of the active research within the area of FSMs has shifted away from the mathematical definition of FSMs and over to the application of FSMs [22]. FSMs can be found in several fields of control, such as flight control systems, motion controllers, constrained robotic systems and computer hardware [9]. They have also been used for several of the autonomous vehicles competing in the DARPA Grand Challenge [28,33,48,49] and are common within other autonomous vehicles as well [3]. MathWorks also develops a control logic tool called Stateflow, which uses state transition diagrams, a variant of FSMs, for designing control logic.

Behavior Trees were introduced in the late 2000s and were first developed for use in the control of Non-Player Characters (NPCs) in computer games [11]. Previously, the control of NPCs in computer games was commonly done with FSMs, but they are becoming less common as they are replaced by BTs. The computer game industry is an industry with a high pace of development, and BTs are to some extent already being replaced by utility-based AI [34]. In academia, BTs have been a subject of research for the last ten years, with robotics being one of the fields where BTs have shown potential [11,31].

One of the main advantages of both FSMs and BTs is the ability to represent them graphically with relatively little effort. This makes the logic easy to follow and increases the readability and understandability for both developers and operators.

In subsection 2.5.1 and subsection 2.5.2 these two different methods will first be presented and then compared with regard to their use in underwater robotics.

2.5.1 Finite State Machines

Finite State Machines can be conceptualized as an abstract machine that can only be in one out of a finite number of pre-defined states. To enter a different state a transition is performed; this transition will only take place if the corresponding condition is true. Once in a new state, some action can be performed until a condition for switching states is true, which will cause the FSM to proceed to the next state.

This section will present some examples of FSMs, such that the reader should be able to grasp the basic concepts should they be unfamiliar with FSM. The examples in this section are adapted from *Elements Of Robotics* [8]. A more in-depth description of FSMs, as well as several examples within embedded systems, can be found in *Handbook of Networked and Embedded Control Systems* [9] and the formal definition and more indepth information can be found in *Introduction to Automata Theory, Languages, and Computation* [22].

The basis for the mathematical model in FSMs is the concept of states and transitions. The concept that something can have different states that can be changed with some transition is familiar for most people. A basic example of this is appliances that can be either on or off, such as a toaster. The states of the toaster will be "on" and "off," and for switching between these states, some transition must be taken. For a toaster, there will be two transitions, one from "on" to "off" and one from "off" to "on." These transitions must consist of a condition and an action. The condition causes the transition to be taken, and the action is performed when the transition is taken. Continuing with the toaster example, the conditions can be the position of the lever and the value of the timer. For the transition from "off" to "on," the condition will be that the lever is down, and for the transition from "on" to "off," the condition will be that the timer has expired. The actions will be to either turn the heat on or off, corresponding to the state of the toaster. A finite state machine also has a designated initial state. The toaster is initially in the "off" state when plugged in. The states and transitions with conditions and actions can easily be visualized with a graph instead of in writing. In the graph, each state is represented by a node and each transition by an edge. The conditions and actions are labeled on the transitions. The initial state is represented by an edge that is only connected to one node. A diagram of the FSM for the toaster can be seen in Figure 2.3

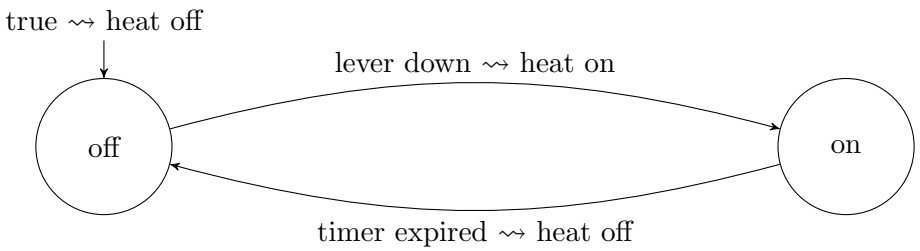

Figure 2.3: Finite State Machine Toaster Example

The example in Figure 2.3 is among the most straightforward finite state machines. The Finite State Machines found in the field of robotics can be quite complex. A simple example from robotics can be a primitive wheeled robot searching for objects. In this example, the robot is supposed to drive in a zig-zagging pattern until it finds an object. It should then approach the object and stop right next to it. The robot has four different states:

Turning Left	The robot turns left while driving forward
Turning Right	The robot turns right while driving forward
Approach	The robot moves towards a detected object
Found	The robot stops near the object

Table 2.1: The four states of the primitive wheeled robot

The robot is constantly searching in the turning states. When the robot reaches $\pm 45^o$ from the heading it started with, it will start turning in the other direction. If the robot detects anything along its path, it will approach the object and stop near it.

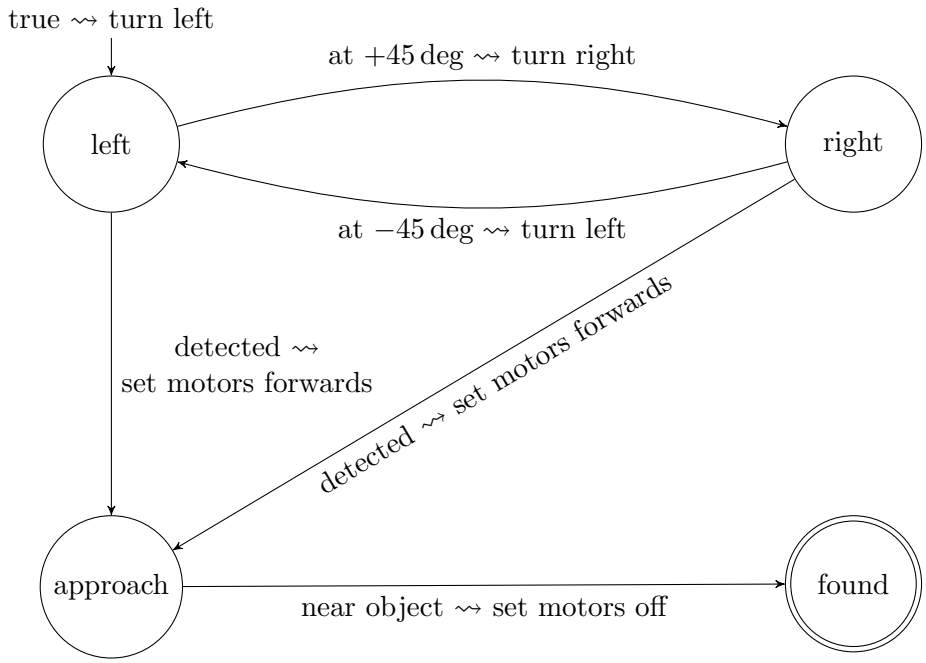

Figure 2.4: Finite State Machine Example

Constructing the diagram seen in Figure 2.4 can assist in both the implementation of the control system as well as identifying failures during runtime.

FSMs can be used in different parts of the mission control system. Most notably, it can be used to plan and execute the mission by combining the different higher-level behaviors to achieve the overall goal. It can also be utilized within the different high-level behaviors for tying together different low-level behaviors and achieving the goal of the high-level behavior.

Implementing an FSM is relatively easy in a single-software, single-layer control system.

However, when implementing an FSM in more advanced systems, the implementation can be divided between different software. This might not directly correspond with the layered approach, but the overall concepts from finite state machines and the layered control approach still apply.

FSMs are widely used due to being easy to understand and implement, as well as being a common structure within computer science [11].

2.5.2 Behavior trees

A behavior tree is a rooted tree, where the internal nodes are called control flow nodes, and the leaf nodes are called execution nodes. There are four different types of control flow nodes; Sequence, Selector, Parallel, and Decorator, and two different types of execution nodes; Action and Condition. Behavior trees are executed in ticks with a given frequency; for each tick, the tree is traversed in pre-order (depth-first). When an internal node is ticked, it will route the tick further down the tree by ticking its children and then return either "Success," "Running," or "Failure" to its parent node depending on the outcome of the child nodes. A leaf node will either perform some computation or action depending on the type of node and then also return either "Success," "Running," or "Failure." The different types of nodes will be further explained in the upcoming subsections, and a behavior tree will be presented for the same robot example as in subsection 2.5.1. More information about behavior trees, pseudocode, and several examples can be found in *Behavior Trees in Robotics and AI - An Introduction* [11].

Sequence Node

A sequence node will tick its children from left to right. If a child returns "Failure" or "Running," the sequence node will return the same and not tick the next child. It will only return success if all its children return success. The sequence node is commonly represented in the diagram by a square containing an arrow. A representation of the sequence node in a diagram can be seen in Figure 2.5.

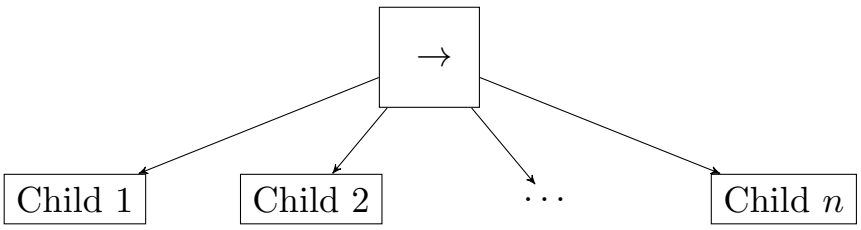

Figure 2.5: Sequence node with n children

Selector Node

The selector node[1] starts by executing its children from left to right and returns "success" as soon as one of the children returns "success." If a child node returns "running," the sequence

[1] Also called the fallback node

node will also return "running" and not tick any remaining children. If none of the children returns "success," the selector node will return "failure." The selector node is commonly represented in the diagram by a square containing a question mark. A representation of the selector node in a diagram can be seen in Figure 2.6.

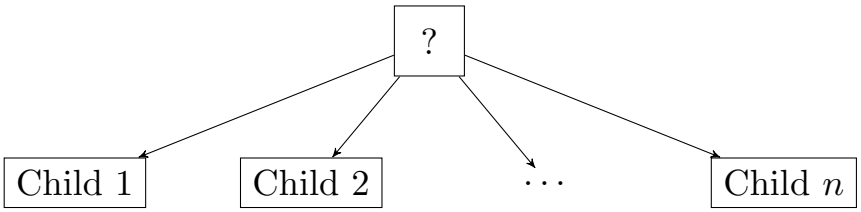

Figure 2.6: Selctor node with n children

Parallel Node

The parallel node ticks all its children simultaneously and will only return success if a pre-defined number of children succeeds. If most child nodes are still running and the parallel node cannot decide an outcome, it will return running. Failure is returned if the number of child nodes still running plus the number of child nodes that have already returned success are not enough to achieve the pre-defined number of successes. The parallel node is commonly represented in the diagram by a square containing two arrows. A representation of the parallel node in a diagram can be seen in Figure 2.7.

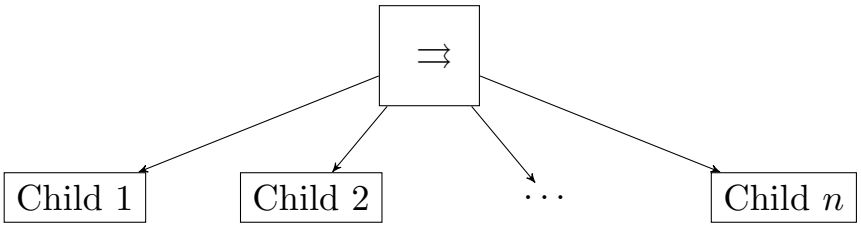

Figure 2.7: Parallel node with n children

Decorator Node

The decorator node manipulates the return from its single, underlying child according to some user-defined rule. It could invert the result from the child, repeat the execution of a child, or limit the number of attempts a child node can have before always returning failure. The decorator node is commonly represented in the diagram by a diamond, often with a label describing the rule of the decorator. A representation of the decorator node in a diagram can be seen in Figure 2.8.

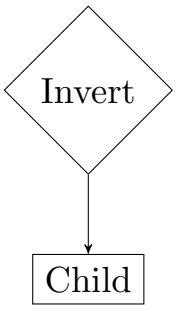

Figure 2.8: Example decorator node with child

Condition

When ticked, the condition node checks a pre-defined condition and then returns the outcome of the condition. It does not have any children, and it will never return running. The condition node is commonly represented in the diagram by an ellipse, often containing the condition as a label. An example of a condition node used with a selector node and an action node can be found in Figure 2.9

Action

When ticked, the action node executes a pre-defined function or action and then returns success if the action is completed successfully or failure if it cannot be completed. If the action is still being performed, the node will return running. The action node is generally where the physical processes of the system are completed. It is commonly represented in the diagram by a rectangle containing a label describing the action. An example of an action node used with a selector node and a condition node can be found in Figure 2.9

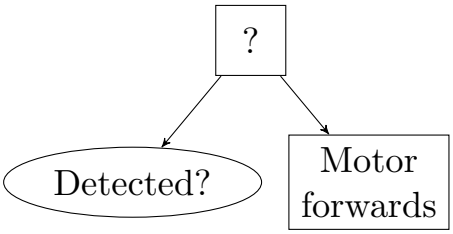

Figure 2.9: A small behavior tree consisting of a selector node, a condition node and an action node

Node Type	Succeds	Fails	Running
Sequence	If all children succeed	If one child fails	If one child returns "Running"
Selector	If one child succeeds	If all children fail	If one child returns "Running"
Parallel	If $\geq M$ children succeeds	If $> N - M$ children fail	Else
Action	Upon completion	If unable to complete	During completion
Condition	If true	If false	Never
Decorator	Custom	Custom	Custom

Table 2.2: The node types of a BT, from *Behavior Trees in Robotics and AI - An Introduction* [11]

In Figure 2.10 is an example of a complete behavior tree. The example is the same as the one represented in Figure 2.4, with a simple robot searching for an object by zig-zagging forwards. There was no need for the decorator node or the parallel node; hence, the example only utilizes four of the six nodes: The sequence node, the selector node, the condition node, and the action node. Due to behavior trees being traversed depth-first, the default behavior should be on the right side, and the most critical reactive behavior should be on the left side. In this case, the default behaviors are the two "turning and driving"-actions.

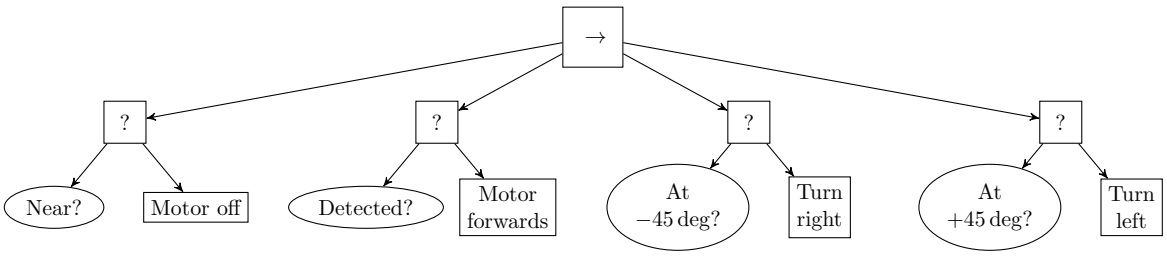

Figure 2.10: Behavior Tree Example

In a similar way to FSMs, constructing this BT will ease the implementation of the control system as the logic flow is much easier to follow. It will also reduce the risk of introducing programming errors while changing the logic flow. Also similar to FSMs, BTs can be used in different parts of the mission control system. The action nodes can be used to represent different behaviors, which in combination can achieve the goals of the autonomous system. These behaviors could both be low-level reactive behaviors, as well as high-level deliberative behaviors. Action nodes in a BT could also be replaced by smaller BTs dealing with the inner workings of the action. This makes it hard to place a BT in a specific layer, as it could be a part of all three layers. The overall structure of the BT could be seen as the planning layer, with conditions keeping track of what behavior (Action) should be executed at any given time. These behaviors could be both deliberative and reactive, and some could be a part of the planning layer themselves, i.e., a route replanning task, while others could be purely reactive and belong in the behavioral layer.

Implementing a behavior tree requires more initial work, as the looping/ticking and all the different nodes must be implemented before the actual BT can be constructed. Additionally, it can be very challenging to implement the BT in a system consisting of several different software. If anything is to be divided between software, it will be easiest to do so in the

implementation of the actions. Even though the initial implementation can be challenging, building the BT becomes a simple task of combining the different nodes to achieve the desired BT. Editing an existing BT is also relatively straightforward, whether it comes to removing nodes, adding new ones, or editing the existing nodes. This makes BTs scale quite well in terms of complexity and maintainability, and BTs would be a good choice for large-scale autonomous mission control systems.

Another asset of BTs is modularity. Subtrees of a BT can be reused in other BTs, or in other areas of the same BT without changes. This reduces the amount of code that must be written, and it reduces the complexity of restructuring the BT.

While FSMs have been the predominant design tool for transition between different modes for marine craft control systems, BTs are growing in popularity within robotics and could replace FSMs for some applications. Examples can be found in [43] where behavior trees are utilized for creating a modular, versatile, and robust control architectures for mission-critical systems in an AUV, and in [32] where behavior trees and control barrier functions are combined to take both sequential and concurrent objectives into account for a collaborative AUV coverage mission.

2.5.3 Choosing A Dynamic Specification Model

Both finite state machines and behavior trees could be valuable tools for creating and understanding the logic flow through an autonomous control system. They both have their strengths and weaknesses, and there is no definitive answer for which one to choose for a given mission control system.

The use of FSMs is widespread within the industry. While this might indicate that they would be fit for the purpose, it does not necessarily mean that FSMs are the best choice for any given mission control architecture. FSMs have been around for a long time, and there exists a large amount of research and applications within several fields. Which makes finding information about the creation and implementation of FSMs less demanding.

FSMs are also relatively straightforward to implement and could, in some instances, occur where no explicit decisions were taken regarding mission control architecture. An example of this could be a control system with three different controllers achieving different tasks, where conditional logic is used for deciding what controller to use when. In this case, the different controllers would be the different states, and the conditional logic for switching controllers would be the transitions.

While FSMs are easy to create, implement, and understand when they only consist of a few states and transitions, they can become unmanageable as they grow. Every transition in an FSM must be implemented manually, which can be tedious work for a developer. In the worst case, the number of transitions could be $O(n^2)$. If one state is removed or changed, all transitions to/from this state must also be changed. This increases the amount of work for the developer and thereby also increases the chance of an error being made. More states and transitions also mean increased complexity which makes it harder to understand the logic flow through the system, in turn partly defeating the initial purpose of the FSM. FSMs also

suffer from their lack of modularity. While some methods or states could be reused, most of an FSM will need to be custom-made to its exact purpose. Being able to reuse working code can save time and reduce the chance of errors. Attempts to mitigate this lack of modularity have been made, most notably Hierarchical Finite State Machines (HFSM) [17, 25]; however, they still suffer from low maintainability and manual implementation of transitions.

Behavior trees are both more modular and less complex in comparison with FSMs. In a BT, the transitions are implicitly defined by the structure of the tree, massively reducing the required work from a developer. A BT also scales much slower, with the worst-case number of transitions being $O(n)$. The tree structure of the BT will also enable the logic flow to be easily followed, even with a large number of nodes. When it comes to modularity, parts of a BT can be reused in other trees or other applications, as well as in the same tree. Another issue with BTs is the fact that some behaviors could exhibit blocking behavior, causing the system to be less reactive until the blocking is removed. This requires modifications of the standard BT structure, removing some of the initial benefits of using a BT. Another challenge with BTs is implementing them into an existing system. As the BT requires a ticking function and full access to most of the control variables, it can be hard to retrofit to an already existing system, i.e., something set up for manual control.

The computational difference between FSMs and BTs is not significant enough to warrant the use of one over the other. They are also both able to represent the same flow of logic through an autonomous control system. As a result, the choice of a dynamic mission management system can depend on the type of application. [30]

For smaller systems with few states where no significant changes or growth are expected after the initial implementation, FSMs can be a viable choice. An FSM will make the implementation swift and uncomplicated, and the logic flow will be straightforward to manage. If the mission control system requires a larger amount of transitions and states/actions, then BTs would be a better candidate. While they require some extra work in the beginning, they scale more effectively and allow for modifications without massively increasing the required work to do so. One should, however, be aware of some of the challenges regarding BTs. BTs are less mature as a topic within academia, and fewer tools and research are available when compared with FSMs. One common challenge is concurrency, as most real-world systems require the mission control system to handle different objectives simultaneously. As there is no established method for implementing concurrency in a BT, how to do so is up to the developer of the BT. When implementing concurrency in a BT, one challenge is to maintain the advantageous aspects it has over FSMs. For instance, part of the solution when implementing concurrency could be to introduce state within the nodes. This slightly complicates the implementation and heavily reduces the modularity of the nodes/components with state as they can no longer be fully reused in other parts of the BT. [11, 30]

In general, the number of states and actions increases more rapidly than most expect, meaning that BTs could be a good choice in most systems. The choice of dynamic mission management specification model should be made early in the development process, as BTs could require most of the mission control system to be built around it. If one already has an existing control system for manual control, it could be far easier to implement an FSM in the existing system instead of rewriting the control system around a BT.

2.6 Risk for Autonomous ROV

For autonomous ROVs to be a viable replacement for manually operated ROVs, the autonomous ROVs should not increase the risks connected to the use of ROVs in aquaculture. Hence, a mission control system for an autonomous ROV should be able to detect situations where the risk is higher than acceptable. Therefore, it is also important that the autonomous ROV is capable of handling these situations to minimize the risk and the consequences. In order to detect and handle these situations, it would first need to know what situations it should avoid, and secondly, realize when the risk during regular operation is too high. In these situations, the risks should be considered for the humans involved, the fish in the net cage, and the ROV itself.
This section is primarily based on the paper written for the specialization subject "TMR06 - Autonomous Systems", which, as stated earlier, was written by the author in collaboration with fellow students at the Department of Marine Technology at NTNU and can be found in Appendix A.

2.6.1 Risk for Humans

Autonomy will reduce both the amount of overtime work and the general workload for the ROV operators. Large amounts of overtime and a significant workload increase the possibility of mistakes and operator-induced failures. The introduction of autonomy will therefore reduce some of the current risks in aquaculture. However, the ROVs will also introduce new risks for the operators. Currently, there are some risks tied to the handling of manual ROVs in aquaculture. The ROV must be launched and recovered from the water, which is typically done with cranes. Crane operations are connected to several risks for the operators; however, little can be done by the autonomous mission control system to reduce these risks. Operators could also be exposed to risks in situations where the autonomous ROV requires assistance. This could happen if the ROV or some part related to it gets stuck. The ROV itself could get stuck in the net, mooring lines, or other structures. It could also get the tether stuck by wedging it somewhere or wrapping it around something. This would require the operator to leave the relative safety of the control room and perform some physical action to get the ROV back in operation. The mission control system should therefore minimize the risk of getting stuck in structures by keeping track of its current and past positions relative to the structures in the net cage.

2.6.2 Risk for the Fish

When it comes to risks related to the fish in the net cage, there are two different aspects. Firstly, the risk of reduced welfare or fatality for individual fish should be considered. Secondly, risks affecting larger numbers of fish, or even the whole population of a net cage, should be considered.

Fish as individuals

For the individual fish, the highest risk of damage or fatality is the risk of coming into contact with the ROV and its moving parts. This rarely happens with salmon as it will usually stay clear of the ROV [24]. However, cleaner fish seem to be less mindful of the ROV and sometimes comes right up to it [3]. This is not easily handled in the mission control system, as the fish are unpredictable and in large numbers. Therefore, measures to reduce these risks should instead be taken in the design of the ROV and the thrusters.

Another risk that could be increased by the ROV is the risk of reducing welfare due to stress. Stress could lead to sickness, which again could lead to fatality [6]. The fact that the salmon stays clear of the ROV could be an indication of increased stress. Research regarding how different shapes, sizes, and motions affects the salmon was performed in (Kruusmaa, et al. 2020) [24] but few conclusive results were found. However, the mission control system of the autonomous ROV could minimize motions and movement through regions where the fish typically gathers, i.e., minimize tracking across the net cage, for reducing the direct contact with the fish.

Fish as a population

The highest risk for the whole population of fish, related to the use of ROV, is the risk of creating holes and causing escapes. Escapes lead to economic loss for the aquaculture company, fatality and reduced welfare of the fish, and most importantly, negative ecological and genetic impacts for wild fish populations. While little evidence exists of ROVs causing holes in the net, it is a risk that is present. ROVs could create holes when coming in contact with the net, especially if they have sharp edges/points or something protruding from the ROV. Therefore, it is important that the mission control system can perform the desired operations without coming into contact with the net unless it is absolutely necessary. The mission control system should also be able to decide in real-time if it is safe to proceed with the operation or if it should be aborted.

2.6.3 Risk for the ROV

There are fewer risks related to the safety of the ROV, and the worst consequences would be loss of the ROV or damages to a point where it no longer functions. As the ROV is tethered, the tether would need to be cut for the ROV to disappear. This could happen if the tether is gnawing on something over time or if it comes into contact with a spinning propeller. Keeping track of the tether is challenging for the mission control system, but by keeping track of the previous positions of the ROV, some knowledge of tether location can be assumed. The mission control system could also minimize the movement near known vessels to lower the risk of the tether being cut. Finally, the mission control system should be able to detect if the tether is stuck.

Chapter 3

Method

The mission script in this project was implemented for SINTEF Ocean's Argus Mini ROV, used extensively for net inspection and fish monitoring purposes. The Argus Mini ROV has a complex architecture tied to the guidance, navigation, and control of the ROV. The ROV uses the simulation software FhSim, developed by SINTEF Ocean, for the low-level control and operational modes such as station keeping and net-following. The interface between FhSim and the operator is Aqueous, and it can also perform higher-level control tasks such as mode switching. Aqueous was developed by students at NTNU as a part of the course *TDT4290 Customer-Driven Project* in the autumn of 2019. FhSim and Aqueous communicate by using the Inter-Module Communication Protocol (IMC) developed by Laboratório de Sistemas e Tecnologia Subaquática (LSTS).

This chapter will look into the current implementation of the control system, potential new tasks, where they could be implemented, and how a mathematical model of computation could support the implementation of the mission control system. As the implementation was done in several iterations, it will be covered in chapter 4.

3.1 Argus Mini ROV

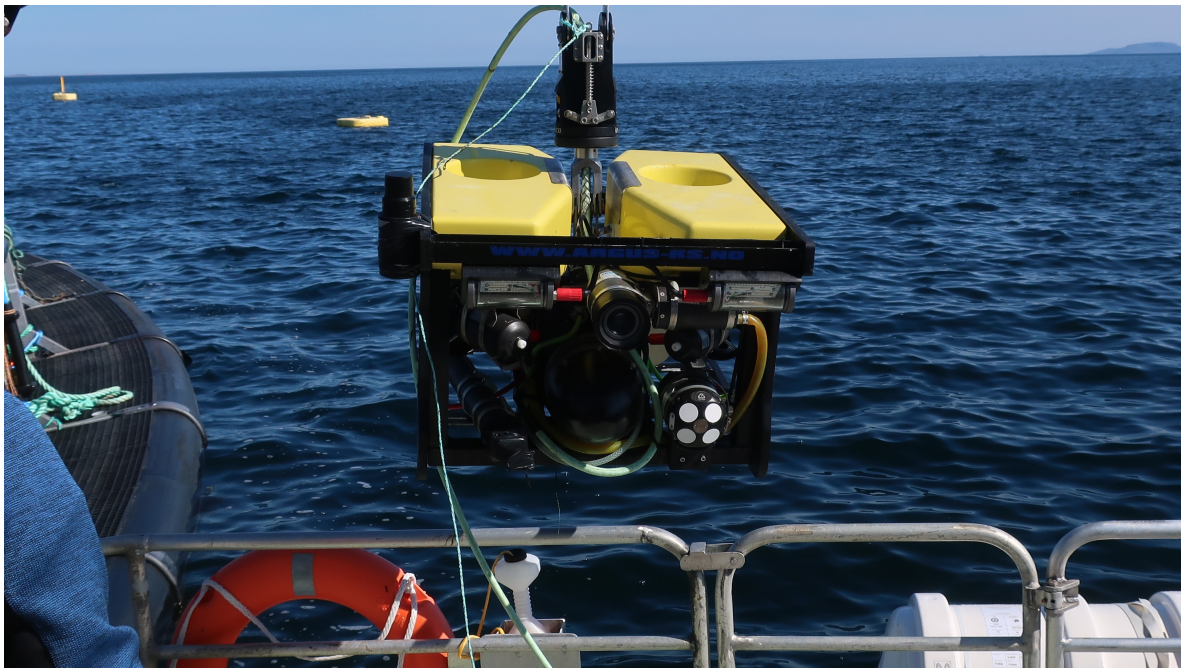

Figure 3.1: SINTEF Ocean's Argus Mini ROV, with a front-facing DVL sensor mounted on the port side and the USBL transponder mounted on the outside of the frame on the starboard side.

The Argus Mini ROV is a class II ROV developed by Argus Remote Systems AS for both offshore and inshore use, marketed towards uses within science and aquaculture. It is 0.9 meters long, 0.65 meters wide, 0.5 meters tall, and weighs around 90 kilos in air. The Argus Mini has six thrusters. Four of the thrusters are exclusively horizontal and are all angled $\pm 35°$ from the longitudinal axis. These four thrusters enable the ROV to be fully actuated in the horizontal plane. The two remaining thrusters are vertically mounted and actuate the ROV in heave. Hence, the ROV is fully actuated in surge, sway, yaw, and heave. The ROV is made to be self-stabilizing in pitch and roll. The Argus Mini comes with a depth sensor, compass, lights, and a camera. In addition, up to 5kg of payload sensors and equipment can be fitted. SINTEF Ocean has fitted their Argus Mini with a front-facing Nortek DVL 1000 for net-relative positioning and a Sonardyne Micro-Ranger 2 ultra-short baseline (USBL) acoustic positioning system for dynamic positioning. [3, 5]
SINTEF Ocean's Argus Mini ROV is depicted in Figure 3.1

3.2 FhSim

The software used for low-level control of SINTEF Ocean's Argus Mini ROV is the simulation software FhSim. FhSim is a software framework aimed at the simulation of marine systems in the time domain. It does this by solving ordinary differential equations (ODE) such as the equations of motion for the ROV or other models of reality. FhSim is written in C++ and utilizes the fact that C++ supports object-oriented programming. More information about

the details of FhSim can be found in [35] and [45]. The FhSim setup in this thesis was based on the aquaculture robotics setup which is being continuously developed by SINTEF Ocean. This setup includes a mathematical model of the Argus Mini ROV and allows for control of the ROV in both real use and in simulation.

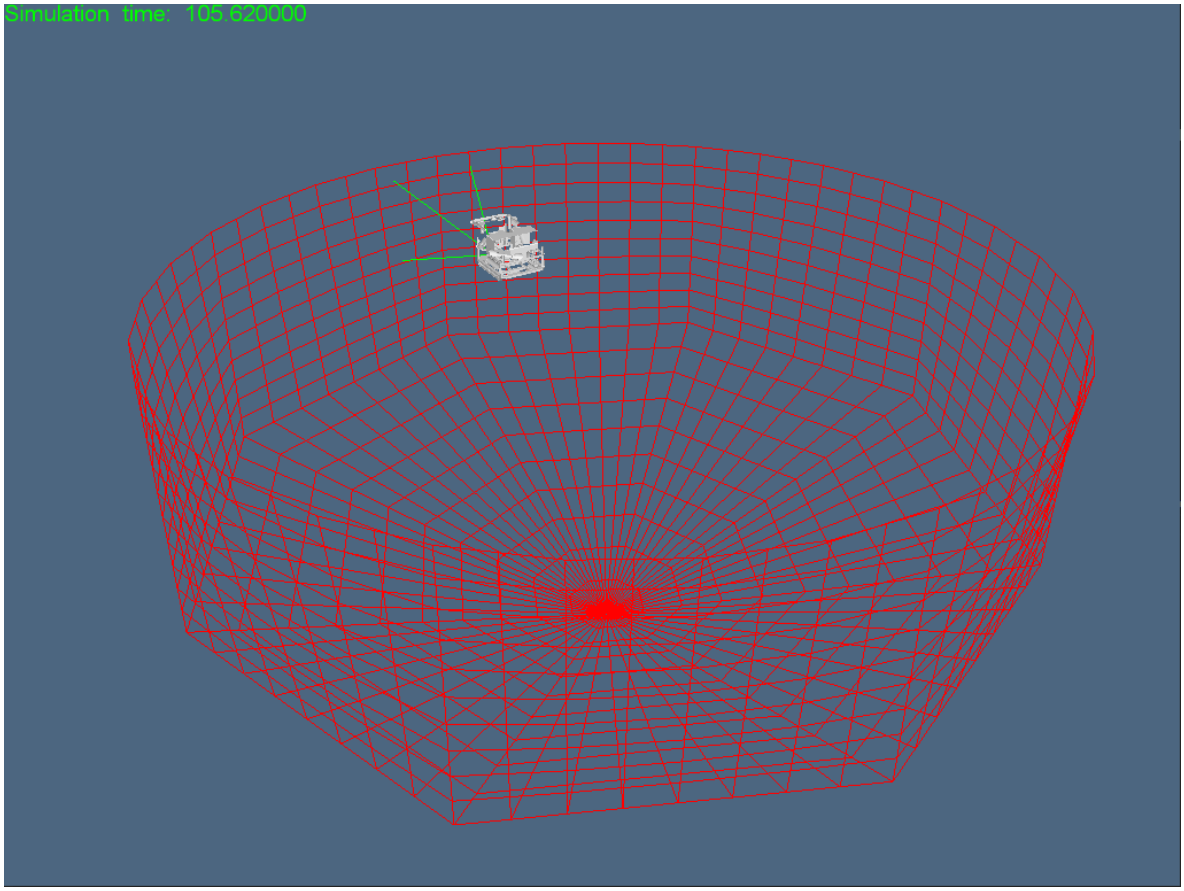

Figure 3.2: FhSim's visualization of the ROV in the net cage, traversing the net by using the DVL for estimating the net position

A predecessor of the current setup was used in [4]. It enables simulation, control, and visualization of the Argus Mini ROV for manual IMR operations in aquaculture, as well as dynamic positioning and autonomously traversing the net. It has later been expanded to also include vertical traversal of the net. An example of a simulation using this setup can be seen in Figure 3.2. This simulation is utilizing a model of the ROV, a simulated DVL, and a simulated net cage in order to represent the real system as closely as possible. As the aquaculture robotics setup has been in development for a while, it includes several low-level behaviors that are useful when implementing autonomous IMR tasks. Examples of these behaviors are; automatic heading hold, automatic depth hold, and station keeping. Some of the low-level behaviors are useful as independent behaviors, while others are parts of higher-level behaviors. They can also be included as parts of new, autonomous, higher-level behaviors. The existing low-level behaviors might also require some alteration to be fully utilized in an autonomous mission control system. FhSim would also be a suitable place for implementing new low-level behaviors, such as reactive behaviors, for autonomous IMR tasks.

Further, when referring to FhSim the author is alluding to the aquaculture robotics FhSim setup, developed by SINTEF Ocean. In this thesis, FhSim has been used for all simulations where a model of the ROV was required as well as all real-life experiments.

3.3 Aqueous GUI

The Aqueous GUI is a tool for monitoring the ROV and interacting with the control system. Aqueous allows the operator to manually control the ROV, enable different modes, monitor the different parameters of the ROV, and see the camera feed. It is written in the JavaScript library React and was developed by students at NTNU as a part of the course *TDT4290 Customer-Driven Project* in the autumn of 2019 [12]. It has since been further developed and maintained by SINTEF Ocean. For communicating with FhSim, Aqueous uses the IMC protocol. Aqueous consists of two views, one view for choosing control mode and modifying control values, and one view for showing the video feed from the ROV with some overlaid data. Further in the project, these two views will be referred to as the control app and the video app, respectively.

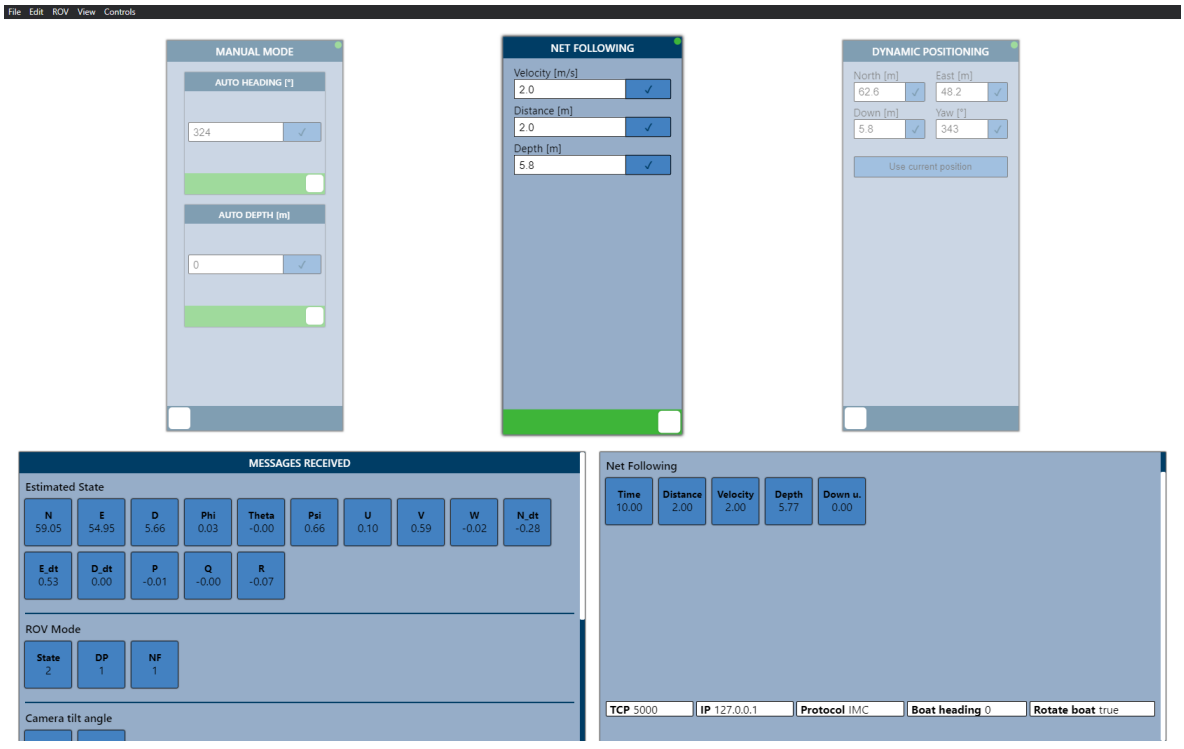

Figure 3.3: The control app in Aqueous while in the net-following mode, with all modes available.

3.3.1 Control App

The control app, as the name suggests, allows for control of the ROV. This is where switching between modes is done, and where the different parameters can be changed. The control app also displays the incoming data. Both the parameters and the incoming data are dependent on

the different modes. The state of the ROV, the available modes, and the camera is available to monitor in all modes, but each mode can have some extra data that can be monitored. The interface of the control app can be seen in Figure 3.3. The different modes are Manual Mode, Net Following, and Dynamic Positioning.

Manual Mode

In manual mode, the operator can control the ROV with the keyboard or a game controller. This is achieved by changing the force bias in the different directions. Apart from the input from the keyboard or game controller, the operator can also input a heading for the automatic heading holder, or a depth for the automatic depth holder. In manual mode, the state of the ROV, as well as the desired biases from the keyboard or controller, can be monitored. Manual mode will always be available and is the fall-back mode if one of the other modes fails.

Net Following

In net-following, the ROV is traversing the net by using the DVL measurements for estimating the position of the net. The operator can input the desired depth, distance between the net and the ROV, and the velocity perpendicular to the net, as well as choosing between vertical and horizontal net-following. Similarly to in manual mode, the state of the ROV can be monitored, in addition, there is a net-following state showing the current velocity, distance to the net, and net angle. The current input for the net-following is also displayed.

Dynamic Positioning

Dynamic positioning (DP) allows the ROV to keep a stationary position or move small distances with great accuracy. The operator can input the desired position in north, east, down, and yaw (heading) or use the ROVs current position. The desired position can also be changed incrementally with input from the keyboard or a game controller.

3.3.2 Video App

The video app receives the video feed and displays it to the operator. Overlaid on the video feed are data such as depth, heading, position relative to the service vessel, which modes are available, which mode the ROV is currently in, and what the current state of that mode is.

3.3.3 Signal Flow Through Aqueous

Aqueous receives IMC messages from FhSim, translates the data, and saves it in global objects, which can be accessed from anywhere in the Aqueous software. Part of this data is directly displayed to the operator; examples of this are the position and depth. Some of the data is synthesized before being displayed, such as the heading and velocity. And some of the data is only available in the settings, which is the case with the camera settings. When the operator changes some parameter of the ROV, i.e., the desired position in DP, the global objects are modified, the data is validated, and then translated into IMC messages before being sent back to FhSim.

3.4 Inter-Module Communication Protocol

The communication between FhSim and Aqueous is done over the transmission control protocol (TCP) with the Inter-Module Communication Protocol (IMC) running on top. IMC is developed by Laboratório de Sistemas e Tecnologia Subaquática (LSTS) (Underwater Systems and Technology Laboratory) which is an interdisciplinary research laboratory located in Porto, Portugal. The main goal of IMC is to define a communication protocol for interconnected systems that can work for all types of vehicles and computer nodes in networked environments [26]. To achieve this, IMC abstracts the hardware it runs on and the communication layers it uses. It then uses a pre-shared set of messages that are serialized and transported over different communication technologies. This makes IMC independent of hardware and communication technologies.

IMC relies on a standard set of messages, but custom messages can be pre-defined and added. The communication between FhSim and Aqueous relies on a mix of standard messages and custom messages. Currently, FhSim can receive all IMC messages, but not all messages will result in control output [18]. Aqueous only supports the messages currently in use. When expanding the existing software for autonomy, new custom messages would need to be implemented, and standard messages not currently in use could become useful.

3.5 Existing Implementation

The implementation of autonomous functions in this thesis builds on the existing control system implementation and the autonomous features already implemented in it. The existing control system allowed for manual control of the ROV, autonomously following the net and dynamic positioning, and appears to the operator as the three modes presented in section 3.3. These modes build on several different controllers and behaviors implemented in FhSim. For a clear overview of the current implementation, both in Aqueous and FhSim, the modes will now be presented with the origin in how they appear in Aqueous.

3.5.1 Manual Mode

The manual mode allows for direct control over the ROV. In Aqueous the operator can input desired control for control of the ROV in four degrees of freedom (DOF). This input is sent to FhSim through IMC and first routed through an interface for reading messages from Aqueous and writing messages to be sent back, called the Aqueous interface. From this interface it is routed to a ModeSwitcher-object which checks the current mode and routes the desired control forces into the thruster allocation object. If autoheading or autodepth is active, a PI-controller calculates the desired forces for achieving the desired depth/heading. It is then sent to the ModeSwitcher which replaces the original desired control forces with the calculated control forces and routes them to the thruster allocation object. The Aqueous interface could also cause a switch from one of the other modes into manual mode, if some required sensor data is unavailable.

3.5.2 Net Following

The net-following mode enables the ROV to autonomously follow the net cage. It was created in [2], and more details can be found in [2] and [4]. It has later been expanded to include

net-following in the vertical direction. In short, the net-following mode utilizes a DVL for estimating a vertical plane of the net relative to the ROV and then runs an Integral Line of Sight (ILOS) controller for moving perpendicular to this estimated plane of the net. The operator can choose the desired velocity, horizontal or vertical direction, depth, and distance in Aqueous. Similar to the manual mode, this is sent to FhSim through IMC and routed through the Aqueous interface. From the Aqueous interface, the parameters are entered into the ILOS controller object. The ILOS controller object also requires the crosstrack error relative to the desired path, the tangential angle to the desired path, and the heading and pitch of the ROV. The heading and pitch are retrieved from gyro and compass of the ROV. For calculating the crosstrack error and path tangential angle, a plane of the net must first be estimated. This is done by the net approximation object, which takes in the distance beams from the DVL and the position of the ROV, and estimates the plane relative to the ROV. The net heading (normal vector in the horizontal plane), net pitch, and distance from the ROV is then sent to a "NetFollowingManager"-object, which calculates and returns the crosstrack error and path tangential angle. All of this is then combined in the ILOS controller object for calculating the desired speed in surge, sway and heave and yaw. The desired speeds are then sent to different low level controllers for calculating the desired forces before being fed to the thruster allocation object. As the net-following mode relies on the estimated net plane based on the DVL distance beams, it cannot function without valid DVL signals. Because of this, the Aqueous interface checks if the DVL beams are available and valid, and if it is not, it switches mode to manual mode and relays this through IMC to the Aqueous GUI. When using a DVL for measuring the distance and speed difference to the ocean floor, it can be said that the DVL has bottom lock if the DVL has valid readings. A similar term for the case where the DVL has a valid reading of the net can be "Net-lock," and this will be used as a term regarding the availability of the DVL measurements for the remainder of this thesis.

3.5.3 Dynamic Positioning

The dynamic positioning mode allows the ROV to stay stationary in a chosen position or move a small distance to a desired position. The operator can choose a desired position within 3 meters from the current position. This position is routed through to FhSim in the same way as in the manual and net-following modes. The desired position, as well as the current position, is inputted into a non-linear 4DOF controller, which calculates the required forces for the ROV to reach, or stay in, the desired position. For calculating the required forces, the distance and direction from the current position to the desired position must be known. Hence a position-reference system is required. Position-referencing can be challenging underwater, and SINTEF Ocean's Argus Mini ROV utilizes a combination of sensors to achieve this. An extended Kalman filter is used in FhSim for taking in measurements from different sensors and outputting a position estimate. The extended Kalman filter takes in measurements from a Sonardyne USBL acoustic positioning system, in combination with measurements from the DVL, compass, depth sensor, and gyro.

3.6 Tasks in Autonomous Mission Script

In order to autonomously perform the operations mentioned in section 2.1 these operations and their underlying behaviors must be implemented in the different layers of the control system in the autonomous ROV. Some of them will be directly related to the IMR-operations

listed in section 2.1, such as net inspection and mooring inspection. Other tasks will be required for the different phases before, in between, and after IMR-operations, such as launch and tracking. There will also be some more general behaviors that can be used in several of the other tasks, such as dynamic positioning, automatic heading hold, and automatic depth hold. Most tasks and behaviors should include functionality for monitoring and handling risks.

3.6.1 Manual Control

The manual control mode will be the basis mode, similar to how it is today. If there is no way the system can handle the situation, the ROV should switch to manual mode, and a human operator should take over the controls. The manual mode in ROVs is familiar, as this is what is standard use today. It should therefore be available through the Aqueous interface such that the ROV can be used for operations that are not implemented in the autonomous mission system. The manual mode does not need a large amount of functionality for monitoring and handling risks, as this would be left to the human operator. Some measures for monitoring risk could still be included for aiding the operator, i.e., the ROV could inform the operator that it is about to hit something or that the tether is stuck. As the manual control mode is already implemented, it will not be changed in this project.

3.6.2 Dynamic Positioning

The dynamic positioning (DP) mode is implemented in FhSim and enables the ROV to keep stationary at a desired position. It can also do low-distance relocations to new desired positions. The desired position in the current implementation allows for set points in surge, sway, heave, and yaw. The two main situations in the DP mode that the autonomous mission control system should be able to handle is if the ROV is unable to reach the desired position, or if the USBL measurements are lost. Previously, a reactive behavior was implemented for the second situation. If the USBL measurements were lost, FhSim switched to manual mode and set automatic heading and depth hold such that the movement of the ROV was minimized. However, the USBL position-estimate was replaced with the output of a Kalman filter, which always outputs an estimate, effectively removing this behavior from the mission control system. In the case where the ROV is unable to reach the desired position, the autonomous mission control system should make a decision based on previous knowledge and current position data.

3.6.3 Net Inspection

A mode for net-following is implemented in FhSim. It enables the ROV to autonomously traverse the net while keeping a set depth, velocity, and distance relative to the net. The net-following mode will be further expanded in this thesis in order to conduct inspections of the full net cage autonomously. The decision to expand the net inspection mode was made as the net-following mode is it could allow for fully autonomous net inspection. The net-following mode needs several both reactive and deliberative behaviors to perform a fully autonomous net inspection. The different behaviors will now be further explained.

Basis Behavior - Net-Following

The basis behavior of the autonomous net inspection will be the net-following controller from [4]. This utilizes a forward-looking DVL for estimating the distance and heading to the net. The DVL measures the distance to four different points with a set angle between them and estimates the net as a flat square between these four points. This is then used for estimating the net and controlling the ROV relative to it. The net-following controller will let the ROV traverse the net, with a distance and velocity given by user input. Currently, the only risk handling functionality implemented in net-following is a simple switch to manual mode with automatic heading and depth holding enabled. This switch will trigger if the DVL loses its measurements of the net, i.e., loses net-lock. This is something that frequently happens due to fish swimming between the net and the DVL. When a fish comes between the net and the DVL, it usually only takes a few seconds before the DVL regains net-lock. Hence, some filtering to allow this to happen and then continue traversing the net could greatly improve efficiency. If the measurements of the net do not return, the ROV could enable DP in the position where it last had net-lock. The ROV should then go back to net-following if the net-lock returns. If this pattern repeats, or no net-lock is found while in DP, the ROV should either start looking for the net, using a behavior for finding the net, or await controls from an operator. A behavior for managing the loss of net-lock was implemented in this thesis. A simple version of the find net behavior was also implemented. More details regarding the implementation of these behaviors can be found in chapter 4. Another thing that could happen while in net-following is that the ROV could get itself or the tether stuck, rendering the ROV unable to move as desired. If this happens, the ROV should carefully attempt to backtrack in order to free itself. And if this does not succeed, it should stay stationary to minimize possible damages. This would require some method for detecting whether the ROV can move as desired or not. Such a method does to the authors knowledge not yet exist, and it was deemed out of scope for this thesis.

Path Planning

The net-following controller will currently only traverse the net at a given depth until instructed otherwise. By implementing a deliberative behavior, the ROV could be able to know when it should change its depth or direction in order to inspect the whole net cage. For this to be as efficient as possible, the system would need to include path planning. Path planning could also reduce the risk of the ROV getting stuck and of the tether getting tangled. A path planning behavior should ideally know the size and depth of the net cage, as well as the location of different ropes and obstacles inside the net cage. When the net is fully inspected, a "return home" behavior should be called. A deliberative behavior for tracking the inspection progress and planning the inspection path, allowing the net to be fully inspected, was implemented in this thesis. More details can be found in chapter 4

Locate Net and Achieve Net-Lock

When the ROV is placed inside the net cage, it should be able to autonomously locate the net and achieve net-lock with the DVL to start the net-following behavior. It could also be used during an inspection if net-lock is lost and the net needs to be found again. The ROV should rotate around and attempt to gain/regain net-lock. If the DVL cannot locate the net and regain net-lock, it should go to the DP mode and await operator interaction. A proof

of concept "locate net" behavior was implemented for this project. Details regarding the implementation and discussion regarding the associated challenges can be found in chapter 4.

3.6.4 Mooring Inspection

A mooring inspection mode would have several similarities with the net inspection mode. It would also need a following behavior as the basis behavior, but following the mooring line instead of the net. For a mooring following behavior, the ROV would need some system for positioning itself relative to the mooring line. Similar to in the net inspection mode, the mooring inspection mode would need some sort of deliberative behavior to determine the progress of the inspection. This could be a path planning behavior if information such as anchor depth and mooring line length is provided. Otherwise, the deliberative behavior must be able to identify if it has reached an end of the mooring. Again in similarity to net-following, the mooring inspection mode must have some behavior for locating the mooring from a position without a relative position reference. As mentioned in the theory chapter, a method for localizing a flexible anchor line was proposed and tested in [20, 37]. It was, however, not tested for use with an ROV. The general risk in mooring inspection, compared to that of net inspection, is lower. This is due to the ROV being placed outside of the net. The ROV could still run into the net, but the probability is lower. The ROV could also get stuck in mooring lines or other obstacles, but this risk is also reduced as the ROV should keep track of the mooring line during the inspection. One risk that is increased is cutting the tether, as it might be cut if it comes in contact with the running propellers of a service vessel. This could also lead to the ROV drifting off, potentially being lost.

3.6.5 Net Repair

A net repair mode would also have some similarities to the net inspection mode. Firstly the net repair mode would need some way of locating the hole to be repaired. Ideally, the ROV should be able to identify the hole in the net inspection mode and then switch to net repair mode for repairs as this would reduce the complexity of the "locate hole"-behavior. If the ROV is unable to conduct inspections and repairs in the same run, the repair mode would need to know the position of the hole and possibly be able to identify the hole. This would require the ROV to have some positioning system, both georeferenced and also in reference to the net cage. In addition, the ROV should have a sensor capable of measuring the distance between the net and itself, such as the DVL used in [4]. This would be for minimizing the risk of the ROV coming into contact with the net at an undesired point or time. Lastly, the net repair mode would also require some behaviors that would not be found in the net inspection mode. If the ROV is unable to conduct both inspection and repair in the same operation, the ROV would require a tracking mode for navigating between the launch and recovery site and the hole to be repaired. This will be explained further in Subsection 3.6.10. A basis repair behavior would also need to be implemented and could be further divided into two behaviors; one behavior for keeping the ROV stationary relative to the net and one for conducting the repair using the relevant tools. Keeping the ROV stationary relative to the net could be achieved with the method presented in [4] and would be important to avoid enlarging the hole by getting the tool stuck in the hole. Conducting the repair itself requires controlling some actuator or tool that seals the hole and identifying if the hole is properly sealed. Sealing the hole could be performed by a sewing tool and the identification of completeness by a

computer vision algorithm. Other solutions, such as attaching an extra piece of net over the hole, could also be viable. Some solutions have been proposed by the industry but seem to get little use [3, 41]. The net repair mode would greatly benefit from ROVs being used more commonly for net repairs before being made autonomous.

3.6.6 Net Cleaning

The net cleaning mode would differ slightly from most of the other modes as it requires constant contact with the net. However, it would still have some of the low-level behaviors in common with other modes. In similarity with several of the other modes, it would need a tracking mode for moving to or from the net cleaning operation. It could also benefit from a behavior locating the net such as mentioned in net inspection, there is however a possibility that the net locating behavior from net inspection could not be reused due to differing sensors and tools. Another behavior similar to what can be found in net inspection would be a behavior for keeping track of the cleaning progress. This would be useful for avoiding moorings and obstacles in the net cages and for knowing what parts of the net have been cleaned and what parts are remaining. It would also need a behavior for traversing the net while in contact with it. This behavior should be careful not to apply unnecessary force to the net and should be able to measure the force applied to the net. A behavior connected to the activation and control of the cleaning tool would be useful for avoiding large consequences in cases where the ROV is stuck or without control, and the moving parts of the cleaning tool could make the situation even worse. While some purpose-built ROVs for cleaning of the net exist and further research is being conducted, autonomous net cleaning has not been publicly demonstrated, and no solution currently seems ready.

3.6.7 Automatic Heading Hold

The automatic heading hold behavior is a behavior that currently can be activated while in the manual control mode. It controls the heading of the ROV such that it always points in the same direction, while the ROV is available for control in any of the other directions. The automatic heading hold behavior should incorporate some risk managing functionality. The most relevant risk is that the heading hold is unable to reach or hold the desired heading. In manual mode, this would just be relayed to the operator, while in other modes, it might impact the decisions made by the system. Automatic heading hold would be most useful in the net repair mode and possibly the mooring inspection mode, depending on how the mooring inspection mode is implemented. The underlying controller of the automatic heading hold behavior is used for controlling the heading in the net-following mode. This mode will further be referred to as autoheading.

3.6.8 Automatic Depth Hold

Similar to the automatic heading hold behavior, the automatic depth hold behavior can currently be activated in the manual control mode. It controls the depth of the ROV but leaves the other directions available for control. Automatic depth hold is currently used in all of the other modes, and it would be useful for most other modes in the autonomous mission control system. Again in similarity with heading hold, the depth hold behavior could have some risk functionality regarding if the ROV is unable to reach the desired depth or not. If the ROV is unable to reach the desired depth, there can essentially only be three reasons,

the ROV is stuck in the vertical direction, the ROV is at the bottom, or the ROV is at the surface. Based on previous knowledge, the mission control system could then decide what would be the best action. This mode will further be referred to as autodepth.

3.6.9 Launch and Recovery

Typically, ROVs used for aquaculture operations are launched from service vessels fitted with cranes. The ROV is lifted by the crane and released into the net cage. Figure 3.4 shows the Argus Mini ROV being manually lifted into the net cage. When the ROV has completed its mission, it is lifted back onto the service vessel. This is currently a manual operation and will probably continue to be so until the ROVs are fully resident and an autonomous Launch And Recovery System (LARS) is implemented. The autonomous system would probably not be ready to start operations directly after being placed in the net cage, and a launch behavior could be useful. The launch behavior should check that all sensors, actuators, and other relevant systems function as expected. If the positioning system is found to be functioning as expected, the position of the launch should be saved as a possible safe position. The ROV should also take measurements of environmental conditions such as current and waves for use in the low-level control system. When all systems have been validated, and the calibration has been completed, the autonomous mission control system should continue with the desired operation.

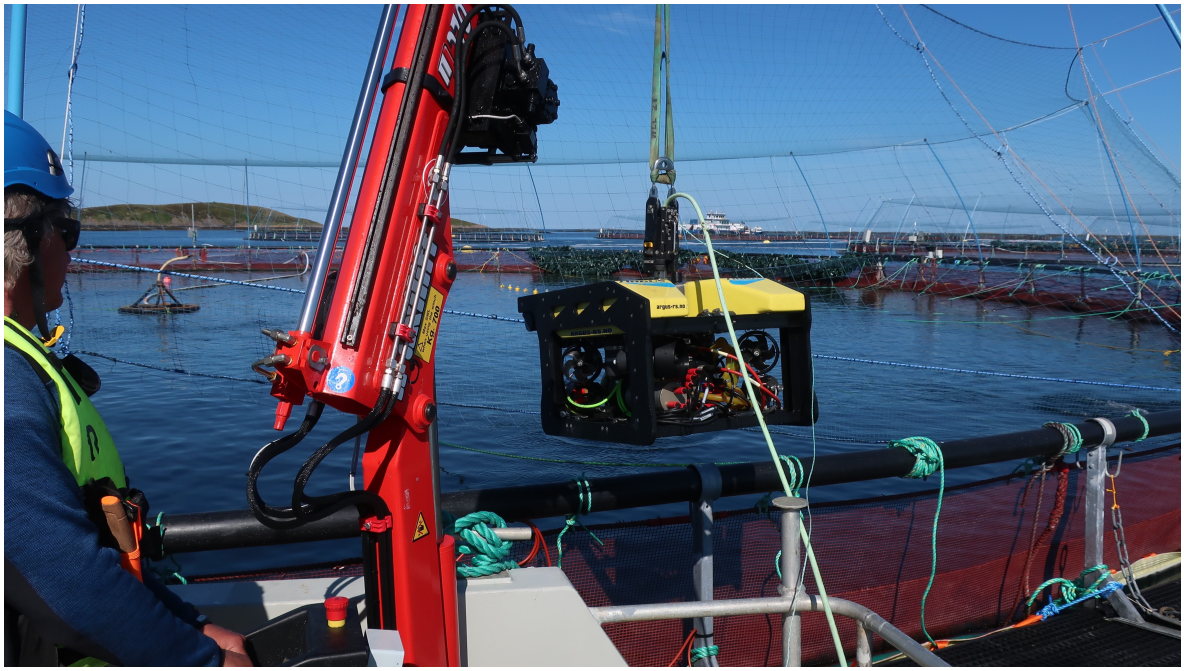

Figure 3.4: SINTEF Ocean's Argus Mini ROV being lifted into an aquaculture net cage by a manually controlled crane

3.6.10 Tracking

The tracking behavior would enable the ROV to move from a point A to a point B. This would be useful for most of the modes, exceptions being DP and manual control. All the

other modes would at some point be completed, and the ROV would need to return to some position for recovery. The net repair mode might also need to start by tracking to the position of the hole. Tracking behaviors for ROVs exist today but are not common in aquaculture [3]. When implemented for ROVs in aquaculture, the tracking behavior must be able to detect whether it is stuck and keep track of where it has been to minimize the risk of tangling the tether. Ideally, it should also have capabilities for collision avoidance. The tracking behavior would be closely related to path following or path tracking, which are well studies areas of control. If the approximate locations of mooring lines and other obstacles are known, a path planning algorithm could be implemented, both for efficiency and for reducing the risk of tangling the tether.

3.7 Location of Further Implementation

To achieve a higher level of autonomy, new modes, behaviors, and switching logic would need to be implemented. The different components could be implemented in different parts of the software stack, and an important part of this thesis was to identify exactly where the different components of the mission control system should be implemented. This was achieved by combining the GNC architecture with the layered hybrid approach to mission control. The autonomous components can be split into three different categories; deliberative components, switching logic, and reactive components.

3.7.1 Implementation of Deliberative Components

As Aqueous handles parts of the high-level control such as managing modes, it was natural to examine if it could be a suitable place for implementing the deliberative components. The Aqueous software, as presented in [12], contains no deliberative functions, and all deliberation must be done by the human operator. For deliberative functions to be implemented in Aqueous, new modes should be implemented, and the existing modes and methods must be expanded to enable higher levels of deliberation. Some of the modes could even be hidden from the operator, as the operator only should intervene if the autonomous mission control system is unable to handle a certain situation. A major part of the changes in Aqueous will be to create methods and algorithms for building the knowledge required for deliberation, based on the data from the ROV. Aqueous could be seen as part of the planning layer in the layered control approach; it also performs some high-level guidance tasks, which would be a part of the guidance block from GNC. With the existing software and software interaction, Aqueous is a suitable location for implementing deliberative functions, as it has the data from all the sensors readily available. Aqueous could also be running on a larger and more powerful computer located onshore while the computer running FhSim must be in close proximity to the ROV, possibly even be a part of the ROV itself. The proposed new modes and extensions to the existing ones will be presented in Section 3.6. Further details regarding the implementation of the deliberative components can be found in Section 4.

3.7.2 Implementation of Switching Logic

Both Aqueous and FhSim handle a share of the mode-switching. Aqueous handles the switching by operator input, while FhSim can switch to manual mode if the required sensors for the other modes are missing. This can happen in DP-mode if the USBL signal is invalid or in

net-following if the DVL does not have net-lock. As the different software and the communications between them does not precisely mirror the ideas from hybrid control or the different architectural layers, the location of the switching-logic implementation will be dependent on the different modes. Each mode will have some risk or goal-based criteria for changing away from itself and to some other mode. For this thesis, the different switching logics could therefore be implemented in both Aqueous and FhSim, depending on what mode or behavior is implemented for. As mentioned in section 3.5 the FhSim implementation already handles mode switching based on loss of USBL and DVL measurements (net-lock). Further details regarding the switching logic can be found in chapter 4.

3.7.3 Implementation of Reactive Components

Reactive behaviors could be implemented in Aqueous but should preferably be implemented within the lower-level behaviors in FhSim as this would lead to faster response times and make the ROV less reliant on Aqueous. In this thesis, a reactive behavior was implemented in Aqueous. An attempt at implementing the same reactive behavior in FhSim was made, but it was not used in the final simulations or the field tests. More information about this can be found in chapter 4.

3.8 Dynamic Mission Management Specification

In order to create a mathematical model of computation, it must first be known what the goals of the system are. In this section the focus will be on an autonomous mission control system for fully autonomous inspection of the net cage. Hence, the goal will be to have a mission control system capable of conducting a fully autonomous inspection of the cage. This requires several of the tasks presented in section 3.6 to be implemented as high-level behaviors in the autonomous mission control system. Depending on the chosen model, the high-level implementation of the tasks could either be as states in an FSM or as actions in a BT. Regardless of the model, the goal will stay the same. The mission control system must be able to handle the following: When being placed in the net cage, the ROV should check that all systems are working as intended. This check should continue throughout the entire mission, and if something does not function as required, the ROV should return to the recovery position. After the initial checks, the ROV should acquire net-lock if it does not already have it. If the DVL loses net-lock, the ROV should reacquire it. The ROV should then conduct a net inspection until the entire net cage has been inspected. If the ROV at any point gets stuck, it should set DP and await assistance from the operator. When the entire net cage has been inspected, the ROV should return to the start position for recovery.

The following mathematical models of computation have been made as suggestions for how the required behavior stated above can be achieved in a fully autonomous mission control system. The models include several of the high-level behaviors presented in section 3.6, but do not go into detail for all the different reactive behaviors that should be implemented within the high-level behaviors for handling the relevant risks.

3.8.1 Finite State Machine

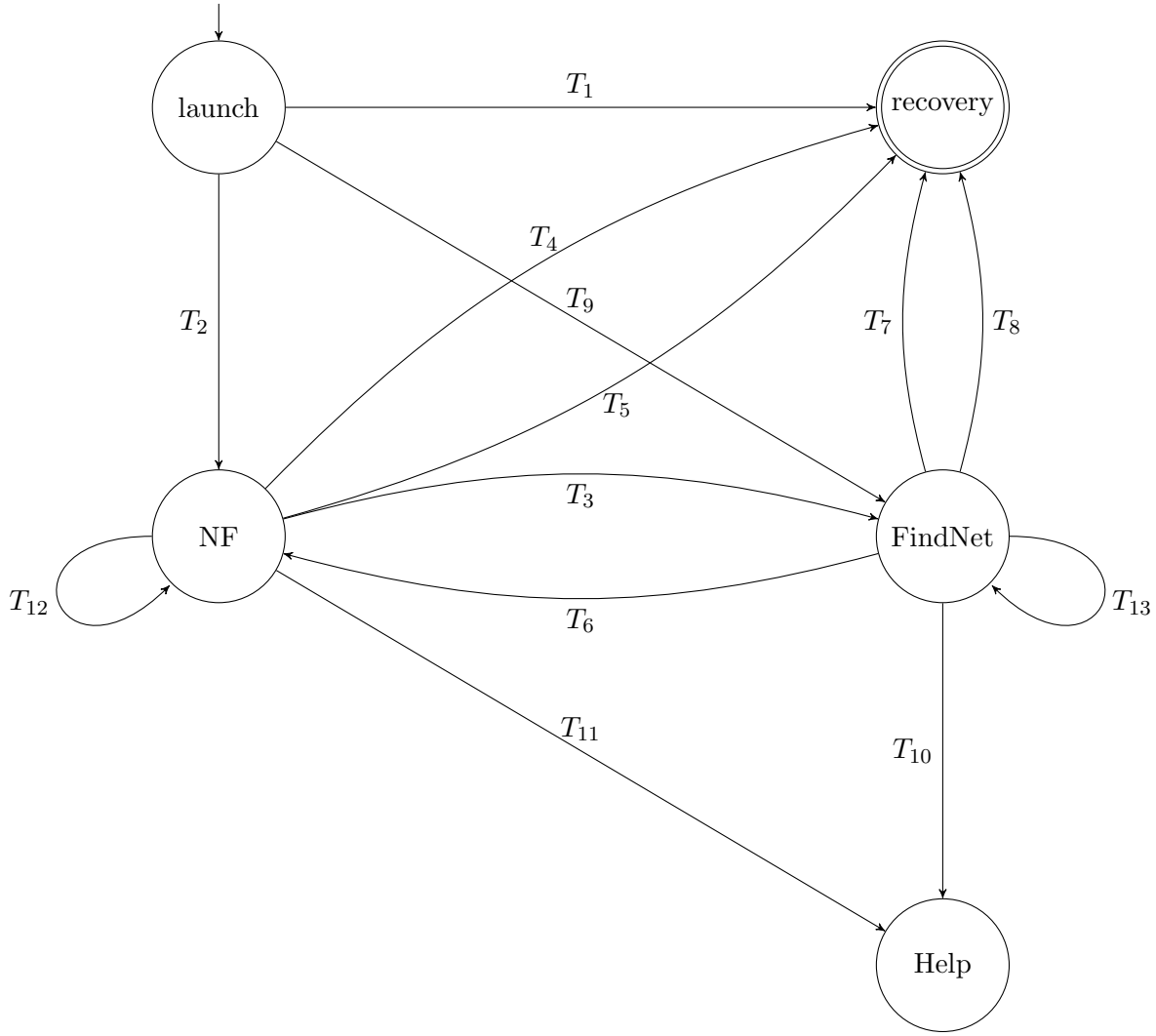

Figure 3.5: A Finite State Machine for conducting an autonomous net inspection

3.8.2 Behavior Tree

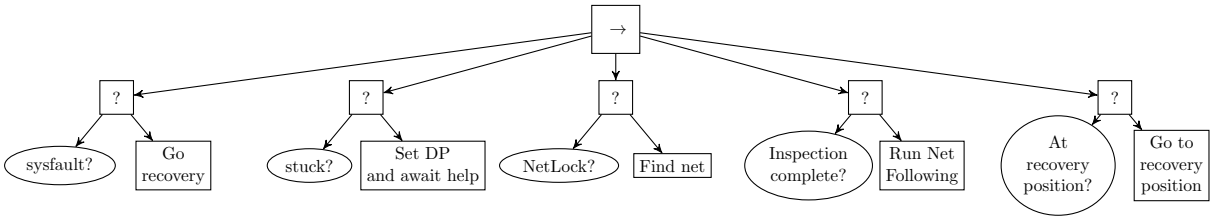

Figure 3.6: A behavior tree for conducting an autonomous net inspection

T_1	Systems failure $\rightsquigarrow$ Go to recovery
T_2	Net detected $\rightsquigarrow$ Start net-following
T_3	Net lost $\rightsquigarrow$ Look for net
T_4	Systems failure $\rightsquigarrow$ Go to recovery
T_5	Net inspection complete $\rightsquigarrow$ Go to recovery
T_6	Net found $\rightsquigarrow$ Continue net-following
T_7	Systems failure $\rightsquigarrow$ Go to recovery
T_8	No net found within timeout $\rightsquigarrow$ Go to recovery
T_9	No net detected $\rightsquigarrow$ Look for net
T_{10}	Unable to move $\rightsquigarrow$ Set DP and await operator interaction
T_{11}	Unable to move $\rightsquigarrow$ Set DP and await operator interaction
T_{12}	Net inspection not complete $\rightsquigarrow$ Continue net-following

Table 3.1: The transitions of the FSM in Figure 3.5

Chapter 4

Iterative Implementation Process

A selection of the operational modes and behaviors presented in section 3.6 has been implemented for SINTEF Ocean's Argus Mini ROV. The implementation of these modes and behaviors was completed in several iterations. First, a proof of concept mission control system was implemented and tested by simulation in FhSim. Parts of this implementation were chosen for a field test in a real-world, full-scale aquaculture setting. Some alterations were made, and the second iteration of the mission control system was tested at SINTEF ACE Rataren, one of SINTEF Ocean's full-scale aquaculture laboratories. Further alterations were then made, taking the results from the field tests into consideration, and a final iteration of the mission control system was implemented. This was first tested in simulations before being tested in full scale at SINTEF ACE Tristeinen. Unfortunately, the final experiment was cut short due to a local COVID-19 outbreak, but it still yielded some results.

In section 3.6, the words "mode" and "behavior" were used for describing different parts of the mission control system. Modes are of a higher level; as mentioned in section 3.5, modes can be seen and chosen in the GUI. Modes would also be important for a fully autonomous system as a large part of the deliberative choices conducted in the planning layer would be based on the active mode. In the case of SINTEF Ocean's Argus Mini ROV and its control system, the different modes would be implemented in Aqueous and call lower-level behaviors implemented in FhSim. Most of the lower-level behaviors would be similar to what can be found in classic control systems and suitable for implementation in FhSim. However, some might require external toolboxes or software, and implementation into Aqueous would be more suitable. Generally, it can be said that the planning layer is implemented in Aqueous and the behavioral layer in FhSim. The executive layer will be split between Aqueous and FhSim; where FhSim would handle the execution of reactive and less deliberative behaviors, while Aqueous would handle the execution of more general and deliberative behaviors.

In this thesis, it was chosen to implement an autonomous mission control system capable of performing a fully autonomous net inspection. This meant expanding the existing net-following mode into a net-inspection mode. The net-following mode developed in [4] was used as the basis for further extensions in this project. The reason for choosing to implement a net inspection mode was that the net-following mode had already been established. It also had the potential for implementing both reactive and deliberative behaviors. Several parts of a fully autonomous net inspection mode could also be reused by other modes for fulfilling

other tasks.

4.1 The Initial Implementation of the Hybrid Control Architecture

In the initial implementation of this thesis, two reactive behaviors and two deliberative behaviors were implemented in Aqueous. The first and most straightforward of the two reactive behaviors handled the situation where the distance between the ROV and the net went below some predefined threshold. The second reactive behavior handled the case where the DVL lost net-lock. Both the reactive behaviors were implemented within the same function in the Aqueous control app. When it comes to deliberative behaviors, the first was a behavior for tracking the progress of the inspection in order to do some basic path planning, and the second was for locating the net and achieving net-lock.

4.1.1 Reactive Behaviors

The control app uses a `setInterval`-function for listening to data from the ROV at given intervals. The interval has a length of 300 milliseconds, meaning that the data updates every 300 milliseconds. Another such `setInterval`-function with a length of 1 second was implemented for checking if the ROV was in a high-risk situation. Inside the interval, a function for detecting high-risk situations is called. In the initial implementation, this function contained the two reactive behaviors.

Distance Below Threshold

This behavior receives the measured distance between the ROV and the net and checks if it is below some predefined threshold. This threshold was chosen to be 1 meter for the initial implementation. If the distance is below the predefined threshold, the behavior will add another meter to the desired distance in the net-following controller. To avoid adding 1 meter several times during the same violation of the threshold the implemented behavior also checks if the time since the last violation is larger than 5 seconds. This time is counted up every time the high-risk detection function is run, i.e., every second. If the threshold is exceeded, the time is reset to zero.

Lost Net-Lock

Aqueous receives a boolean flag from the DVL, indicating whether the DVL has net-lock or not. This flag is checked, and if net-lock is available, the current position is saved as a last known point with net-lock. Simultaneously, a counter which will count the seconds without net-lock is reset to zero. If the DVL does not have net-lock, FhSim will automatically switch to manual mode with autoheading and autodepth activated, and the counter will accumulate time since the DVL last had net-lock. A function in Aqueous for switching to manual mode when this happens was removed in order to resume net-following if the net-lock returned. This leaves Aqueous in the net-inspection mode, while FhSim is in manual mode. Hence, if the DVL regains the net-lock while FhSim is in manual mode, Aqueous will resume the inspection of the net by switching FhSim back to net-following. However, if the DVL does not regain the net-lock within a predetermined period, it will trigger Aqueous to switch to

the DP mode. This will then cause FhSim to also switch to the DP mode. For assisting the operator with resuming the net inspection, the last known position with net-lock will be set as the desired position for the DP mode. In the initial implementation, the predetermined period was chosen to be 10 seconds. The mission control system will keep the ROV in DP until an operator manually switches modes. This could be changed in the future, such that the mission control system would resume the net inspection if net-lock is regained while in DP. It would require changes in the DP mode such that FhSim could be running the low-level DP behavior while Aqueous stays in the net inspection mode.

4.1.2 Deliberative Behaviors

The deliberative behaviors were implemented as individual functions in the net-following mode, which is a sub-app of the control app. The behavior for tracking the inspection progress relies on a `setInterval`-function with an interval of 500 milliseconds for keeping track of the time. The behavior for locating the net is called when the operator manually switches into the net inspection mode.

Track Inspection Progress

In the `setInterval`-function, two conditions are checked. Firstly the flag indicating if the DVL has net-lock is checked, and secondly, it is checked that the IMC message containing the data regarding the current net-following state of the ROV is not undefined. If both conditions succeed, an initial net-heading will be saved, and a function for tracking the inspection progress will be called. This function firstly increments a variable that counts how much time has passed since the current loop of the net started. It then calculates the difference between the current net-heading and the initial net-heading. This difference is used for determining the location of the ROV relative to the starting point of the inspection. If the difference is less than 5° and the time since starting the current circumnavigation is above 30 seconds, the time counting variable will be reset, and the depth will be increased by 2 meters. The different threshold-values for the initial implementation were chosen based on trial and error in simulations and not real-world experience. The ROV will continue to circumnavigate the net cage and go deeper for each completed circumnavigation of the net. No end-condition was implemented in the initial implementation of this behavior.

Locate Net and Achieve Net-Lock

Find net was implemented as a function that would be called when switching to the net inspection mode while no net-lock was available. Previously, if the operator attempted to switch to the net-following mode while the DVL had no net-lock, it would simply return nothing and stay in the mode it was already in. This was removed for the implementation of the function for locating the net. Instead, Aqueous is switched to the net inspection mode, and the desired depth is inherited from the ROVs' current depth. It then calls the function for locating the net. In this function, a while-loop runs for 20 seconds or until the DVL gains net-lock. The heading towards the origin is calculated inside the while-loop and, auto-heading is enabled with this heading. Autodepth is set to 6 meters. For the autodepth and autoheading to function, the mode is switched to manual mode. The ROV will then turn towards the origin and gradually go towards a depth of 6 meters. If net-lock is achieved, the loop will quit, and the net-following behavior will be activated. Aqueous will then switch back to the

net inspection mode. If no net-lock is achieved before the while-loop has completed the 20 seconds, the ROV and Aqueous will be set to DP in the current position, awaiting interaction from the operator. This will be the case if the ROV is located at the side of the net cage that lies furthest from the service vessel/origin.

The reason for choosing to point the ROV towards the origin when attempting to locate the net is that the origin would usually be placed in the position of the service vessel that launched the ROV, which would be outside of the net cage. Hence, if the ROV is placed in the net cage from the service vessel, there should only be a small distance between the ROV and the service vessel, and they would be separated by the net. However, the behavior for locating the net could be used in other situations or areas of the net cage, and the behavior should rely on something else for locating the net. This will be further discussed in section 4.2. The reason for choosing a depth of 6 meters is that the beams from the forward-facing DVL are not pointing straight forward; two of the beams are pointing slightly upward and the other two slightly downward This can cause the two upward pointing beams to point over the top of the net cage, such that no net-lock is achieved.

4.2 Initial Implementation - Results and Discussion

Several simulations with the initial implementation were conducted; however, this section will focus on a single simulation that tested all the implemented behaviors. To simulate DVL losses due to fish swimming between the ROV and the net, a basic signal modifier was implemented in FhSim. This modifier is a pulse multiplied with the DVL signal such that the signal is the actual DVL signal for 40 seconds and then 0 for 5 seconds. The desired velocity for the net-following mode to traverse the net was set to 1.5 ms. This velocity is much higher than intended and extends the expected ability of the controller; SINTEF Ocean ordinarily keeps a velocity of under 0.3 ms [3]. Setting the desired velocity substantially higher than what is ordinary was done to provoke situations that the reactive behaviors would need to handle, such as the ROV coming too close to the net. The desired distance was changed during the simulation to provoke the distance below threshold behavior several times. For the majority of the simulation, the desired distance was set to 3 meters. The result of the simulation can be seen in Figure 4.1. The ROVs track is represented in yellow, where the red sections indicate that the DVL lacks net-lock while the yellow parts indicate that the DVL has net-lock. The blue grid represents the net cage.

4.2.1 Results

Locate Net and Achieve Net-Lock

After the simulation was started, Aqueous was toggled to the net-inspection mode. As the ROV was hovering at the surface, facing away from the net, it did not have net-lock, and the find net behavior was initiated. The find net behavior set the desired heading of the ROV such that it was directed towards the origin, as well as increasing the desired depth to 6 meters. The ROV then started simultaneously rotating towards the origin and increasing its depth. Before the desired depth of 6 meters was reached, the ROV was facing the origin and,

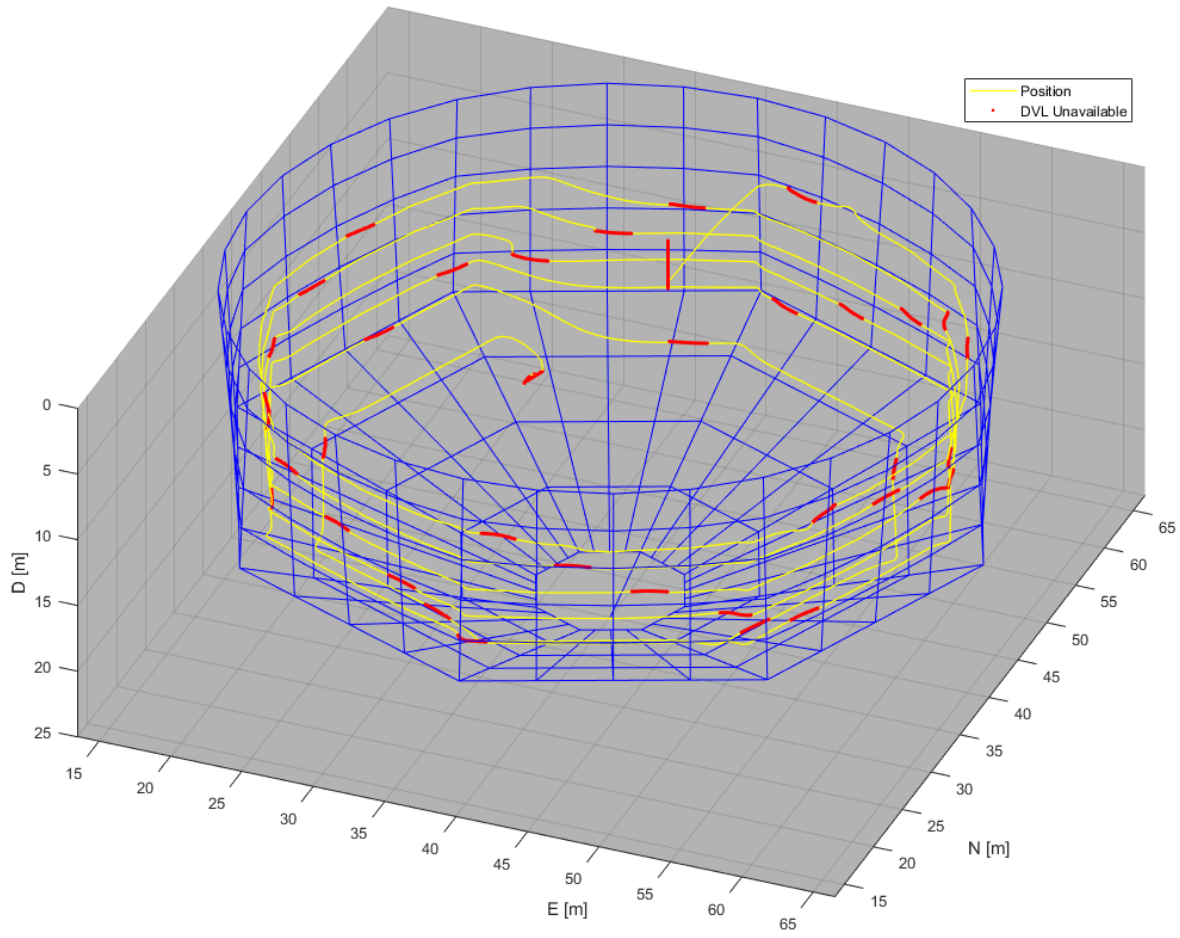

Figure 4.1: The full plot of the simulation illustrating the different results of the initial implementation

by that, also the net. At around 5 meters of depth, the DVL was able to achieve net-lock, and Aqueous signaled FhSim to start the net-following controller.

Distance Below Threshold

The net-following controller successfully traversed the net, and within a short amount of time, the ROV was situated 1.5 meters from the net as initially desired. Due to the high velocity, the ROV was unable to keep the desired distance of 1.5 meters, and the threshold of 1 meter was exceeded. The desired distance was then automatically increased to 2.5 meters, and the ROV continued to traverse the net. To provoke the behavior to run again the desired distance was manually reduced to 1 meter. As expected, this caused the ROV to get closer to the net than 1 meter. The desired distance was automatically changed to 2 meters. Due to the high desired velocity, the desired distance of 2 meters still left the ROV close to 1 meter from the net on occasions such as the corners of the net cage. After a little while, the ROV was within 1 meter of the net, and the threshold was exceeded again. This caused the final automatic update of the desired distance, which was set to 3 meters. It was left at 3 meters for the remainder of the simulation.

Lost Net-Lock

Simultaneously as the threshold behavior ensures a safe distance between the ROV and the net, fish swimming between the DVL and the net was simulated. This caused the DVL to lose the approximation of the net, leaving it impossible for the net-following controller to continue traversing the net. When the DVL lost net-lock, FhSim automatically switched from the net-following controller to manual mode. However, Aqueous was still kept in the net inspection mode, causing the net-following controller to be reactivated as soon as the DVL regained net-lock. This functioned as desired, allowing the ROV to continue the inspection even though net-lock was lost at regular intervals. None of the simulated losses of net-lock were long enough to cause the switch to the DP mode. However, in the final stages of the simulation, the ROV reached the bottom of the net cage, and the DVL was unable to achieve net-lock. After ten successive seconds without net-lock, Aqueous switched to the DP mode, and the ROV moved slightly back to the last position where the DVL had net-lock. The simulated inspection was then terminated due to the inspection being complete.

Track Inspection Progress

The final behavior tested in the simulation was the behavior for tracking the inspection progress and creating a path that allowed the ROV to fully inspect the net cage. When the ROV had circumnavigated the net, i.e., the heading of the estimated net plane was within $\approx 5^\circ$ of the initial net heading, and some time had passed since the last time the ROV was at this heading, the behavior increased the depth by 2 meters. This happened every new circumnavigation of the net cage, although there were some variations in where the depth change would occur. As no exit condition was implemented, the inspection continued until the ROV reached the bottom of the net cage, and the DVL was no longer able to achieve net-lock.

4.2.2 Discussion

In general, all the behaviors implemented in the initial implementation were able to achieve what they were designed to do, and some performed quite well in doing so. The reactive behaviors were able to quickly react to undesired situations and keep the ROV in safe locations. The deliberative behaviors were able to ensure that a full net inspection was conducted. However, all behaviors also had room for improvement, with the deliberative behaviors having the most significant potential. The reactive behaviors could benefit from being implemented in FhSim, such that the delay between sensing and acting is reduced. The autonomous mission control system could potentially be improved by expanding the reactive behaviors to inform the deliberative behaviors of the situation, such that the deliberative behaviors can make better decisions. However, this is not necessarily a requirement considering that the behaviors are able to achieve their goals. The deliberative behaviors in the initial implementation only functioned for the special circumstances defined in the simulation. The implementation of deliberative behaviors requires a great understanding of the different problems and how they can be solved autonomously. In the initial implementation, the author had little to no experience with ROV operations in aquaculture, and while the implemented behaviors performed decently in the simulation, several challenges would arise if these behaviors were to be tested

in a real-world aquaculture setting. Both the reactive and the deliberative behaviors could benefit from more tuning.

Distance Below Threshold

The reactive behavior for assuring that the distance between the ROV and the net is above a predefined threshold worked as intended. The exact value of the threshold, the size of the increase in desired distance, and how much time the ROV is given to move away from the net could be changed for improving the behavior. These changes should be conducted by someone with experience in using ROVs in aquaculture or by extensive testing.

The behavior could be made to be more responsive by adding a separate, more responsive controller for avoiding contact with the net. It should also be considered that keeping a safe distance does not necessarily mean the inspection will be conducted successfully. In situations with low visibility, the ROV might need to be closer to the net to achieve acceptable video footage for the operator or the computer vision algorithm to find faults in the net.

Lastly, if the net-following controller is unable to counteract the current and the ROV constantly stays too close to the net regardless of desired distance, the reactive behavior could relay this information to the deliberative behaviors, such that they could change the goal or abort the mission.

In general, this behavior is relatively straightforward and could be improved by incorporating different measurements such as velocity towards the net, velocity parallel to the net, history of distances, amount of exceedances, filtering of the input signal, etc. However, by incorporating several different measurements, the complexity of the behavior is quickly increased. This balance between complexity and responsiveness must be managed.

Lost Net-Lock

The reactive behavior for managing lost net-lock also worked as intended. During the simulated 5 second periods without net-lock, it kept a relatively stable position, and when the net-lock returned, it resumed with the net-following. In periods without net-lock for above 10 seconds, the behavior switched to the DP mode with the last known position with net-lock as the set point. The length of the period without net-lock before DP should be set would need to be decided by someone with experience of using ROVs in aquaculture or by extensive testing. While the behavior for managing lost net-lock performs as intended, it could still be improved.

The behavior could account for the frequency of net-lock losses, informing the operator if losses happen too often to perform an acceptable inspection. This could happen if the fish starts swimming between the ROV and the net, causing a moving wall of fish between the ROV and the net, which again causes the DVL to lose its net-lock frequently.

In the initial implementation, it was decided that if a 10 second period had passed without net-lock the ROV should use DP mode for going to the last known position with net-lock. It would then stay in that position and await instructions from the operator. A different solution could be to let the ROV continue with net-following if the DVL regains net-lock while

in DP. In this thesis, this was decided against, the reasoning being that if the DVL has been without net-lock for more than 10 seconds, the problem is likely to be something that will not disappear on its own. At the end of the simulation from subsection 4.2.1, the ROV reached the bottom of the net cage, and the DVL was unable to estimate the position of the net. If the ROV were to continue traversing the net from the position where it regained net-lock, it would quickly encounter the same problem which caused the loss of net-lock in the first place.

Another possibility could be using the locate net behavior if no net-lock is acquired after the ROV reaches the DP setpoint. I.e., if the ROV is within a certain distance of the set point and no net-lock has been acquired, the behavior for locating the net and achieving net-lock is called. However, this would rely on a more advanced behavior for locating the net than what was implemented in the initial implementation in this thesis.

Similar to the other reactive behavior, increased use of other measurements and extra calculations lead to increased complexity and reduced responsiveness.

Track Inspection Progress

The behavior for tracking the inspection progress and ensuring the inspection of the entire net cage performs well in the simulated environment. It can inspect the entirety of the net with no interaction from the operator required until the ROV reaches the bottom of the net cage. By including a criterion for ending the inspection, the behavior could switch to a tracking mode and return the ROV to the surface for recovery. This criterion could simply be the expected depth of the net cage such that the inspection would end as soon as the ROV reaches the bottom of the net cage. Other means of measuring the inspection progress are possible and requires more advanced path planning and possibly accurate localization and mapping relative to the net.

While the behavior worked as intended in the simulation, several new challenges occur when the ROV is placed in a real net cage.

First, the net cage will often deform such that the shape differs from the perfect cylindrical shape seen in the simulation. This could lead to different sections of the net with the same net-heading, which would lead to the progress tracking behavior believing that the ROV has completed a circumnavigation of the net. Better tracking of the inspection progress should therefore be found. This could be achieved by including a positioning system such as USBL for tracking the inspection progress. However, this would require previous knowledge of the position of the net cage or a more advanced simultaneous localization and mapping (SLAM) system. Slightly better tracking could be achieved by utilizing parameters such as the circumference, diameter, and depth of the net cage. These could be used for checking if the distances traveled by the ROV during the inspection corresponds with the length of a circumnavigation of the net. Another possibility could be a video-based system that keeps track of the seams in the net and compares this to the known amount of seams. This method is sometimes used by ROV operators when performing net inspections manually. [3]

Another issue that would arise in an actual net cage is the issue of different obstacles in the net cage. A behavior that does several circumnavigations of the net cage without switching

direction could require the ROV to repeat the track in the opposite direction to avoid getting the tether stuck. A different path is therefore required before this behavior will be useful in a real aquaculture setting. Finding an optimal path that avoids obstacles would require the position of the different obstacles to be known in advance. Initially, a more robust path could be developed and implemented without being an optimal path.

Locate Net and Achieve Net-Lock

The behavior for locating the net and achieving net-lock was only functional in specific parts of the net cage. In the simulation, the ROV was placed in a position near the net, and the rotation towards the origin would leave the ROV pointing at the net. Hence, the most essential function of the behavior was to sufficiently increase the depth of the ROV such that all the beams from the DVL were able to acquire net-lock. This would work and could be useful for situations where the ROV is positioned on the other side of the net for the service vessel acting as the origin of the coordinate system. However, if the ROV is placed in the middle of the net cage, it will probably need to move through the shoal of fish swimming in circles before it can acquire net-lock. For locating the net in a situation where the ROV is far away from the origin, the deliberative behavior must rely on different parameters for locating the net. If the DVL has previously had net-lock, the behavior could use the position of the ROV where it last had net-lock as a starting point for locating the net. Another possibility could be to have a rough map of the net cage relative to the origin, such that the ROV could estimate what would be the shortest distance to the net. Other methods for identifying the net could also be used, i.e., computer vision algorithms might be able to identify the net.

Another possible challenge will be if the net is moving. The net could be shifted away from where the behavior expects to find the net, and this would render some of the methods above less useful than if the net kept its position.

Overall Performance and Challenges

The mission control system implemented in the initial implementation was able to perform an autonomous net inspection. However, it was not fully autonomous as it needed interaction from the operator to confirm that the task was completed and would have required the operator to take the ROV back to the surface for recovery if it had been a real net inspection. Although the net inspection was not fully autonomous, it will still facilitate safer and more robust inspections. The initial implementation serves as a proof of concept for an autonomous mission control system and shows that it is possible with the provided tools.

It should be noted that the simulated environment was close to the ideal environment. The simulated net cage was perfectly stationary and did not deform during the simulation, there were no mooring lines or other obstacles in the net cage, and the location of the tether was not taken into account. Some effects of the fish were simulated, but the full influence of the fish on the ROV would be considerably challenging to simulate. Other environmental conditions such as waves and currents were also not simulated.

4.3 The Second Implementation of the Hybrid Control Architecture

For the second implementation, the mission control system would be tested in a real-life, full-scale aquaculture setting. This meant large changes to the initial implementation were required. Some of the behaviors from the first implementation were abandoned, while others received significantly more focus. The behaviors that were kept were the behavior for handling lost net-lock and the behavior for tracking the inspection progress and ensuring the entire net cage is inspected. The behavior for handling lost net-lock was first attempted implemented in FhSim instead of Aqueous; however, this required large changes in the communications between FhSim and Aqueous. It was therefore left approximately the same as in the previous implementation. The behavior for tracking the inspection progress received significant modifications. The location of the implementation of these behaviors did not change, and modifications were carried out within the existing functions from subsection 4.2.1.

The reactive behavior for preserving a minimum distance to the net was abandoned as the net-following controller can keep a safe distance to the net on most occasions. The problem would, therefore, only occur if the operator set the desired distance to be a small distance. The behavior could be more useful as a general collision avoidance behavior utilizing the DVL, but this would require the behavior to be completely rewritten.

The behavior for locating the net and achieving net-lock was abandoned due to it only functioning in special conditions where the ROV were located close to the origin but on the other side of the net. This would also require a significant amount of research and development before being suitable for use in an autonomous mission control system.

4.3.1 Track Inspection Progress

The changes in the behavior for tracking the inspection progress and ensuring the entire net cage is inspected were mainly changes to the path planning of the inspection. The function was still called the same way as in the initial implementation, and the same initial net-heading is used. But instead of checking if the current net-heading is within $\approx 5^\circ$ of the initial heading, the function checks if the difference is greater than 60°. If this is the case, the function will increase the desired depth by one meter, change the direction of the net-following and replace the initial net-heading with the current net-heading. This should create a pattern where the ROV moves back and forth between the initial net-heading and the net-heading 60° away from the initial heading. The pattern will continue until the ROV reaches the bottom and the DVL loses net-lock. As this causes the inspection to end, no proper end-condition was implemented in this implementation of the behavior.

4.4 Second Implementation - Results and Discussion

The second implementation was first tested in simulation before being tested in SINTEF Ocean's full-scale aquaculture laboratory SINTEF ACE Rataren.

4.4.1 Results

Simulation

The simulation setup was the same as the one used for the first implementation. It includes simulated losses of net-lock but no environmental loads. The 3D visualization of the simulated result can be seen in Figure 4.2. Another visualization combining the net-heading and the desired depth can be seen in Figure 4.3. This visualization shows the heading changing back and forth while the depth is increased. The heading has been shifted to avoid wrapping around $[-180°, 180°]$. The exact location of the switches in direction can be identified by examining the step-change in desired depth. The results from the simulation show that the

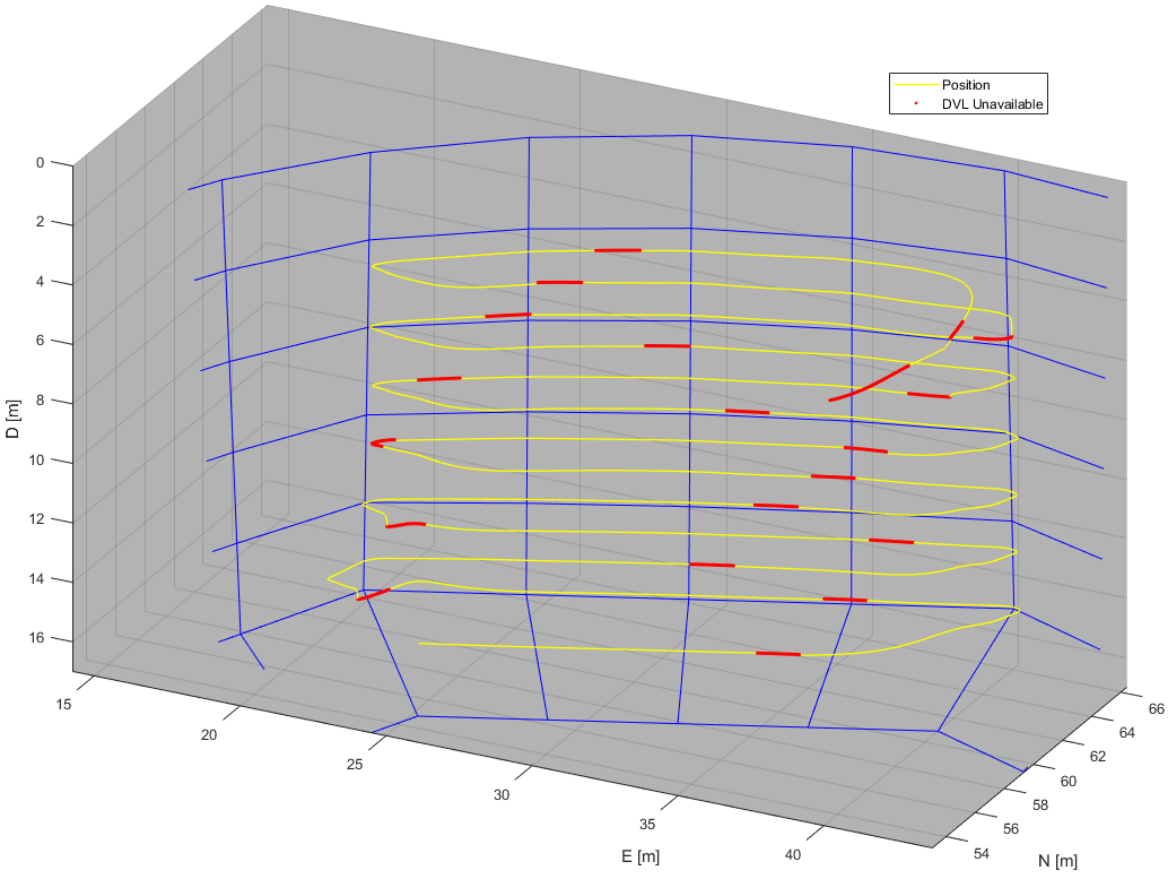

Figure 4.2: The simulated track of the ROV illustrating the back-and-forth pattern in the second implementation

mission control system is capable of producing a pattern which the ROV can follow. The ROV is able to follow this pattern even though the DVL regularly loses net-lock. The loss of net-lock can be identified as red points in Figure 4.2 and by the spikes going up to 180° in Figure 4.3. When net-lock is lost, the net-heading is set to zero, but for visualization purposes, these points were changed to be 180° in Figure 4.3. As the net-following behavior is disabled when the net-heading is set to zero, these headings are not used in the behavior for tracking the inspection progress.

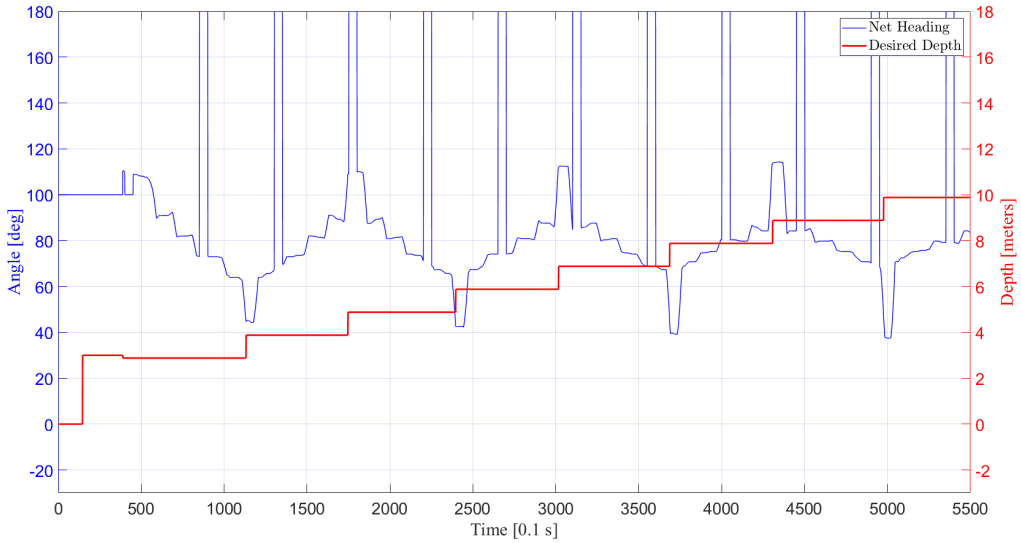

Figure 4.3: The estimated net-heading and the desired depth from the simulation of the back-and-forth pattern in the second implementation

It should be noted that when the desired depth is changed, the change in depth for the ROV is gradual. While some inaccuracies occur, they do not reduce the performance of the inspection. Based on this simulation, the autonomous mission control system was deemed ready for testing in a full-scale aquaculture laboratory.

Real World Test

The real-world test was conducted at SINTEF Ocean's full-scale aquaculture laboratory SINTEF ACE Rataren. During the test the wind was estimated to be ≈ 2 ms to 5 ms, the current was estimated to be ≈ 0.3 ms, and the waves were estimated to be ≈ 0.1 m to 0.2 m. These conditions are excellent conditions for a net inspection. The net cage chosen for the test contained salmon, which were considerably smaller than the target weight. A 3D visualization of the results from the field test can be seen in Figure 4.4. The position is from an extended Kalman filter, combining USBL measurements with measurements from the DVL, compass, depth sensor, and gyro. The red dots indicate positions where the DVL did not have net-lock. Note that the net cage in the figure was manually placed afterward and not extracted from the ROVs data. Another visualization combining the net heading and depth can be seen in Figure 4.5. Similarly to Figure 4.3, the spikes in net-heading hitting 180° indicates that net-lock is lost. The ROV was manually maneuvered to a position where the DVL had net-lock, and the net inspection mode was activated. It then started following the net as expected. As seen in Figure 4.5, the heading estimates based on the DVL contain a fair amount of noise. This noise causes a double switch around 130 s, caused by an upwards spike, shortly followed by a downwards spike. The ROV then increases its depth by 2 meters and goes on in the same direction until it reaches $\approx 60°$ from the net-heading in the downwards spike at around 240 s. It then switches, and the ROV starts moving in the opposite direction, decreasing the net-heading. The section from 240 s to 400 s function exactly as desired. Shortly after switching

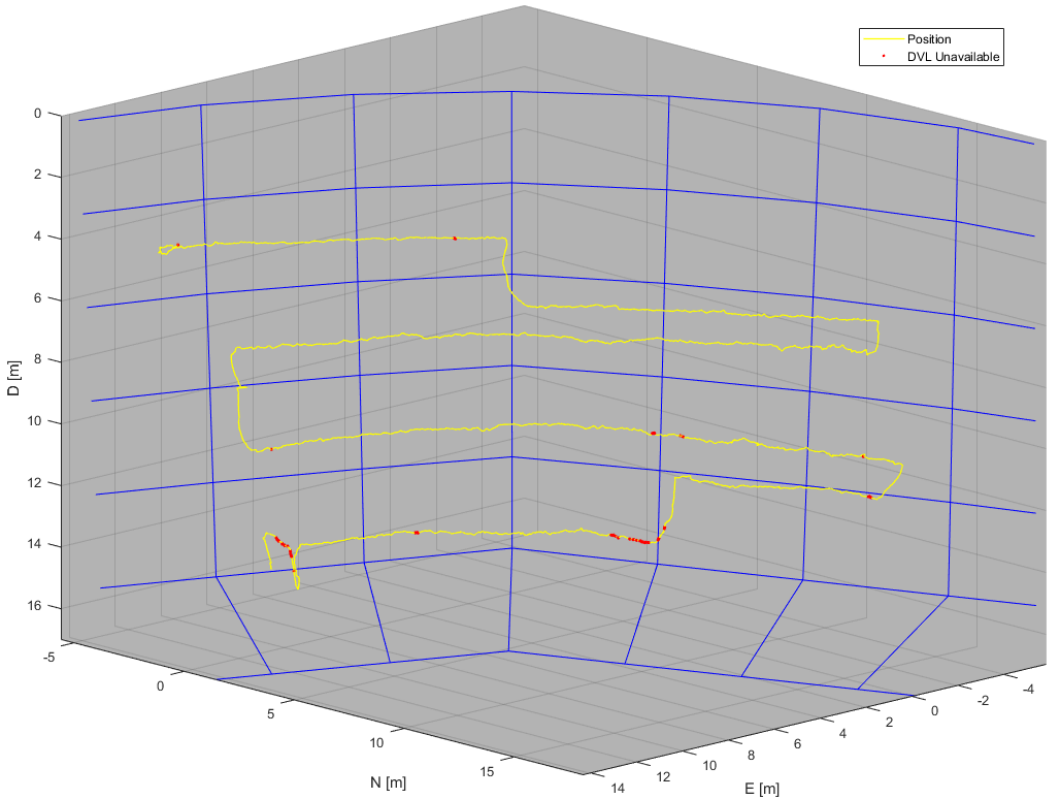

Figure 4.4: The track of the ROV from the field test of the second implementation

direction again, a loss of net-lock causes another double switch when the net-heading is set to zero and the value is read by the track inspection progress behavior in Aqueous. The reason why this occurred will be discussed in subsection 4.4.2. The following section from 410 s to 600 s is largely successful, even with some net-lock dropouts. Shortly into the next section, a group of fish comes between the ROV and the net. A screen capture of this can be seen in Figure 4.6. This causes not only a loss of net-lock but also some incorrect measurements, leading to an incorrect estimation of the net. The loss of net-lock and incorrect estimations causes another double switch. The ROV then continues successfully for a little while until a new stream of fish comes between the ROV and the net, causing another double switch. This time the ROV reached the depth where the net changes from vertical walls to inclined walls, and the inspection was aborted by the operator.

4.4.2 Discussion

The field test proved that the behavior for handling the loss of net-lock is capable of doing so without larger problems. The behavior set FhSim to manual mode, while Aqueous stayed in the net inspection mode. The ROV then hovers around in manual mode with autoheading and autodepth activated, with no significant drift. When the DVL regained net-lock, FhSim switched back to net-following, and the inspection continued from roughly the same position. The initial losses of net-lock only lasted for a few time steps, and an operator would not notice them without paying close attention. One of these short losses of net-lock caused a double

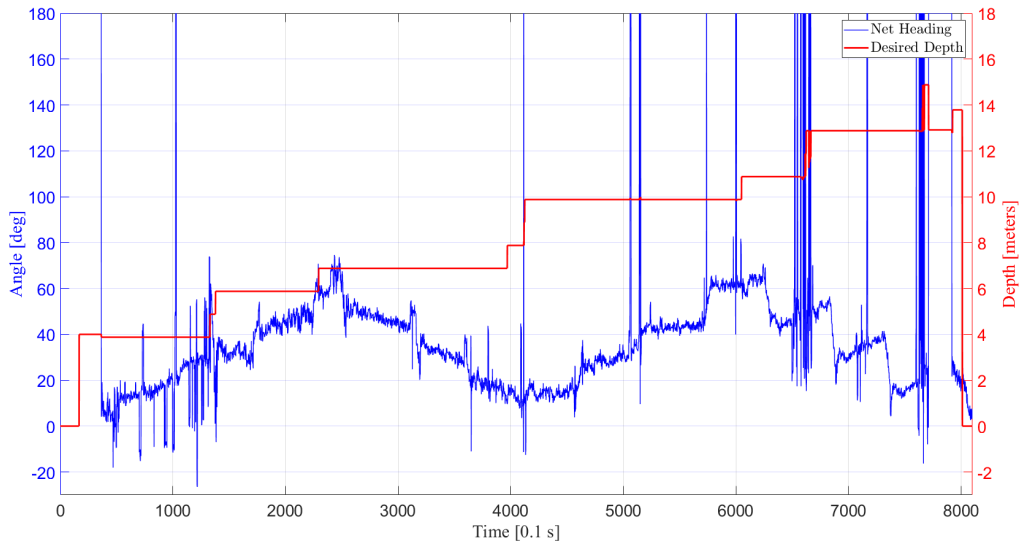

Figure 4.5: The estimated net-heading and the desired depth from the field test of the second implementation

switch, but this was due to a bug which will be discussed later and nothing was inherently wrong with the behavior for managing losses of net-lock.

The field test also proved the viability of the behavior for tracking the inspection progress. From the heading plot in Figure 4.5, it can be seen that the net-heading estimation is prone to noise. Several of the switches happened due to noise and not due to the actual difference in net-heading being larger than 60°. The cause of this noise could be inaccuracies in the DVL measurements, which again are caused by noise in the water column. DVLs are usually utilized in deeper waters for estimating the distance and relative speed to the sea floor. When operating in the wave zone, there is much more movement in the water. In addition, the structures of the fish farm can cause noise when waves slam into them or when moving parts rub into each other. As a result, the amount of noise inside the net cage is probably substantially more significant than what the DVL was designed for. A solution to this could be to introduce filtering for reducing the noise and smoothing out the measurements.

A couple of the switches in the field test came as a result of the DVL losing net-lock. This is caused by the net-heading being set to zero when the DVL loses net-lock. In the simulation, this did not cause the switch to happen, but when using the real hardware of the ROV, this became a problem. In the simulation, Aqueous and FhSim are perfectly in sync, meaning that if the DVL regains net-lock, FhSim outputs an estimated net-heading before Aqueous uses this heading to track the inspection progress. When using the real hardware of the ROV, there could be some delay as the different sensors and systems run with different refresh rates. This leads to the estimated net-heading being zero for a few time steps while the DVL reports that it has net-lock, causing Aqueous to use the incorrect value. This causes double switches because the incorrect value of zero often leads to a difference in net-heading over 60°. When the actual estimated net-heading returns, it will also give a difference in

Figure 4.6: A screen capture of the video, taken around 650 s into the inspection

net-heading over 60°. This could be solved by handling the special case where the behavior for tracking the inspection progress receives a net-heading of exactly zero.

Another thing that seems to cause switches in both the simulation and the field trial is the corners in the net cage. The net cage is not completely round but consists of flat panels sown together. The stitching points between these panels are often connected to the top and bottom rings of the net cage and used for stretching out the net, creating sharper corners. The net cage in the simulation is octagonal at the bottom and fully circular at the top. The net cage in the field trial consisted of a larger number of panels, but the corners still occur where the net is anchored to top or bottom rings. In an actual net cage, these panels might also have their own curves meaning that the net-heading can fluctuate throughout a panel that would have a consistent net-heading in simulation. This could also be solved by introducing a filter with a large window.

Some inaccuracies in the track could also be seen in the simulation, where the depth of the ROV gradually increased after switching direction. This could introduce small areas of the net that would not be covered by the video feed. The change was less gradual in the field test as the depth controller seems to get a higher priority. However, this could be further mitigated by utilizing the vertical net-following controller.

In summary, the second implementation proves that the 60°-transect pattern can be a viable part for fully inspecting the net. By repeating this pattern, the whole net cage could be inspected autonomously. However, the net-following itself needs better filtering to avoid unnecessary switches when inspecting the net in a real-life situation.

4.5 The Final Implementation of the Hybrid Control Architecture

The focus for the final implementation was to make the behavior for tracking the inspection progress more robust and able to do a full inspection of the net cage. The robustness was increased by introducing different filters and utilizing vertical net-following for the changes in depth. For the behavior to inspect the full net cage, an FSM was created to tie together several 60°-transect patterns.

4.5.1 Robustness Improvements

The results from the field test of the second implementation indicated that filtering could significantly reduce the chance of incorrect switches. A median filter was chosen as a median filter object was already implemented in FhSim and could be easily applied. The filter was implemented in the FhSim domain of the mission control system, which normally runs at 10 Hz. To see the effect of the median filter and decide on the filter window a median filter was applied to the results from the field test of the second implementation. The effect can be seen in Figure 4.7, Figure 4.8, Figure 4.9 and Figure 4.10

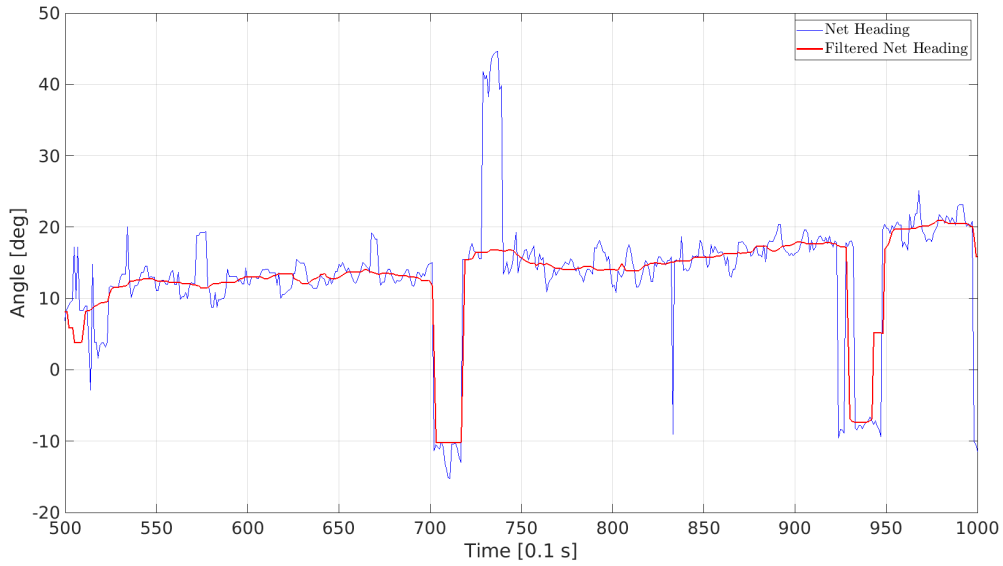

Figure 4.7: A section of the results from Figure 4.5 plotted against the results from a median filter with a window of 30 steps.

Figures 4.7 and 4.8 show two sections of the results from the field test in subsection 4.4.1, first represented in Figure 4.5. This is plotted against the result from the median filter with a window size of 30 time steps. Figures 4.9 and 4.10 show the same two sections of the results from the field test but plotted against the results of a median filter with a window of 100 time steps. Note that the delay these filters would introduce is not included as the filter is applied afterward and not in real-time. Also, note that these sections of the results from the field test were chosen because they contained noise that visualizes the effectiveness of the filters.

Large sections of the results were similar to Figure 4.7 but without the spikes. These were the sections where the behavior for tracking the inspection progress functioned as intended.

Initially, it was found that a window of 30 time steps, where one time step equals 0.1 seconds, would significantly improve the robustness. This would introduce a delay of three seconds, but as the ROV moving through the net cage has a low velocity, this was considered a small window. However, as can be seen in 4.7 and 4.8, there were still some inaccurate measurements that had a length of more than 30 time steps. A window of 100 time steps was then considered for further improving the robustness. This gives a 10-second delay, but this was also deemed acceptable due to the low velocity of the ROV. The results seen in 4.9 and 4.10 showed great promise.

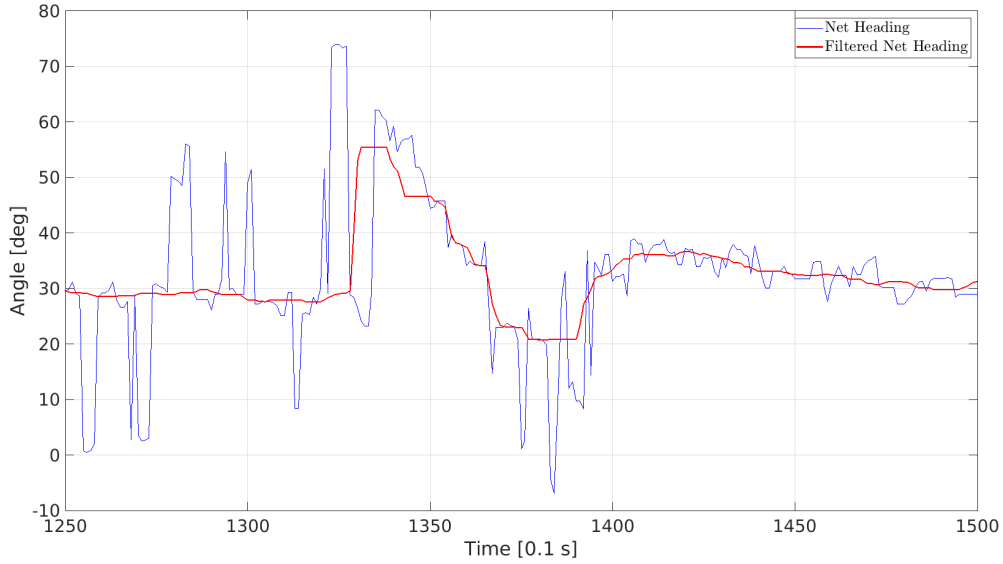

Figure 4.8: A different section of the results from Figure 4.5 plotted against the results from a median filter with a window of 30 steps.

To avoid the gradual changes in depth and improve the net-following during these changes, it was decided to utilize vertical net-following for increasing the depth at the end of a 60°-transect. This was done from Aqueous by switching from the net-following controller in FhSim to the vertical direction. While the direction of net-following is vertical, the behavior for tracking the inspection in Aqueous does not track changes in net-heading, but only track changes in the depth. When the depth of the ROV is within 0.25 meters from the desired depth, the direction of the net-following controller is switched from vertical to horizontal.

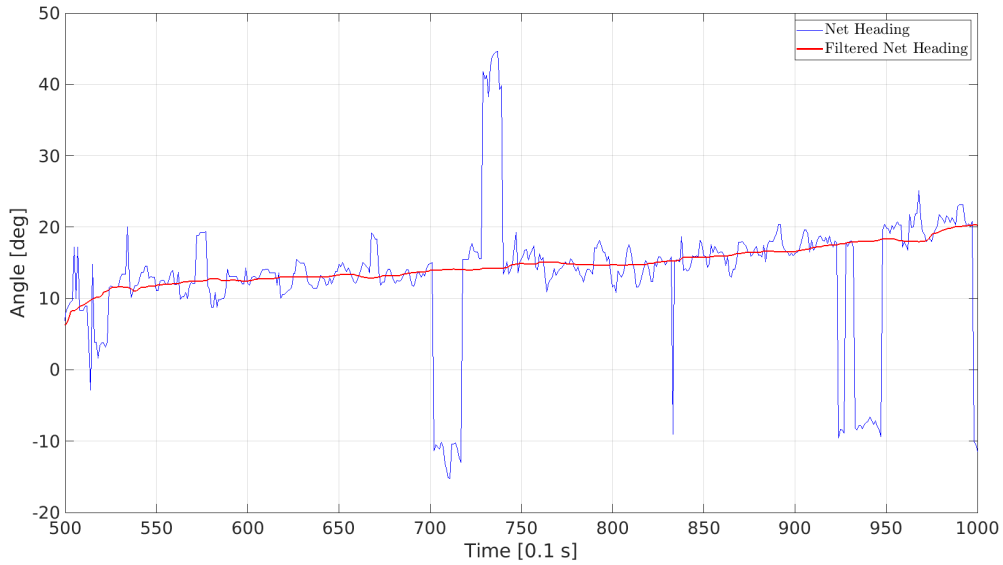

Figure 4.9: The same section of the results from Figure 4.5 as in Figure 4.7 plotted against the results from a median filter with a window of 100 steps.

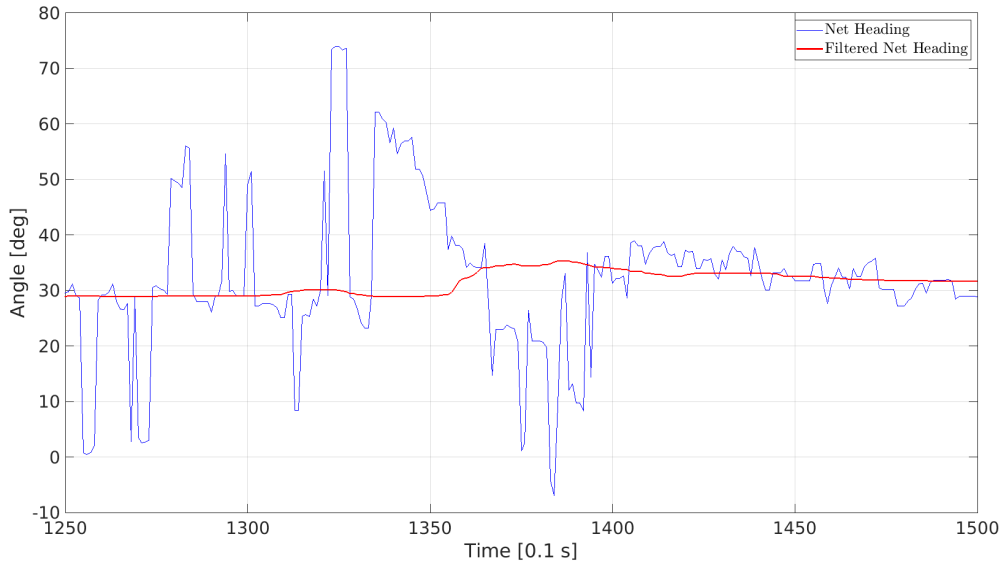

Figure 4.10: The same section of the results from Figure 4.5 as in Figure 4.8 plotted against the results from a median filter with a window of 100 steps.

4.5.2 Inspecting the Entire Net Cage

While the behavior in the first implementation was able to inspect the entirety of the net cage, the inspection pattern left the ROV highly likely to get its tether stuck or tied around

lines or other obstacles in the net cage. Therefore, part of the goal of the new implementation was to inspect the net cage such that the ROV would avoid getting its tether stuck.

A solution to this was to split the inspection into 60° sections, where three sections together would equal 180° and thereby half the net cage. The sections would have a minimum and maximum depth, which would keep the ROV slightly under the surface and avoiding the bottom of the net cage. After inspecting half the net cage, the ROV could return to the initial heading (the point where it started the inspection) and restart the inspection in the other direction.

By avoiding doing a full circumnavigation of the net cage, the risk of getting the tether stuck would be significantly reduced. To achieve this the, path planning algorithm for creating the 60°-transect patterns was expanded. As the algorithm for achieving the 60°-transects was already implemented with conditional logic, it was natural to continue using conditional logic for expanding it. Different conditions were added at the end of a 60°-transect. The ROV finishes a 60°-transect when the difference in net-heading between the current net-heading and a previously saved net heading is above 60°.

If the ROV finishes a 60°-transect between the minimum and maximum depth, the process is the same as in the second implementation. In this case, the function increases the desired depth by one meter, changes the direction of the net-following, and adds or subtracts 60° to/from the variable containing the previous net-heading, depending on the current direction. This was changed from using the current estimated net-heading as this had some inaccuracies and could cause inconsistencies in where the transects ended.

If the ROV finishes a 60°-transect shallower than the minimum depth or deeper than the maximum depth, there are several new conditional statements. A counter keeps track of whether the ROV needs to move another 60° before it can change depth and start a new transect. If this counter is even, the ROV can change depth and start a new transect. If it is odd, it needs to move further, which is achieved by changing the previous heading variable with 60° in the direction the ROV is moving. The behavior then compares the current direction with the initial direction of the inspection to figure out if it should move further in the same direction or if it should switch direction and move 120°. If the ROV is to move 120°, the previous net-heading variable will be changed, and the counter will be reset such that the same check will be done again after a 60°-transect. If the direction is the same as the initial directing, the behavior also checks if the difference between the current net-heading and the initial net-heading is larger than ±160°.

This indicates that the ROV has inspected half the net cage, and a boolean variable indicating that the ROV should return to the initial position is set to true. As long as this variable is true, the ROV will only move back towards the initial position. When the estimated net-heading is within ±7.5° of the initial net-heading the variable is set to false, and the initial direction is flipped.

A counter keeping track of how many times the ROV has been back at the initial position is also increased. If this counter reaches two, it means the ROV has conducted two 180° inspections of the net cage and returned to start. Hence, the entire net cage has been inspected,

and the behavior switches the mode of the ROV to manual as the net inspection is finished. For an easier understanding of this path planning algorithm, an FSM was created.

FSM for Net Inspection

The FSM covers the main logic of the autonomous net inspection mode. It can be seen in Figure 4.11, and the transitions can be seen in Table 4.1. In the diagram, H-NF is short for Horizontal Net-Following, V-NF is short for Vertical Net-Following, and RtS is short for Return to Start. It could be a part of either the net inspection state in the FSM or the net inspection action in the BT. Reactive behaviors such as switching to manual while waiting for net-lock and setting DP if no net-lock is acquired are omitted as they are implemented within every state in the FSM and would require transitions from every state.

The choice to create an FSM was made when the conditional logic of the behavior for tracking the net inspection became hard to follow. Before the FSM above was developed, no explicit decision had been made regarding a mathematical model of the control system. Instead, simple conditional logic was expanded several times, reducing the readability and understandability for each added state or transition. The resulting FSM can be seen in Figure 4.11.

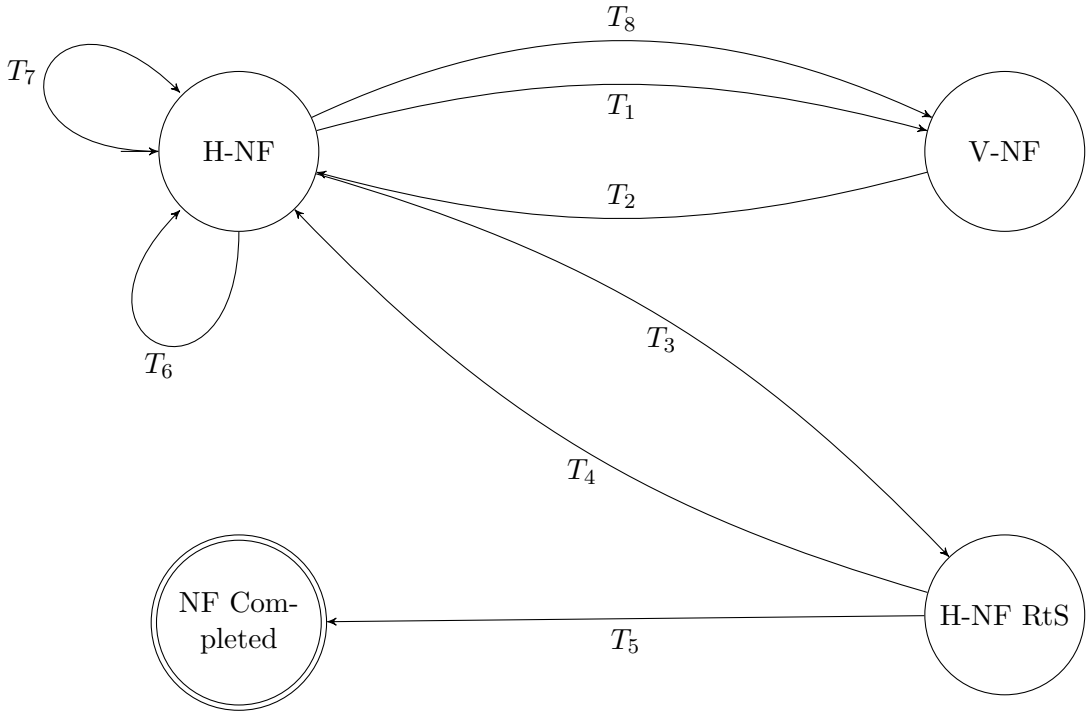

Figure 4.11: Finite State Machine for Net Inspection

T_1	NetHeadingDiff $> 60°$ && Depth inside min/max $\rightsquigarrow$ Change depth
T_2	Depth within 0.25 meters of the desired depth $\rightsquigarrow$ Start horizontal net-following, switch direction
T_3	Depth outside min/max && CurrentDir == InitialDir && InitHeadingDiff >= $160°$ $\rightsquigarrow$ Return to Start
T_4	InitHeadingDiff $< 7.5°$ $\rightsquigarrow$ Start new inspection in the other direction
T_5	InitHeadingDiff $< 7.5°$ && ReturnToStartCounter == 2 $\rightsquigarrow$ Net inspection complete
T_6	Depth outside min/max && GoFurther && CurrentDir == InitialDir $\rightsquigarrow$ Restart horizontal net-following
T_7	Depth outside min/max && GoFurther && CurrentDir != InitialDir $\rightsquigarrow$ Restart horizontal net-following, switch direction
T_8	Depth outside min/max && !GoFurther $\rightsquigarrow$ Change depth

Table 4.1: The transitions of the FSM in Figure 3.5

Chapter 5

Results

This section documents the results of the final implementation. The final implementation was first tested in simulation before being tested in the field. The results are therefore divided into results from the simulation and results from the field test. In all 3D figures, the track of the ROV is represented in red or white, where the red sections indicate that the DVL lacks net-lock while the white sections indicate that the DVL has net-lock. The blue grid represents the net cage. In the figures displaying the results from the field tests, the displayed net cage is placed by hand, and therefore only approximately the location of the actual net cage. In the 2D figures, the lack of net-lock is displayed as gaps in the net heading, and it can also be seen as the small changes in desired depth. The values used to plot the figures have been altered to avoid going through $[-180°, 180°]$, making the figures easier to follow. Some position has also been rotated around the z-axis to allow for better slices in the 3D plots. In all instances, the difference relative to other values has been kept the same, such that the results still convey the same information.

After conducting all the final experiments in this thesis, the author noticed a bug in the Moving Median Filter object in FhSim. The bug caused the input length to be discarded, and a default length of 5 time steps was always used. This was the case for all the experiments in this thesis that utilized the median filter. It was also the case for all the previous trials SINTEF Ocean had done, utilizing the median filter. Considering the proposed filter length in this thesis was 100 time steps, the length of 5 time steps gave little to no filtering. The filter also introduced some other issues and edge cases. This will be further discussed in chapter 6.

5.1 Simulation

The simulation setup used was mostly the same as in section 4.2. The velocity of the ROV was reduced to 0.5 ms, and although this is still higher than what is used in the real world, it makes little difference in the simulation. As the simulated environment and sensors had little to no noise, the main focus of the simulation was to demonstrate the ROVs' capability to inspect the entire net cage in a way that reduced the chances of getting stuck. The minimum depth was set to 2 m and the maximum depth was set to 8 m. The ROV was then supposed to inspect the entire net cage by doing three 60° sections in a row before returning to start and restarting the process in the other direction. This was achieved by positioning the ROV

such that it had net-lock and initiating the final implementation of the net-inspection mode. The following figures are from the same simulation, showing different parts of the simulation. Figure 5.1 shows the full net inspection, while Figure 5.2 shows the first half including the ROVs return to its initial position.

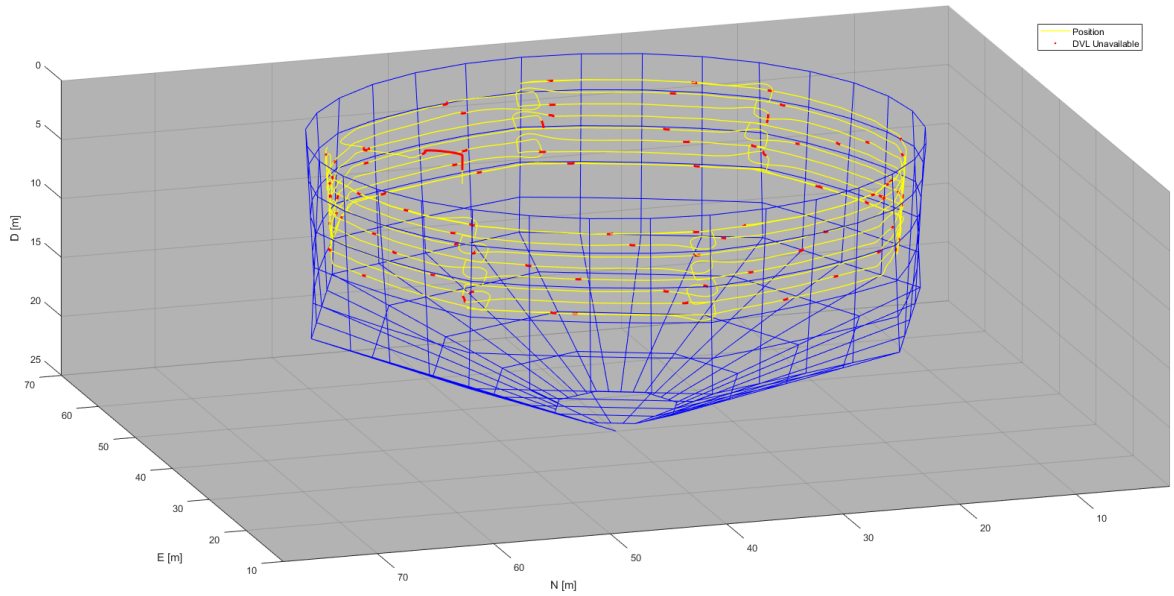

Figure 5.1: The full simulated net inspection

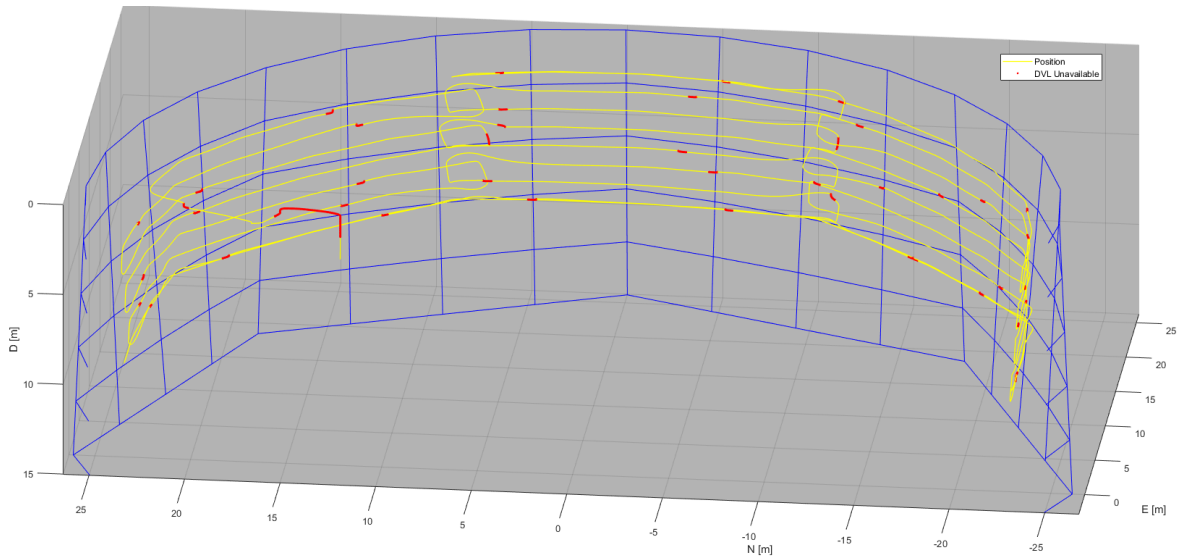

Figure 5.2: The first half of the full simulated net inspection

The ROV was able to fully inspect the net cage. The inspection can be split into two 180° sections, which both start and end in the same location. The first of the two sections can be seen in Figure 5.2. The ROV does not circumnavigate the net cage but still overlaps at the end of each 180° section such that every part of the net is inspected. Within each 180°

section, the ROV does three 60° sections before returning to the position where the section started. The track of the ROV continuing to the next section can be seen at the bottom left in Figure 5.2. Depending on the number of transects the ROV must do before reaching the minimum or maximum depth, the 60° sections could be ended on the opposite side of the section to where the next section begins. If this is the case, the ROV backtracks 60° to the point where the new section starts.

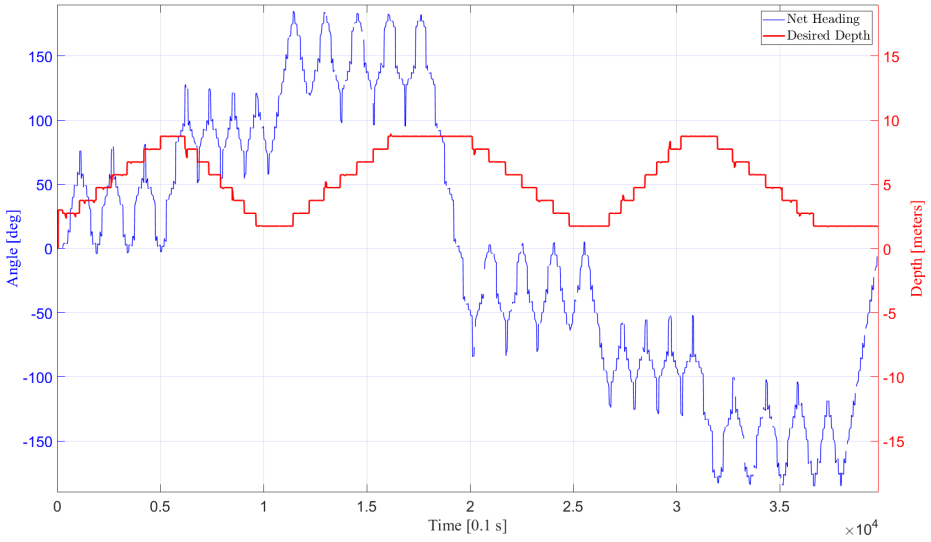

Figure 5.3: 2D plot of the heading and desired depth for the full simulation

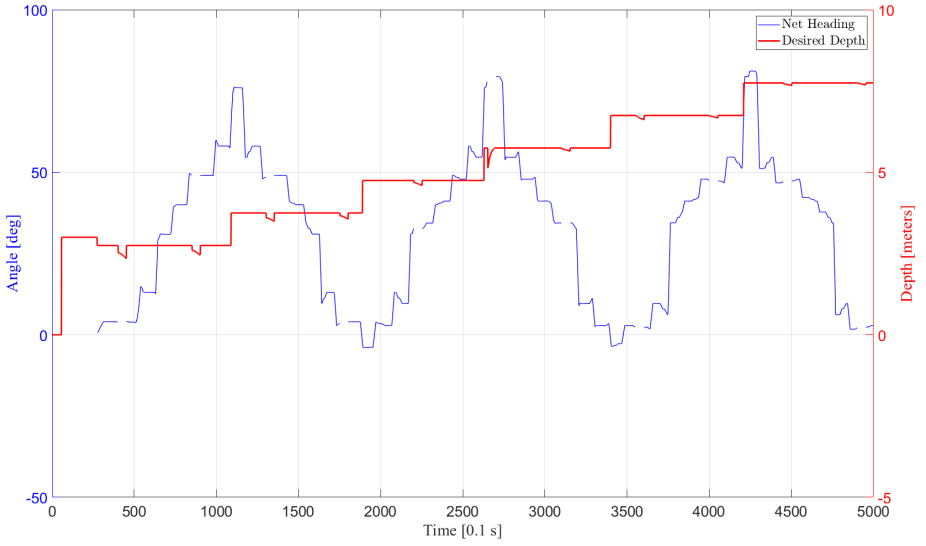

Figure 5.4: 2D plot of the heading and desired depth for one 60° section from the simulation

An example of this is displayed in the middle section in Figure 5.2, where the ROV moves towards the left until it finishes the 60° transect. The three sectors have a slight variation in length because of corners causing significant changes in the estimated net-heading. Figure 5.3 displays the heading and desired depth of the entire net inspection, and it represents how the plot should ideally look for a complete net inspection. Figure 5.4 displays the first 60° section, representing the desired plot of net-heading and desired depth during a 60° section.

5.2 Field

The field tests were conducted at SINTEF Ocean's full-scale aquaculture laboratory, SINTEF ACE Tristeinen. Tristeinen is a fish farm located among small islands and reefs off the coast to the north of Ørland Municipality. The site is located outside of sheltered waters and is more exposed to environmental forces than the average fish farm. The seas were calm (state 1) at the begging of the trial in the morning to then built up during the day (state 3). The weather during the field tests was also calmer than what would generally be expected for the area but similar to the waves, the wind and the current both increased during the day. The observed wave height increased from 0.2 m to 0.5 m, the wind increased from 3 ms to 6 ms and the current increased from 0.1 ms to 0.5 ms. The current was observed around buoys and other stationary structures in the water and could vary within the net cage as the net can influence the current. The weather was primarily coming from the north, which meant that the net cage chosen for the field trials would get negligible sheltering from other net cages and the surrounding islands.

The net cage chosen for the tests was estimated to contain approximately 190 000 individual salmon with an average weight of 2.45 kg, totaling approximately 467 tonnes of biomass.

Seven tests were conducted, but this section only presents the results from the second, fourth and fifth tests. The reason for omitting the first, sixth and seventh test is discussed in chapter 6; however, it can be summarized as follows: The first test started off successfully, but when it was supposed to switch depth, direction and start a new transect, it continuously changed depth. This was thought to be due to a bug in the code, and the test was aborted. The third test was omitted due to not providing new insights or situations which did not already occur in the second or fourth test. From the fifth test and onwards, the DVL had great difficulties achieving net-lock. The reason for this is suspected to be the harsher weather conditions causing underwater noise, which interfered with the acoustics of the DVL. This made it impractical to conduct tests relying on net-following, and the tests were nothing more than attempts at achieving a steady net-lock. All tests suffered from the bug in the median filter.

The tests were conducted by placing the ROV in the net cage, manually positioning it such that the DVL achieved net-lock, and activating the net inspection mode in Aqueous. The desired velocity was set to 0.1 ms for the first and second tests and increased to 0.2 ms for the remaining tests. The minimum depth was initially set to 2 m and the maximum depth to 8 m. They were both increased after the second test, the minimum depth to 2.8 m and the maximum depth to 10 m.

5.2.1 The Second Field Test

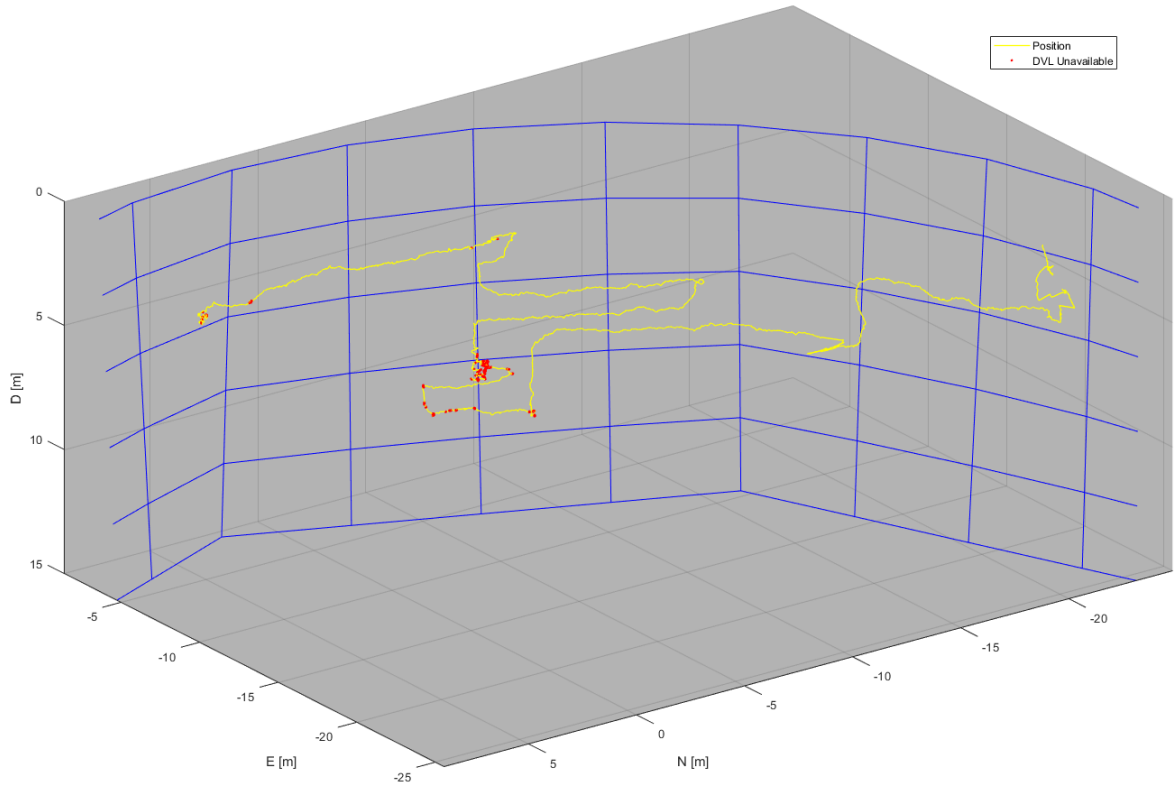

Figure 5.5: 3D plot showing the position of the second field test

In the second field test, the ROV started the net inspection with an initial estimated net heading of 2° and immediately encountered a small challenge. As the net-following controller started working its way towards the net, the ROV rotated a bit back and forward, and for a little moment, the DVL caught a corner in the opposite direction of the desired direction of travel. This led to the estimated net-heading sharply decreasing before sharply increasing again. The sharp decrease and increase can be seen around time step number 1000 in Figure 5.6 and Figure 5.7. As the change was not larger than 60° from the initial net-heading, no switch happened. The ROV then moved further and encountered another corner right before 200 seconds had passed (time step number 2000), this time in the desired direction. When the ROV was heading towards the corner, the estimated net-heading was decreasing and was about $-10°$ right before the corner. The corner then caused a $\approx 40° - 50°$ jump in the estimated net-heading. Again, the net heading steadily decreased until the ROV encountered a new corner at around 280 seconds. This corner caused another $\approx 40° - 50°$ jump in the estimated net-heading. However, in combination with some noise, it also caused the mission control system to finish the transect and change depth and direction, which can be seen as the step in desired depth in Figure 5.6 and Figure 5.7. The ROV then increased its depth and started moving in the opposite direction. It then quickly encountered the same corner, which caused another jump in the estimated heading. Which again, in combination with noise, caused the mission control system to finish the transect. These events are also displayed in Figure 5.5, as the reverse "S" in the top left. The ROV continued without issues

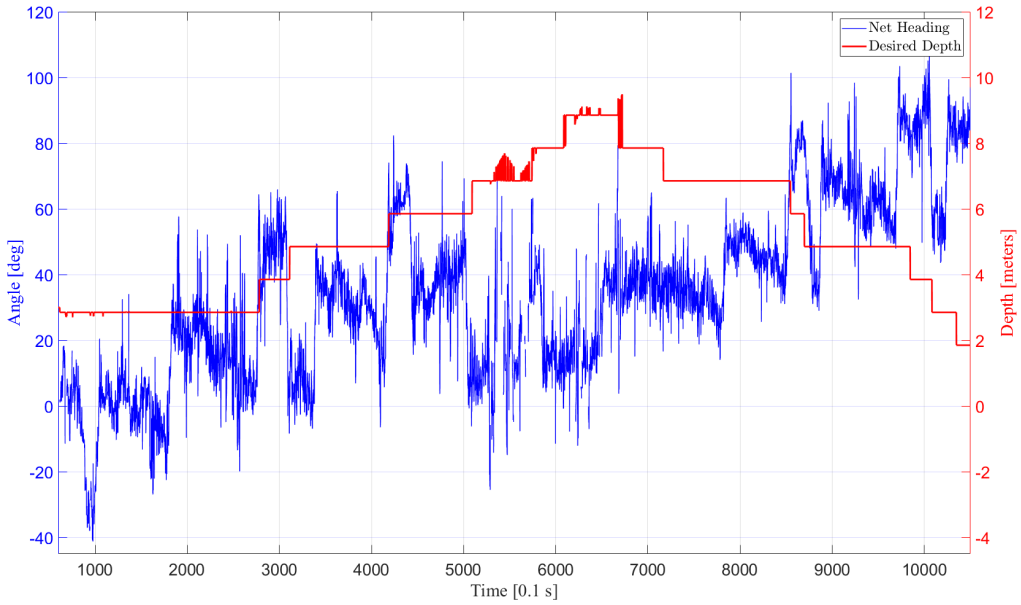

Figure 5.6: 2D plot of the heading and desired depth from the second field test

for a little while until a new corner combined with noise caused a switch after 410 seconds. After changing depth and direction, the ROV reencountered the corner, but this time with reduced noise, hence avoiding the switch. The ROV moved back towards the start without issues until encountering the same corner, which caused the double switch around 280-300 seconds. This corner caused another switch, which can be seen around the 500 second mark in Figure 5.6 and Figure 5.7. After changing depth, at 550 seconds, the ROV encountered a stream of fish which caused the DVL to lose net-lock in short periods. It also considerably increases the amount of noise. The view from the ROVs camera during this event can be seen in Figure 5.8. The ROV drifted slightly away from its original position, but it was able to regain a stable net-lock shortly after. It then started to continue as intended, but due to a combination of noise and the corner, another switch happened at 570 seconds. The ROV then moved away from the corner and continued as intended until the noise caused another switch at 610 seconds. After the switch, it moved back towards the corner. The corner was now less sharp due to the change in depth, and the jump in estimated net-heading was reduced from the earlier $\approx 40° - 50°$ to $\approx 20° - 30°$. Hence, the corner caused no issues for the mission control system. However, shortly after, at around 650 seconds, the noise caused another switch. This switch is not visible in the results due to not changing the desired depth of the net-following controller. This time the ROVs depth exceeded the maximum depth, and the mission control system starts a new transect for moving 60° over before changing depth. However, due to noise or possibly issues with the median filter, another switch happens at 670 seconds. As soon as the ROV changed its depth, another switch happens. This time it was due to the actual net-heading having a 60° difference from the noise point that caused the last switch. When the ROV finally reached the new desired depth at around 710 seconds, it traversed the net successfully for around 170 seconds. During this period, it passed a corner without issues before reaching another corner which rightfully triggered the switch. However,

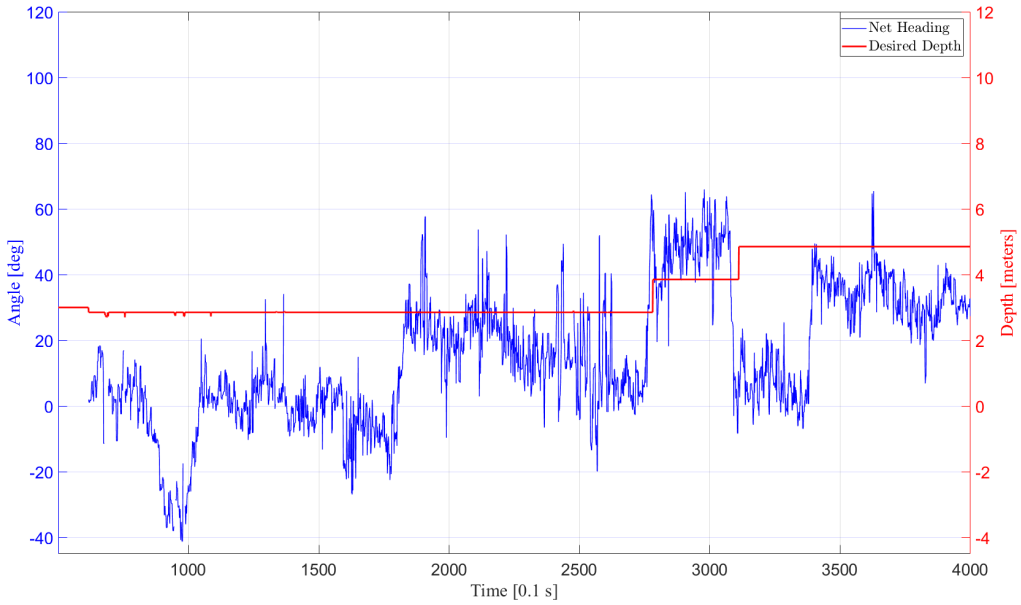

Figure 5.7: A zoomed in section of the plot in Figure 5.6

possibly due to an issue with the moving median filter, another switch was immediately triggered. After reaching the desired depth, the ROV continued traversing the net for around 100 seconds before the test was aborted.

Figure 5.8: A view from the ROVs camera at around 550 seconds into the second field test.

5.2.2 The Fourth Field Test

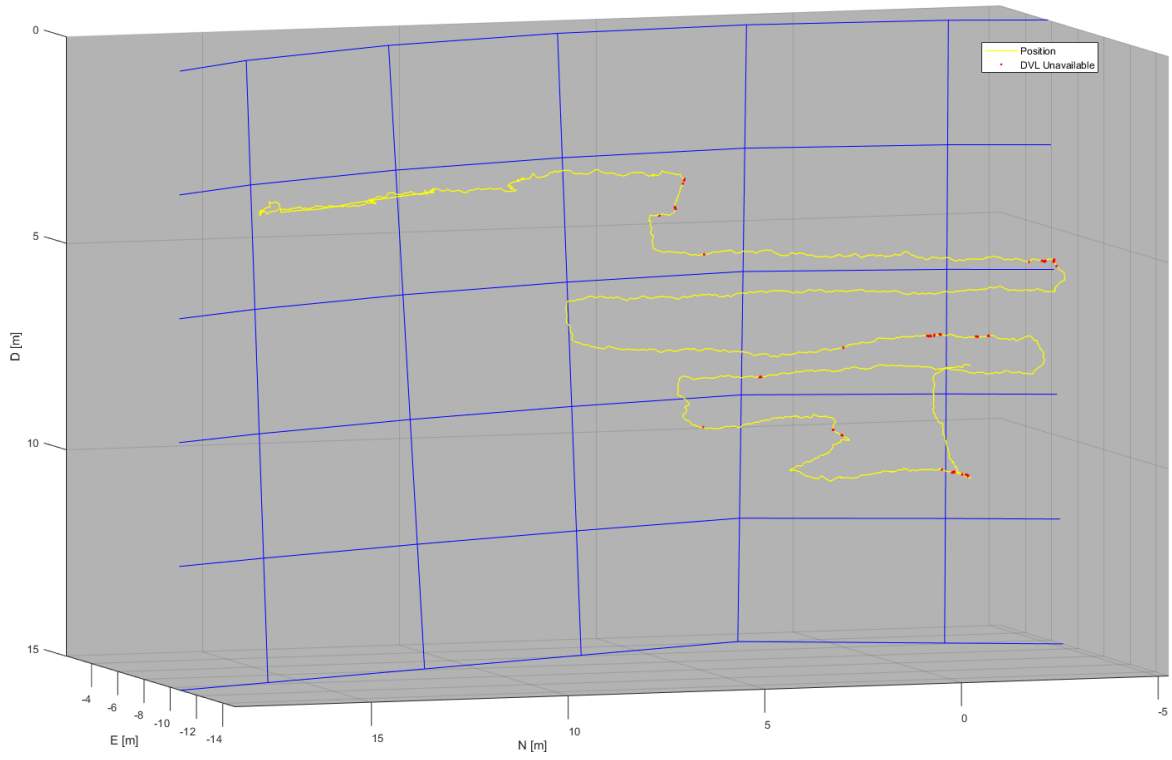

Figure 5.9: 3D plot showing the position of the fourth field test

The fourth field test started with an initially estimated net-heading of 17°. The ROV traversed the net similarly to in the second field test and passed a corner after around 50 seconds. Before this corner, the estimated net heading had dropped by around 20°. The corner caused a jump of $\approx 30°$. The ROV continued traversing the net, and at around 100 seconds, the first switch happens. The switch happens prematurely because of noise. Shortly after, at 115 seconds, another switch happened. This was probably due to an issue with the moving median filter. This double switch can be seen as a reverse "S" in Figure 5.9 and as two steps with few time steps between them in the desired depth in Figure 5.10. After changing depth, the ROV continued in the same direction as before until some losses of net-lock combined with issues with the moving median filter caused the mission control system to switch again at around 175 seconds. From the track in Figure 5.9, the ROV seems to perform three more transects without further issues; however, all the switches happened prematurely due to noise or other issues. While the ROV also encountered corners in the fourth field test, they were not as sharp as the corners in the second field test.

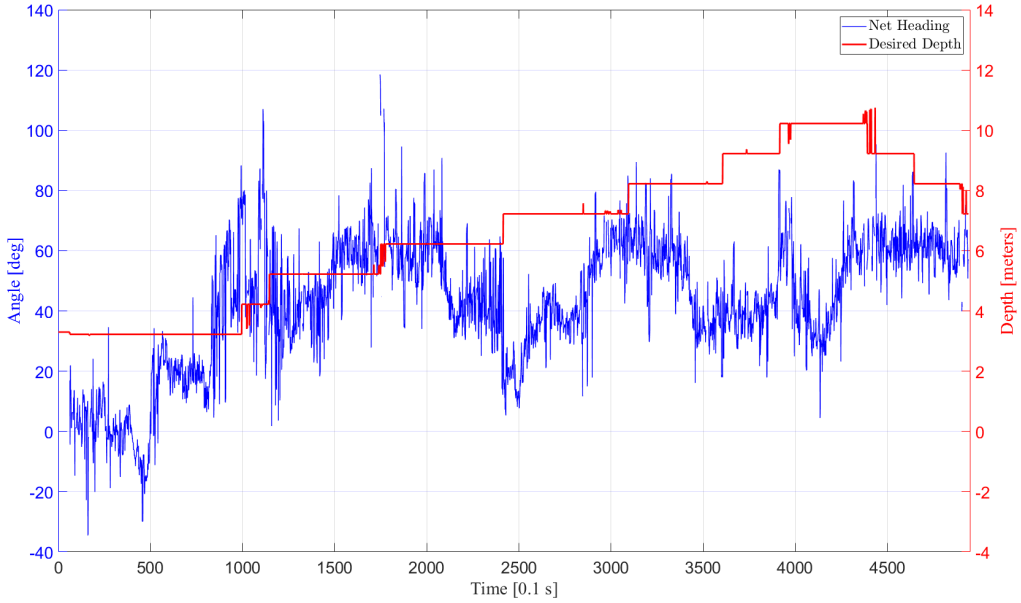

Figure 5.10: 2D plot of the heading and desired depth from the fourth field test

5.2.3 The Fifth Field Test

Between the fourth and the fifth test, an attempt at creating a moving average filter in Aqueous was made. This also meant that some time passed between the fourth and fifth test, which allowed for the wind, waves, and current to increase. When attempting to start the test, the DVL could not achieve steady net-lock. In hopes that the conditions would be better deeper in the net cage, the depth of the ROV was increased. This depth increase can be seen as the near-vertical, mostly red line in Figure 5.11. After reaching a depth of approximately 6 m, the DVL was able to achieve net-lock, and the net inspection was started. As the test was started by attempting net-following, the initial heading was set long before the more successful net inspection started. The test started with an initially estimated net-heading of 9°. After steady net-lock was acquired, the ROV traversed the net as intended, and a little after the estimated net-heading reached $\approx 70°$, the mission control system switched desired direction and depth. This can be seen around time step 2500 (250 seconds) in Figure 5.12. The slightly delayed switch, as well as some outliers not triggering the switch, indicated that the filter could be working; however, as the filter was implemented in Aqueous, its output is not visible in Figure 5.12. The ROV then continued to traverse the net for a little over 100 seconds. During these 100 seconds, no outliers triggered the mission control system to switch, as can be seen between 240 and 300 seconds in Figure 5.12. At around 300 seconds, the DVL measurements significantly worsened, and the net-following controller barely controlled the ROV; instead, the ROV spent most of the time with the autoheading and autodepth controllers activated. This essentially left the ROV to drift until the test was aborted.

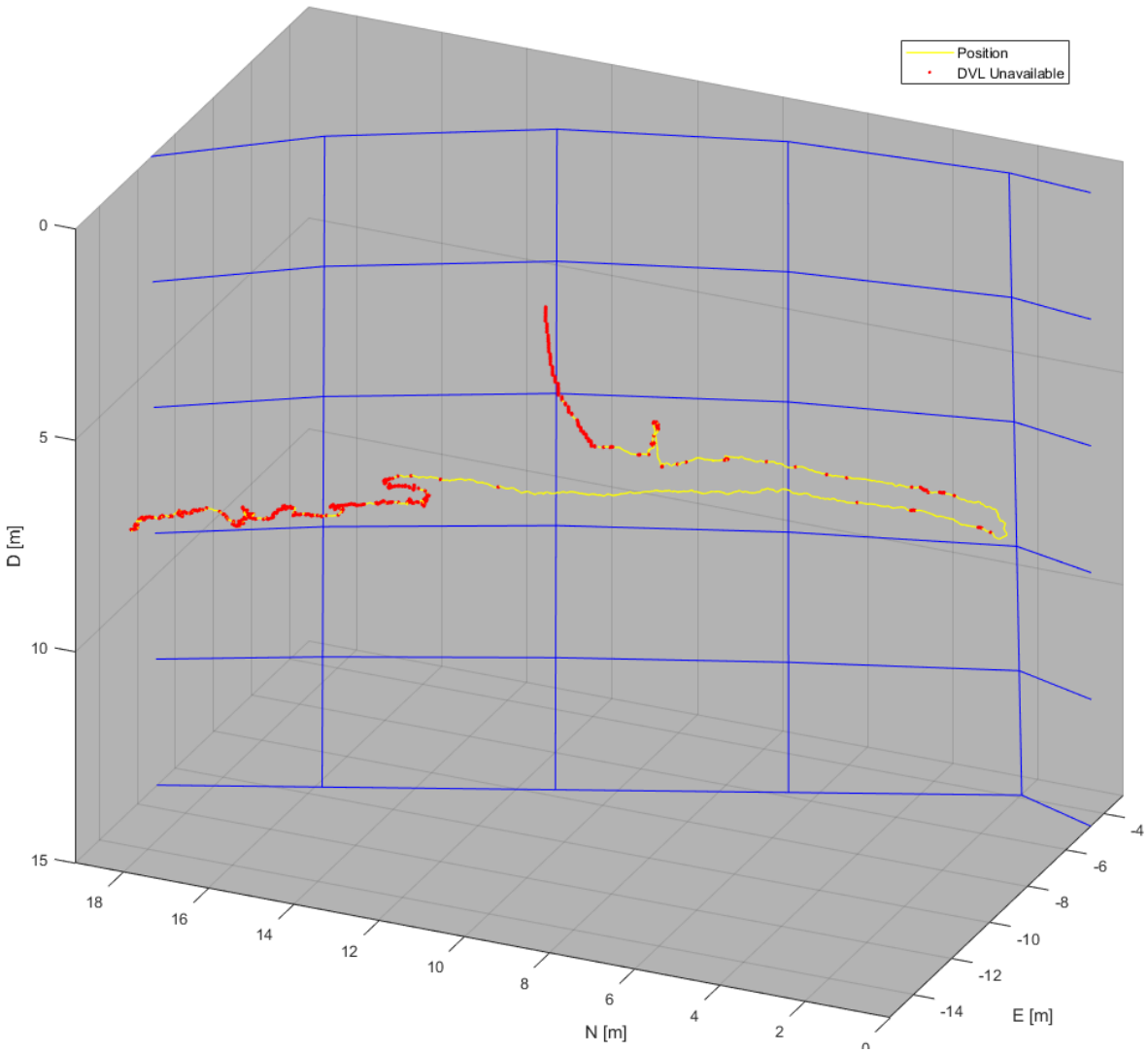

Figure 5.11: 3D plot showing the position of the fifth field test

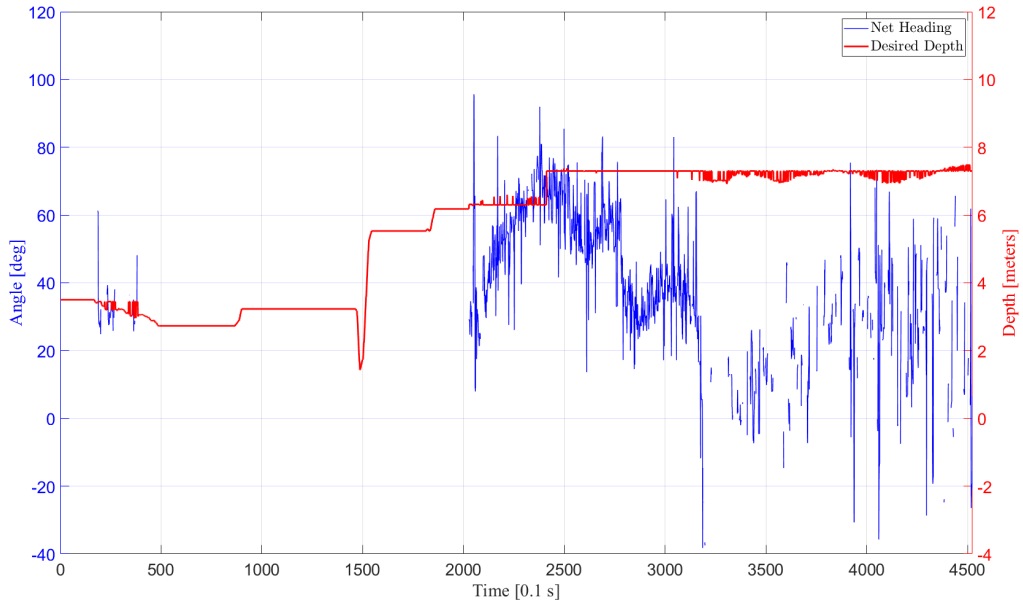

Figure 5.12: 2D plot of the heading and desired depth from the fifth field test

Chapter 6

Discussion

This chapter will discuss the performance and challenges of the final implementation of the autonomous mission control system, which was tested in the field.

The performance of the final implementation in field tests was not significantly improved when compared to the field tests of the second implementation. This largely came down to issues with the filtering and harsher weather. Another thing that affected the results was a local COVID-19 outbreak at Frøya. The outbreak caused the second day of field trials to be canceled, and personnel from SINTEF Ocean and the author were quarantined. This removed a significant number of tests. It also removed the possibility for errors from the first day to be fixed before the tests on the second day.

6.1 Filtering

The main issue in the field tests was insufficient filtering. It was clear that filtering would be needed already from the field test of the second implementation. An attempt at filtering the net-heading with a moving median filter was therefore made in the final implementation. However, due to a bug in the FhSim moving median filter object provided by SINTEF Ocean, the filtering had minimal effect. The bug was discovered by the author of this thesis while investigating possible sources of errors.

The length of the filter in this thesis was selected to be 10 seconds; this was first believed to be 100 time steps. However, this was based on the frequency of the output and not on the frequency of the integration in FhSim. As 100 time steps in the integration equal 1 second, the actual length of the filter should have been 1000 time steps. This was fixed during the field tests, but the input length was discarded due to the bug in the moving median filter object. Instead, the default length of 5 time steps was used. As the filter was intended to have a length of 10 seconds but ended up with a length of 0.05 seconds, the resulting filtering was insignificant.

Instead of improving the net inspection, the moving median filter introduced new issues. Both issues are rooted in the event where the filter would return an average between two values. According to the implementation of the moving median filter, this should only be the case for a filter of even length. However, output values from the median filter, which were never returned by the net approximation object, were found several times in several different

tests.

First, if the estimated net-heading was in the area around $[-180°, 180°]$, the median filter could have two values of approximately $-180°$ and $180°$ as the median. This would return the average value of zero, which would give a completely wrong value to the mission control system. This is suspected to be the cause of several of the switches at the end of both the first and second field tests, as the original heading before being altered for the purpose of presenting the results were around $\pm 180°$. In addition, the average of the two middle values also caused issues when the net approximation object was unable to approximate the net, or a few time steps later, and returned the value of zero. This was managed in Aqueous by omitting values that were exactly equal to zero. However, these were included in the median filter and used for calculating the average of the middle values. If the previous net-heading was around 60° before losing net-lock and getting net-heading values of zero, this could lead to the median filter returning values of 30°, which again could cause the mission control system to switch incorrectly. This was found to be the case for the switch around 440 seconds in Figure 5.10.

Both the bug in the implementation of the moving median filter, as well as the other issues a median filter could introduce, can be hard to notice in simulations and usually require real-world testing to be properly assessed.

The potential of the median filter was proved in section 4.5. To see what effect it could have had on the final field tests a moving median filter with a length of 10 seconds was applied to the net-heading estimate in the second and fourth field tests. The results can be seen in Figure 6.1 and Figure 6.2. It should be noted that if applied in real-time, the moving median filter would introduce a substantial delay.

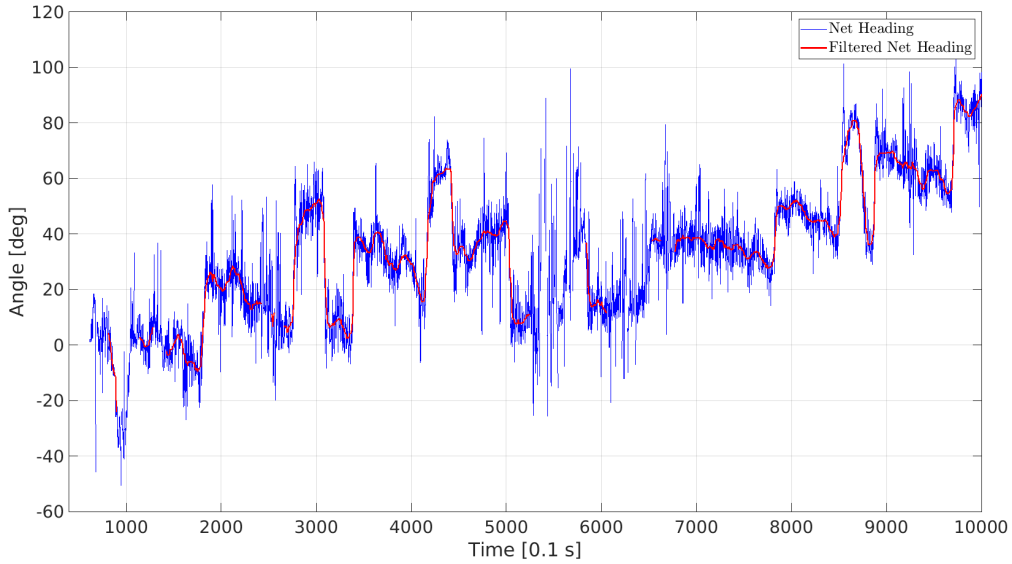

Figure 6.1: The result of subsequently applying a moving median filter to the net-heading estimate from the second field test.

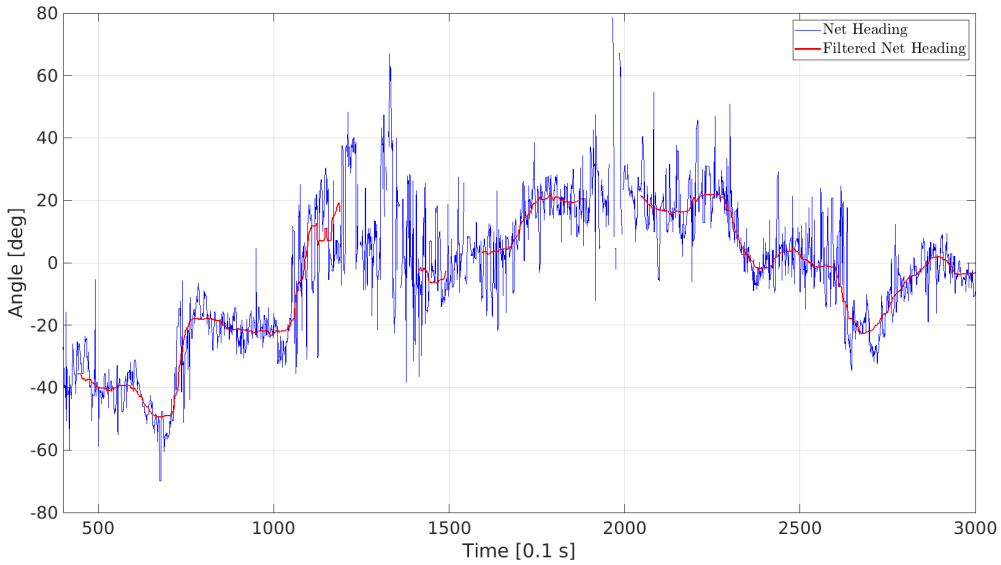

Figure 6.2: The result of subsequently applying a moving median filter to the net-heading estimate from the fourth field test.

While this would have drastically have improved the net inspection, there would still be issues with the net inspection mode. For instance, the sections where the moving median filter cannot filter the net-heading due to losses of net-lock would still occur and would need to be managed. The moving median filter could also require further tuning to avoid being too slow or to avoid large overshoots.

6.2 Corners

An issue that would still occur is the jumps in net-heading caused by the stitching points in the net forming sharp corners. As the net is a soft body, it can flex significantly between the points where it is anchored to the remainder of the net cage structure. This causes the panels between the anchor points to arch, which could create significant changes in the actual net-heading over a small area. These corners appear when the current and waves deform the net significantly, and there are significant forces on the anchoring points. The severeness of the corners could vary between different depths, and sharp changes in the net heading could also occur when traversing the net vertically; however, this is not as common as the horizontal corners. An exaggerated example of such a corner can be seen in Figure 6.3, where the inside of the net cage is the top part of the figure, while the sharp corner would be attached to the remainder of the net cage structure on the outside of the net. If the corner in Figure 6.3 were to be traversed from left to right, the actual net-heading would start at $\pm 180°$, the normal vector of the estimated plane pointing out of the net, straight towards the bottom in the figure. The net-heading would then slowly change towards $-135°$ right before the DVL beams hit the other side of the corner. At this point, the normal vector of the estimated net plane is pointing towards the lower left of the figure. As the DVL beams continue past the corner, the net-heading rapidly changes to approximately $135°$. The normal vector of the estimated net plane now points towards the lower right of the figure. The net-heading

then finally increases back up to $\pm 180°$. Figure 6.4 shows the actual corner, which caused numerous issues in the second field test from the perspective of the ROVs camera.

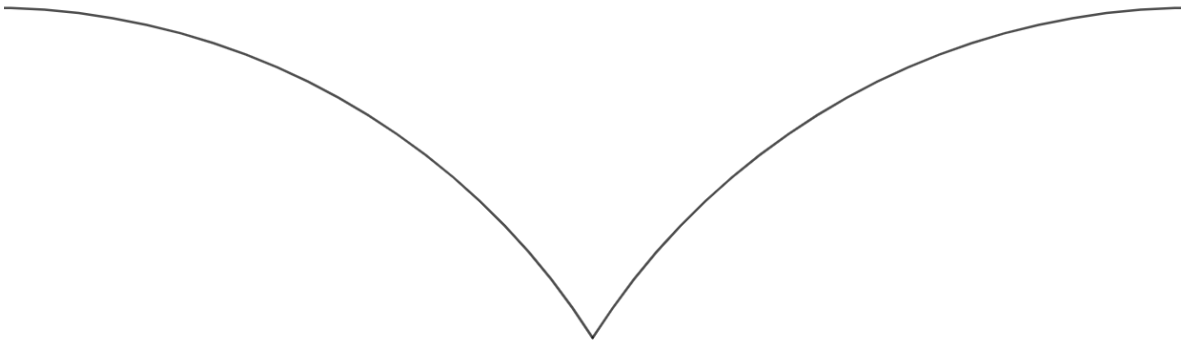

Figure 6.3: An exaggerated example of a possible corner in a net cage seen from above.

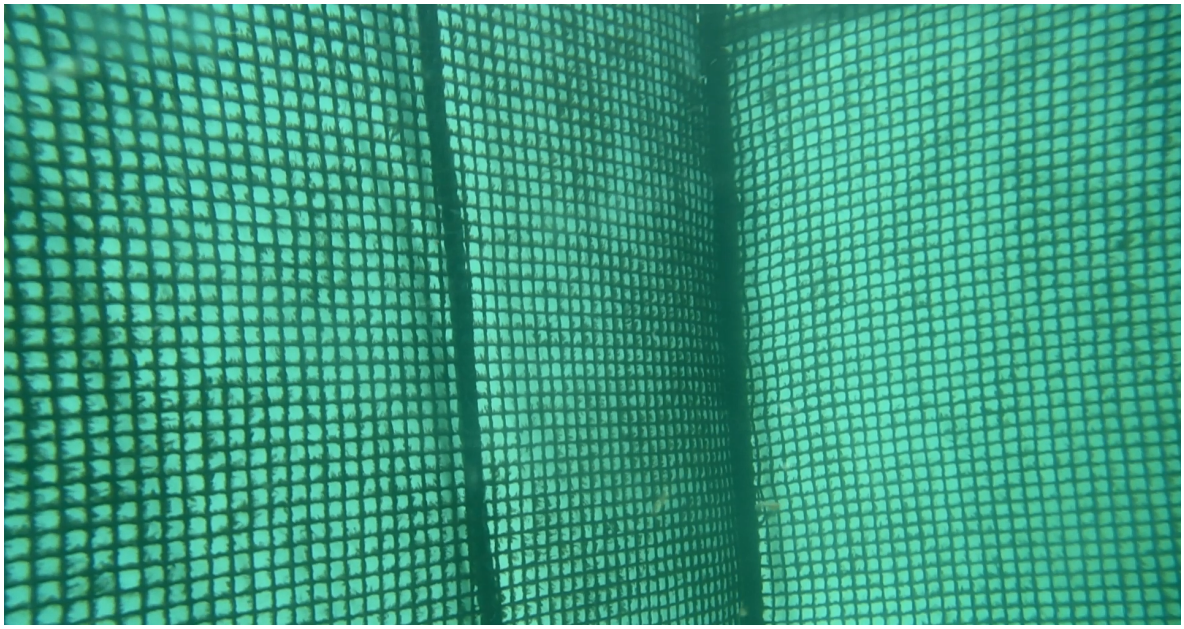

Figure 6.4: An example of a real corner from the second field test.

Using only the net-heading for keeping track of the ROVs' position during the net in-

spection may not be sufficient, even with improved filtering. For overcoming the challenges introduced by corners, the mission control system could include other measurements such as the USBL position estimates, the diameter of the net cage, and the time, velocity, and heading of the ROV. These could also be combined in a Kalman filter or similar to achieve a better situational awareness regarding the localization of the ROV relative to the net cage.

6.3 Signal Processing of the DVL

The DVL is believed to be the source of the noise in the field tests. The gyrocompass of the ROV could be another source; however, gyrocompasses have been used in ROVs in aquaculture for a longer time, and larger deviations would have been noticed. DVLs are typically used for measuring the velocity and distance relative to the seafloor and have only recently been tested for applications in aquaculture. DVLs have been found to be suitable in aquaculture; however, this field is still in its infancy, and as far as the author knows, no DVL has been purpose-made for tracking the net in fish farms. Hence, the DVL used in this thesis could possibly be tuned better for use in aquaculture.

The challenge of using DVLs in aquaculture became apparent during the field trials at Tristeinen, where the performance of the DVL decreased as the day progressed until the point where it became unusable. The cause of the reduced performance is not clear, but it can be assumed that it is related to the harsh weather. As the DVL relies on acoustics to function, it could be disturbed by waves and structures reacting and creating sound. It is also possible that temperature differences in the water could cause different densities and therefore interfere with the acoustics of the DVL. A large amount of biomass moving around in the net cage could also disturb the acoustics of the DVL; however, this has not been an issue in previous field trials utilizing the DVL for net-following.

There could be several ways of improving the performance of a DVL for use in a fish farm. One thing that stands out is SINTEF Oceans' utilization of the DVL. The net approximation object created by personnel at SINTEF Ocean is very conservative in the way it handles the signals from the DVL. The net approximation object relies on a flag from the DVL, indicating whether all the four beams of the DVL provide good measurements. When applying the DVL for estimating a plane of the net, it does not require the same measurements as when measuring the distance and velocity relative to the seafloor. For estimating a plane, only three points are required. Hence, it could be sufficient if the DVL only has three beams providing good measurements. By changing this, the frequency of net-lock losses could possibly be significantly reduced.

6.4 Loss of Net-Lock

Due to the issues with the DVL as mentioned in section 6.3, the behavior for managing the loss of net-lock saw regular use. For the short periods where net-lock was unavailable, the behavior functioned satisfyingly. For longer periods, which were still shorter than the 10-second threshold for setting DP, or periods where net-lock was frequently lost and regained, the ROV could drift more than desired. This could be mitigated by switching to the DP mode instead of only utilizing the autoheading and autodepth controllers. However, using the DP

mode would tear more on the different components of the ROV, and implementing it would be significantly more demanding than utilizing the autoheading and autodepth controllers.

6.5 Tracking the Inspection Progress

The pattern implemented in the final implementation could be more efficient. A large amount of depth and direction changes increases the chance of wrongful switches. All the switches which occurred due to corners in the net are examples of this. By only switching once every 180°, the number of switches would be significantly reduced, and the required difference in net-heading to cause a switch would be 3 times larger. It would still encounter some issues with the corners, but the pattern should be less affected by corners. Another solution could be to mainly utilize vertical net-following, running transects from the top of the net cage to the bottom, with small horizontal change in-between. After inspecting 180°, the ROV would utilize horizontal net-following for return to the start. An attempt at this was made in this thesis; however, it was found that even in simulation, the net-heading would not be sufficient for this to work. To only utilize net-heading for tracking the ROVs position in the net cage would require a completely circular net and a highly accurate net-heading. Instead of doing 60° horizontal moves, the net inspection mainly utilizing vertical net-following would do changes smaller than 10°. Similarly to in section 6.2, the net-heading would not be sufficient for SLAM of the ROV relative to the net cage. A combination of different measurements and previous knowledge could be utilized for improving the tracking of the net inspection progress.

6.6 Practicality of Finite State Machine

The FSM implemented in this thesis functioned as intended in the simulation. Due to the issues mentioned above, the FSM did not see complete testing in the field tests. Only parts of the FSM were tested, and these seemed to function as intended. None of the numerous incorrect switches were due to the FSM. The implementation of the conditions in the FSM should be sufficient, but robustness improvements are required before the conditions will function as intended.

The FSM contributed to easier implementation as well as improved testing. Initially, it reduced the challenge of keeping track of the different states and transitions. However, as the FSM grew, it became increasingly challenging to keep track of the logic flow through as well as understanding the transitions in Table 4.1. A possible alternative to the implemented FSM would be to implement a BT. However, this would require a more extensive restructuring of the net inspection mode.

Choosing a mathematical model of computation for dynamic mission management specification early in the development process could be beneficial for the mission control architecture. Regardless of the mathematical model chosen, it would reduce the clutter when implementing an autonomous mission control system. Creating a mathematical model of computation late in the development process does not significantly improve the mission control architecture.

Chapter 7

Conclusion

The main task of this thesis was to design and implement an autonomous mission control system for connecting the different automatic functions and administering decisions for an ROV such that it is capable of performing autonomous underwater IMR operations. Autonomous ROVs capable of conducting IMR operations could reduce the different risks associated with aquaculture, facilitating the move towards exposed locations.

The design of the autonomous mission control system is fundamentally based on classical control architectures such as the Guidance, Navigation, and Control architecture and then expanded upon for adding autonomous capabilities. Autonomous functions could be introduced in all the different parts of the GNC architecture; however, as mission control systems fall under guidance, it is natural for the autonomous mission control system to do the same. The autonomous mission control system is based on the idea of a hybrid mission control architecture. A hybrid mission control architecture was chosen as it combines the ideas of reactive control and deliberative control. For a system to be autonomous, it must be capable of some deliberation, and for a system to function in a highly dynamic and unpredictable environment such as a net cage in aquaculture, it must be able to quickly react to unplanned events. The autonomous mission control architecture is divided into three different layers. A planning layer for planning the long-range goals of the mission, a behavioral control layer for interacting with sensors and actuators, and an executive layer that functions as an interface between the planning layer and the behavior control layer. For better understanding the flow of logic through the planning and executive layers, FSMs and BTs were studied, and an FSM was implemented in the autonomous mission control system.

The autonomous mission control system in this thesis was created to perform autonomous underwater IMR operations with an ROV. The different IMR tasks the ROV could capable of performing were therefore studied. Most of these tasks, such as mooring inspection, net inspection, fish monitoring, and net cleaning, are currently conducted with manual ROVs, while repairs of the net are usually conducted by divers. Mooring inspection, net inspection, and net cleaning have considerable potential for autonomy. Net repair also has potential for autonomy but requires efficient tools before full autonomy can be considered. In this thesis, the net inspection task was chosen for implementation due to existing automatic functions and research.

As the autonomous mission control system should be able to detect and handle high-risk situations, the different risks connected to the use of autonomous ROVs were studied. The risks with the most prominent consequences are the risk of creating a hole in the net and the risk of the ROV getting stuck.

An autonomous mission control system was then implemented for SINTEF Ocean's Argus Mini ROV. This ROV is controlled by the simulation software FhSim, and operator inputs are provided through the Aqueous GUI. The different behaviors in the autonomous mission control system were implemented in Aqueous. The two behaviors showing the most potential were the behavior for handling loss of net-lock and the behavior for tracking the inspection progress and creating an inspection pattern.

The behavior for handling loss of net-lock was largely successful and significantly improved the autonomy of the net inspection. It could still be slightly improved by utilizing DP for keeping stationary while attempting to regain net-lock.

The behavior for tracking the inspection progress and creating an inspection pattern was successful in simulations but did not function as well in the field tests. The primary reason for this was the noisy measurements in combination with insufficient filtering. However, the shape of the net cage also caused problems, and a better SLAM algorithm could significantly improve the performance of the behavior.

The autonomous mission control system implemented in this thesis was able to perform a fully autonomous net inspection in simulations. It also showed potential for fully autonomous net inspection in real-world fish farms, but further improvements are required for this to be a reality. A net cage is a challenging environment for ROVs as it is positioned in the wave zone with large quantities of fish, creating unpredictable currents. This demands robust control of the ROV as well as robust decisions from the autonomous mission control system. With the aquaculture industry moving further away from sheltered waters, the required robustness is increasing further. The mission control system implemented in this thesis can be of benefit to SINTEF Ocean as it makes ROV operations safer and more robust. However, further work is required before an autonomous ROV can perform IMR operations in exposed aquaculture.

7.1 Further work

The autonomous mission control system implemented in this thesis can be improved by further testing and tuning, as well as including various measurements not utilized in the current implementation. An improved SLAM algorithm could significantly improve the performance of the autonomous mission control system. Significant improvement can also be found by researching the use of DVLs in aquaculture and using a less conservative approach to its utilization. Methods for automatically detecting holes in the net are also of interest when it comes to fully autonomous net inspection. The system could also be expanded to include other IMR tasks in aquaculture, such as mooring inspection and net repair. Autonomous mooring inspection would require further research into methods for detecting and tracking mooring lines. For autonomously net repair, new tools and systems for controlling them must be developed. Different systems could be created for net cleaning, using some of the ideas

presented in this thesis combined with purpose-built ROVs or specialized tools. In addition, the field of resident, possibly tetherless, ROVs could be further researched in order to allow for fully autonomous IMR capable ROVs in the future.

Appendix A

Risk Analysis for Autonomous ROVs in Aquaculture

Henning Ødeby Karlsen * **Hwang Jae Hyeong** *
Vilde Økland Drønen *

* *The Norwegian University of Science and Technology, 7491 Trondheim, Norway.*

Abstract: The objective of this paper is to describe and discuss the risk factors and safety challenges regarding autonomy in aquaculture. The Norwegian sea-based aquaculture is a dangerous occupation for the human operators and expanding fish farms offshore will increase the need for autonomous monitoring, operations and decision support systems. New technological solutions are necessary for ensuring safety and efficiency. Not only for the human operators but also for the fish and the environment. Autonomous ROVs could help reduce these kind of risks however there are still risks tied to autonomous ROV operations. These risks are discussed and used to build a decision base for the autonomous ROV. By considering risk factors and suitable actions for the autonomous ROV, it is possible to prevent unwanted situations in advance. Fuzzy logic was implemented to analyze risk factors in fish farming with an autonomous ROV. Simulations were done with a fuzzy logic model implemented in SIMULINK.

Keywords: Aquaculture, Autonomy, Risk Analysis, Unmanned Marine Vehicles, Remotely Operated Vehicles, Fuzzy Logic.

1. INTRODUCTION

1.1 Background

Risk analysis has wide applicability to aquaculture. It can be applied in assessing risks to the environment posed by hazards created by aquaculture development, which includes risk of environmental degradation, pests, genetic impacts and so on. Another aspect could be the risk to the fish, such as the risk of fish escape, which would have an impact on both the environment and the economy of the fish farm. On the other hand risk analysis can be applied to the people working at the aquaculture facilitates. The sea-based fish farming is one of the most dangerous occupations in Norway, and sometimes operate at the edge of safety limits (Utne et al., 2015). High manual workload, utilization of heavy equipment and harsh weather conditions, contribute to the risk.

With an increased growth of the fish farming industry, the need for expanding to new locations is inevitable. This includes moving fish farms into more exposed areas with more severe weather conditions and demanding sea states (Bjelland et al., 2015). Operations and technologies in fish farms are today highly dependent on human interactions with tools and fish cage structures. The environmental challenges at sea create limited operational weather windows, which makes it difficult to carry out operations without exposing the operating personnel for a significant risk of harm. It is therefore desirable to develop solutions where less operators must be present to carry out tasks manually. New and adapted methods that improve tools, technology and platforms in aquaculture with more autonomy are needed.

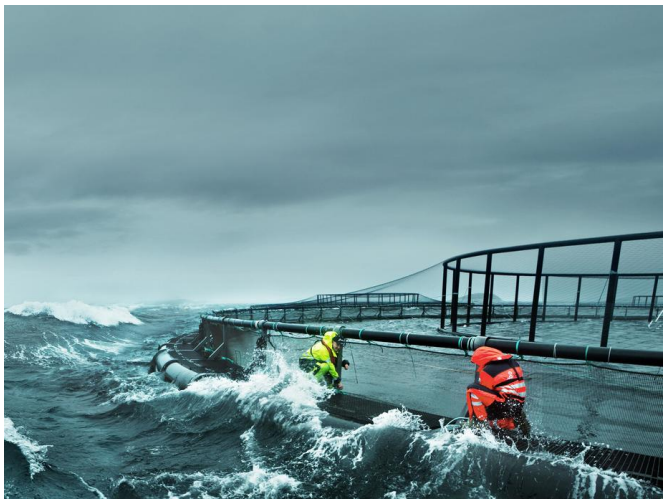

Fig. 1. Fish cage with operators in exposed area.

More exposed locations for aquaculture makes the use of remote control and underwater vehicles (UV) feasible for underwater operations (**?**). More frequent use of autonomous and remote control options such as vessel-control, Remotely Operated Vehicles (ROV) and cage-integrated intervention tools will help enlarging the weather window without putting any human operators at risk. One can also argument for reduced risk of fish escapes and net-faults by implementing autonomy. For instance, having ROVs doing net inspections and continuously reporting holes or other failures of the fish cage will help prevent fish escapes.

1.2 Objective and scope of paper

The objective of this paper is to present the different challenges connected to risk in autonomous aquaculture and discuss autonomy as a tool of preventing risk and cause of risk. Several risks are present in relation to ROV operations in aquaculture today, and by implementing autonomous ROV operations the challenges regarding safety and risk will change, possibly reducing some while introducing others. These challenges will be presented and discussed. An approach of using fuzzy logic to develop a safety decision basis for the ROV will also be presented.

2. RISK RELATED TO AUTONOMY IN AQUACULTURE

Risk can be divided into consequence categories in several different ways. We have decided on 4 main categories; Safety, Environment, Asset and Biological Welfare and Safety. In an aquaculture setting the safety category refers to the safety of the humans involved, i.e. the risk of injuries and fatalities. The environment category refers to the possible consequences for the ecosystem, an example being the risk of escaped fish spreading diseases to wild salmon. The asset category is related to the risk of damaging or losing the asset which is what brings value. For an ROV in aquaculture this could be divided in two. The most important asset in aquaculture is usually the fish, as it is the primary value of a fish farm. The other asset is the ROV itself, which is costly to replace or repair. We also include the safety and well-being of the fish, as it is a living being that can feel pain and should not be exposed to unnecessary trauma (Ashley, 2007).

2.1 Safety

Sea based aquaculture is one of the most dangerous occupations in Norway (Holen et al., 2016). Most of these accidents are not directly related to ROV operations, but ROV operations could be a contributing factor. Most ROV operations requires a service vessel next to the submerged net-cage which contains the fish. As the fish farms and net-cages can be found in a variety of different areas, they can be exposed to a variety of different sea states. According to (Holen et al., 2016) harsh weather is a risk influencing factor both for risk concerning humans, and risk concerning the structures. When using an ROV in connection to a fish farm, the usual sequence of events is as follows: A service vessel maneuvers to the net-cage, and launches the ROV. The service vessel then stays more or less in the same position while the ROV conducts an inspection, maintenance or repair (IMR) operation. After the operation is finished the service vessel will pick up the ROV and return to its base. During this stage the involved humans will be stationed on board the service vessel. The consequences for the humans in this operation are mostly connected to the service vessel. Either due to events on the service vessel itself, such as loss of power, unexpected movement or loss of vessel, or due to events such as moving heavy objects and using cranes when launching/retrieving the ROV (Solem, 2017). This type of operation will likely be the same in the coming years. Even though ROVs could become autonomous, they would not remove the need for the service vessel until they are fully resident.

2.2 Environment

The major risk for the environment is the risk of fish escapes, as they could have detrimental genetic and ecological effects on wild fish or other fauna in the coastal environment (Østen Jensen, 2010). ROVs are important for inspecting the net and structures containing the fish, and thereby reducing the risk of fish escape. However the use of ROV could also contribute to fish escapes, i.e. if it gets stuck in the net, or runs into the net with something sharp protruding from the ROV. The general understanding is that using ROVs are beneficial when it comes to inspecting and maintaining the fish farm, as the risk of the ROV breaching the net is small when compared to the risk it reduces by inspecting the net or other structures. Another possible consequence for the environment that can be connected to the use of ROVs is pollution of micro plastics. This is something that happens when the net is being cleaned with high pressure water, and microscopic parts of the nylon net is torn off.

2.3 Asset

The most important asset is the fish and the largest risk regarding the fish in aquaculture is again the risk of escape. If a large portion of the fish in a fish farm escapes, a large amount of the value is lost with it. However, as the fish is a living asset there are several other risks tied to loss of the fish in aquaculture. Stressing or injuring the fish could devalue it or in the worst case kill it which again will cause it to lose value. This will be further discussed later, but in relation to the welfare and safety of the fish and not in relation to economic value.

The other asset is the ROV, and the common risks would be damaging or losing the ROV. Losing an ROV can happen if the tether is cut and the ROV is in an open area, and not inside a net-cage. This is not something that commonly happens, but it has happened (Olsen, 2020). Damaging the ROV is more common, and could happen if it runs into some obstacle, gets stuck in the net or tangles the tether in some mooring lines.

The fish in Norwegian fish farms is protected by the Animal Welfare Act (The Norwegian Ministry of Agriculture and Food, 2009). As a result, the welfare and safety of the fish must also be considered when discussing consequences. In relation to the usage of ROVs the largest consequence is fatality, which could occur if the fish get stuck in the structure of the ROV or if the fish comes into contact with the propeller blades in one of the thrusters. Fatalities could also occur as a result of sicknesses induced by stress, injuries or lice infection(Ashley, 2007). The use of ROVs could influence the stress in fish, as well as cause injuries. How ROVs influence fish welfare is not widely researched, but the use of ROVs probably have some influence on fish welfare. This could be both positive and negative. An example with positive outcome is the use of ROVs for improving net pen conditions. However, according to SINTEF personnel working closely with ROV operations in aquaculture and (Kruusmaa et al., 2020), salmon usually stays away from the ROV. This is indicating that the ROV scares and stresses the salmon, which is a negative consequence. Cleaner fish seem to be indifferent to the

ROV, coming right up to the ROV. While this could be an indication that the cleaner fish are less scared or stressed by the ROV, it means that the risk of the ROV directly damaging the cleaner fish is higher than what it is for the salmon.

3. KNOWLEDGE BASE

We have been able to identify some risks related to the use of ROV in aquaculture, the largest being the risk of fish escape. The most frequent reason of escape of fish is structural reasons, including mooring failure, breakdown of net-cage structures, abrasion and tearing of nets (Østen Jensen, 2010). For ROV operations the most relevant of these are abrasion and tearing of nets. While damages to these structures are among main reason for fish escapes, ROV operations are not among the most critical operations in terms of a potential escape (Thorvaldsen et al., 2015). However, from the perspective of an ROV operation the risk of fish escape is the largest risk directly related to the ROV.

We will now look into how we can use the identified risks in a decision making process for an ROV to avoid fish escapes. The focus will be on scenarios where the ROV could come close to the net and cause damages to the structures containing the fish. Initially we will look into the operational modes of the ROV, and identify the risk scenarios in relation to each operational mode. The operational modes and the related risks have been specified based on (Solem, 2017) and an interview with relevant people at SINTEF, working closely with ROVs in aquaculture.

3.1 Operational modes

An autonomous ROV would need several operational modes for autonomous inspection, maintenance and repair in aquaculture. Some of these could be quite simple and obvious, such as launch, transit, station keeping (DP) and recovery, while others are more advanced and require extra sensors or equipment. The simple ones will probably be common in most autonomous underwater vehicles (AUV), and they have already been researched and implemented (Rist-Christensen, 2016). A few more advanced operational modes are already researched and some even implemented for some ROVs, examples are net following, net repair and net cleaning. Other potential modes that could be researched and developed in the future are; mooring line following, dead fish collecting, net-relative navigation, etc.

3.2 Launch

During the launch phase of an ROV operation, the ROV is put into the water and all systems are tested to see if they are stable and ready for operations. Today, this is usually performed manually. In this phase, the ROV is in close proximity to the work boat and if any system is inoperative to the extent that the ROV does not function normally the operation should be aborted and the ROV should be recovered. For the autonomous system, there is not many decisions to make apart from possibly discontinuing the mission. The only event tied to risk in the control mode for launch will therefore be a "Systems Faulty" event. This event will also be relevant for all the other modes. If the ROV does not function as expected, it could damage structures such as the net. If it later were to get stuck/inoperable somewhere away from the work boat this would require some human controlled operation to recover the ROV, which leads to increased risk of both damaging structures and injuring humans.

3.3 Transit

In the transit mode, the ROV is moving from one point to another, either along a predefined path or by moving in a direction while avoiding obstacles if the ROV is capable of detecting and avoiding obstacles. Autonomous transit has been implemented in research projects, (Rist-Christensen, 2016), but currently transit is most commonly done by manual control. In autonomous transit, there are several events that could lead to unwanted consequences. One example is the possibility of being off a predefined track, such that the path no longer is cleared of obstacles. Another example is when the ROV has detected an obstacle and need to avoid it. A final example is if the ROV has taken a bad path and managed to get its tether stuck, preventing the ROV from moving as expected. This scenario will be relevant for several of the modes, but most relevant for transit as transit is the mode where the ROV travels the greatest distance. Both the ROV being unable to move as expected, and the tether being stuck in some structure leads to an increased risk of damaging structures. It could also require some human controlled operation to free it from being stuck.

3.4 Station Keeping

In the station keeping mode, the ROV is staying stationary in one position. It will be able to do small and precise changes in its position/heading, but for a larger distance it will use the tracking mode. Station keeping or dynamic positioning (DP) as it is often called, is common for surface ships such as supply vessels in the oil and gas industry. DP is well studied and an example of this is (Chen and Moan, 2004), which looks into the risks of collision between a shuttle tanker and FPSO while offloading. They mainly look into the possibility of a "drive off" which is when the shuttle tanker drives away from the desired position due to wrong control input or some other error causing wrong thruster forces. They also mention "drift off", which is when no thrust or not enough thrust is produced and the vessel drifts away from the desired position. These two events can be adapted for an ROV in station keeping as well.

3.5 Recovery

The recovery mode is when the ROV is back at the surface and close to the work boat. Similarly to the launch mode, this is currently usually done manually. In this scenario there are no big risks of any structural damage or damage to the fish. As the recovery itself is an operation involving humans there is some risk of injuries and fatalities, but this would not be something the ROV needs to take into account.

3.6 Net Following

During the last decade ROVs have gradually replaced divers when it comes to inspection of the net pen. A method for autonomous net following has therefore been researched and implemented (Amundsen et al., 2020). In this mode the ROV is using DVL (Doppler-Velocity-Log) to estimate the relative position of the net, and moving along it at a constant distance. There are several scenarios in the net following mode that could lead to structural consequences, as the ROV operates in close proximity to the net. One scenario is the scenario where the ROV is closer to the net than some predefined threshold, and even small disturbances can cause the ROV to hit the net. Another scenario is the scenario where the DVL loses track of the net, leaving the ROV unable to follow the net or know where the net is. There could be several reasons for this, but a common one is due to fish swimming between the ROV and the net. The fish usually keep on swimming and most of the time the DVL is able to regain track of the net within a short amount of time. Finally there is the scenario where the ROV is following the net, but the distance estimate or angle is wrong, leading the ROV to drive into the net. In these three scenarios, all it takes is a small disturbance and it could lead to the ROV hitting the net, possibly getting stuck and thereby causing a tear in the net.

3.7 Net Cleaning

The accumulation of biofouling on cage nets is a major problem in aquaculture (Bloecher et al., 2013). ROVs with purpose built washing tools, or purpose built ROVs are used for cleaning the nets (Solem, 2017). When cleaning the net these ROVs have to come close to, and sometimes even in contact with, the net. This could lead to situations where the ROV makes a tear in the net, or enlarges an existing tear. The cleaning rigs are usually made in such a way that they minimize the wear and tear on the net, but there is still a risk of tearing the net. Scenarios where this could happen are similar to the drive off and drift off scenarios in the station keeping mode, as the control and movement of the ROV will be similar to station keeping until it touches the net. There are also some scenarios related to the contact with the net, such as applying too much force, or getting the cleaning rig entangled or stuck in the net.

3.8 Net Repair

Holes in the net containing the fish is one of the most common causes of fish escape (Østen Jensen, 2010). While usually fixed by divers, there have been research and development of tools that enable ROVs to fix holes (SINTEF, 2012). It is therefore possible to assume such tools and operations will be researched and implemented for autonomous ROVs as well. When repairing the net, the ROV has to come in contact with the net. This is something that can have great consequences if something were to go wrong, especially if there already is a hole that could expand. Some solutions also require the ROV to apply force to the net for the repair tool to work, something that increases the risk of a further tearing of the net. Due to the aforementioned reasons, there are several scenarios in this operational mode that could lead to unwanted structural consequences. Similarly to net cleaning, a couple of these scenarios can be the ones mentioned in the station keeping mode. Another scenario, highly relevant in this mode, is if the ROV applies to much force to the net. Lastly the ROV could get itself or the repairing tool stuck in the net.

3.9 Other operational modes

The other modes mentioned in the introduction of operational modes are not as common, less researched, or less used in current aquaculture when compared to the different modes discussed above. As a result, we have not looked any further into these modes and the risk connected to them.

3.10 Knowledge Base for Fuzzy Logic

In this paper, the focus will be on a few of the specific scenarios listed in the net following mode as it is a mode where the ROV will spend a substantial amount of time in close proximity to the net. One of the scenarios we want to include in a decision making system for an ROV is the scenario where the ROV is closer than some predefined distance. If the ROV was able to keep at a safe distance, there would never be any risk of tearing the net. The other scenario we will include in our fuzzy logic simulation is the scenario where the DVLs estimate of the net position is lost. If this happens the ROV no longer knows its position relative to the net, and a collision with the net could easily happen.

3.11 Rules

Based on the scenarios above, we structured some rules for use in the decision making system, which in this paper will be a fuzzy logic simulation. The rules can be found in Table 1. The different categories in the table will be defined in section 4.

Table 1. Knowledge Rules.

DVL Reference	Distance to Net	Action
Red	Green	AbortMission
Red	Yellow	AbortMission
Red	Red	AbortMission
Yellow	Green	Continue
Yellow	Yellow	GoToLastSafeWithReference
Yellow	Red	GoToLastSafeWithReference
Green	Green	Continue
Green	Yellow	Continue
Green	Red	GoToLastSafeWithReference

4. FUZZY LOGIC

Fuzzy logic was first developed by Lotfi Zadeh in 1960s who was a professor in the University of California at Berkeley. He introduced fuzzy set in order to quantify vagueness of natural language. Fuzzy logic describes uncertain or vague situation as various kind of status, not only expressing it as true or false. It considers 0 and 1 as the extreme cases of truth and false but also thinks various

status in between 0 and 1. For example in risk assessment 0 may describe safe and 1 would describe dangerous. Depending on the resulting values of fuzzy logic, we can figure out just how dangerous the situation is.

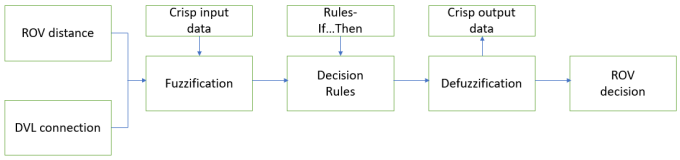

Fig. 2. Fuzzy logic interference system

4.1 Fuzzification

A crisp set of input data are gathered and converted to a fuzzy set using fuzzy linguistic variables, fuzzy linguistic terms and membership functions. This step is known as fuzzification. It is the process of generating membership values for a fuzzy variable using membership functions.

ROV Distance

We will divide area in three zones where ROV operates, green, yellow and red. Red zone is the most risky area which is from 0m to 2.5m. Yellow zone is from 2.5m to 5m. Green zone is from 5m to 10m. "ROV distance" in the rest of the paper refers to the current distance to the net, or the last distance sent from the DVL before it lost connection.

Time Since Last DVL Reference

Another factor that may influence the risk of hitting the net in net following is the sensor data. In our system, a DVL is used for measuring distance to cage. Three regions are set for "Time Since Last DVL Reference", hereby referred to as "DVL lost connection time". These three regions are green, red, yellow. Green is the case when DVL lost signal from 0 to 2.5 seconds ago, yellow is for the case where the signal was lost from 2.5 to 7.5 seconds and red is the case where the signal was lost 5 seconds ago or more.

ROV Decision

Possible decisions were simply categorized as three. Abort, GoToLastSafeWithReference (It will be described as GoToRef) and Continue. When the situation is highly risky the ROV should abort the mission and return to the base. Those are cases when the ROV is too close to the net and the DVL has lost connection for long time. Based on the DVL connection status the ROV can decide to go to a safe reference point (GoToRef). When both conditions are fine, the ROV is allowed to continue the mission.

4.2 Fuzzy Rule Set

In this paper, 6 input fuzzy sets and 3 output fuzzy sets are defined. Every fuzzy set is expressed with membership functions. Figure 3, figure 4 and figure 5 represents the membership functions for input and output variables. How the graphs in the membership functions are configured is listed in Table 2

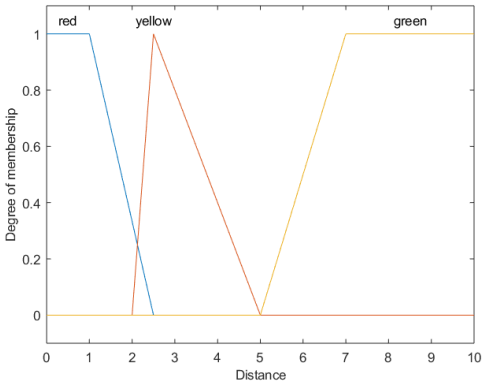

Fig. 3. Input Variable ROV Last Distance.

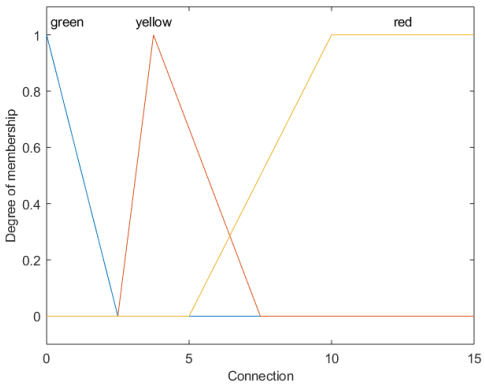

Fig. 4. Input Variable DVL Lost Connection.

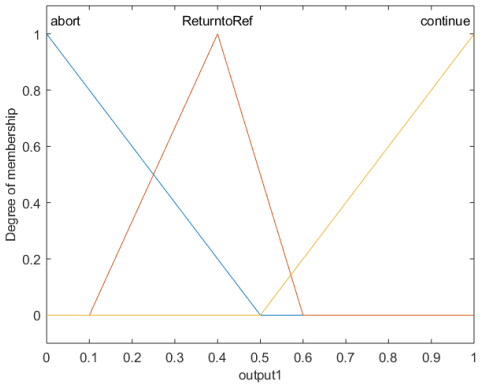

Fig. 5. Output Variable ROV Decision.

4.3 Defuzzification

The defuzzification step is for calculating crisp output from fuzzy output using membership functions. After fuzzification, output data shall be numerically generated, which will tell the ROV which decision to choose. GoToLastSafeWithReference is described as GoToRef in Table 2 in order to make it fit into the table length.

Table 2. Fuzzy sets and membership functions.

Input and Output variables	Fuzzy Set	Fuzzy Set parameters
ROV Distance(m)	red	-2.5 0 1 2.5
	yellow	2 2.5 5
	green	5 7 10 12
DVL Lost Connection(t)	red	5 10 15 16
	yellow	2.5 3.75 7.5
	green	-2.5 0 2.5
Decision	abort	-0.5 0 0.5
	GoToRef	0.1 0.4 0.6
	continue	0.5 1 1.5

Table 3. Simulation result from simulink.

ROV Distance	DVL lost Connection	ROV Decision
0.47	0.70	0.36
6.78	10.18	0.16
6.79	10.18	0.16
9.34	14.02	0.16
3.83	5.75	0.33
5.19	7.79	0.23
8.30	12.46	0.16
0.34	0.51	0.36
0.53	0.80	0.36
5.29	7.94	0.23

5. SIMULATION

5.1 Setup

MATLAB SIMULINK was used to setup our simulation of the fuzzy logic. The command `fuzzy` enables the user to set the graphs and rules of fuzzy logic and by implementing it in SIMULINK, we can run different cases with random inputs and get different outputs. ROV Distance and DVL lost Connection time can be given by using the Uniform Random Number block in SIMULINK, by choosing different maximum values. These random numbers can be checked with a display block for each cases. After running these values through the Fuzzy Logic Controller with Rule-Viewer, defuzzified output value can also be checked with a display block. By using this value the ROV can decide which action to take. By running the SIMULINK model we can get output data and analyze it by surface output and rule output. In order to record every simulated cases, a simout block was used for sending the data to them MATLAB workspace. 10 cases were simulated in total.

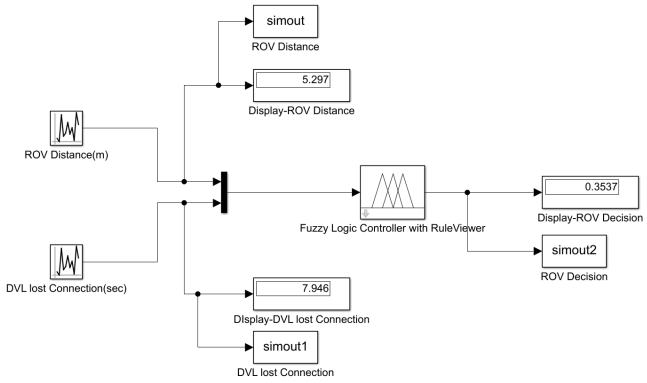

Fig. 6. Simulink model.

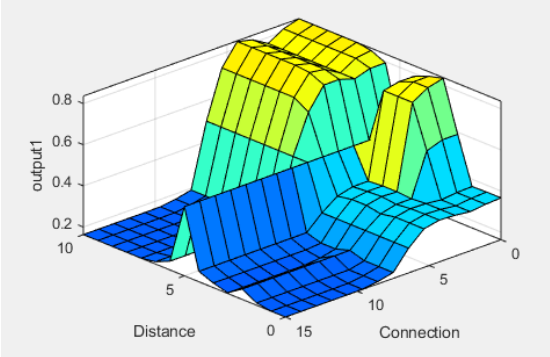

Fig. 7. Surface Output

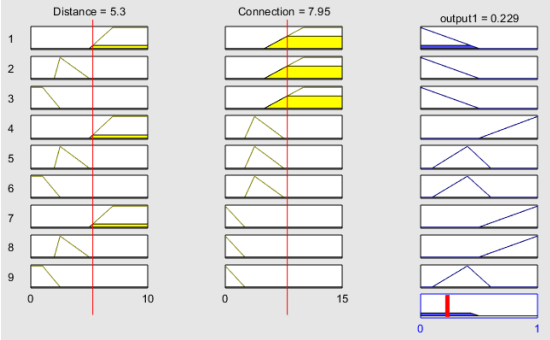

Fig. 8. Fuzzy Interference diagram.

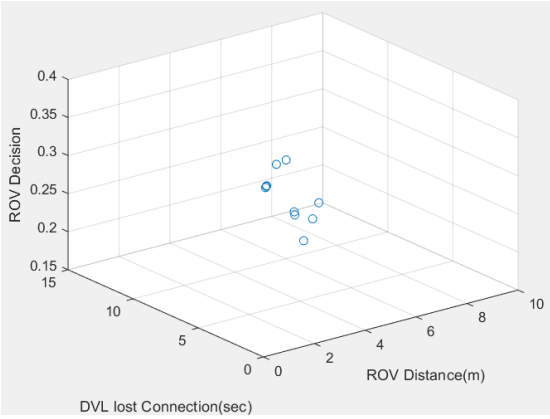

Fig. 9. Result Plot.

5.2 Results

Even though we used random numbers in SIMULINK, the output values are located between 0 and 0.4. We can see when the ROV Distance is green(5.297m) and DVL lost Connection time is red(7.946sec) output value is 0.23 and the ROV has to take the action "Abort". However, we can also see that the defuzzified output tells of an uncertain decision at the separation between the yellow and green zones of ROV distance, at 5m. We can also check this at figure 7. The output of most simulated cases are below 0.5. It seems like every case has to abort mission as we set output values from 0 to 0.5 should abort mission. However, it turns out only four cases should abort mission, [6.78,10.18,0.16], [6.79, 10.18, 0.16], [9.34, 14.02, 0.16], [8.30, 12.46, 0.16], and rest of cases should return to reference point. This shows characteristics of

fuzzy logic that the decision is made by a combination of different fuzzy input variables. By running a larger amount of simulations and tuning the membership functions in the fuzzy logic controller, we could probably have achieved results closer to our expected rules from Table 1.

6. CONCLUSION AND FURTHER WORK

This paper addresses the challenges related to exposed aquaculture and autonomy in underwater operations. Introducing autonomy in ROV operations can reduce operator risk, cost and be time saving. An operator will still be needed to supervise the operations, and in the near future this operator will be on board a service vessel next to the net cage. But in the future there are possibilites for resident ROVs, enabeling the supervising operator to be located on shore in a safer environment. A major challenge is to develop autonomous ROV systems that can handle and respond to the unstructured environment of a fish farm. A major part of this is the decision making system of the ROV, which needs a model for analyzing risk factors. Uncertainty regarding risk influencing factors (RIF) will affect the performance of the ROV, which must be taken into account in the ROV-based operations. In this paper a Fuzzy logic basis for an ROVs decision making system is presented.

In order to mimic the nature of random faults and failures in the ROV subsystem, this study uses a simulation with random inputs. Hence, the results may not reflect true behaviour of the sensors and the ROVs decisions. By using live ROV sensor data the validity of the method would be increased. However, the fuzzy interference system provided a systematic basis for the decision support basis of the ROV. In this system fuzzy rules have been designed according to what was seen as realistic events of possible hazards caused by the ROV. The result obtained could be valuable for development of a decision making system in ROVs, based on consequence analysis and operational safety.

Further research could be to expand this decision making system to include multiple of the modes and scenarios discussed in the Knowledge Base section. There is also potential for further research of the risks and consequences of using ROVs in aquaculture.

ACKNOWLEDGEMENTS

We would like to express special thanks of gratitude to Walter Caharija and Herman Biørn Amundsen who are working in SINTEF Ocean and also guiding us in our master's thesis. They gave us valuable information and resources about real challenges and solutions regarding autonomous ROV operations in aquaculture.

REFERENCES

Amundsen, H.B., Caharija, W., and Pettersen, K.Y. (2020). Autonomous ROV inspections of aquaculture net pens using DVL. *IEEE (submitted)*.

Ashley, P.J. (2007). Fish welfare: Current issues in aquaculture. *Applied Animal Behaviour Science*, 104(3), 199 – 235. doi:https://doi.org/10.1016/j.applanim.2006.09.001. URL http://www.sciencedirect.com/science/article/pii/S0168159106002954. Fish Behaviour and Welfare.

Bjelland, H., Føre, M., Lader, P., Kristiansen, D., Holmen, I., Fredheim, A., Grøtli, E., Fathi, D., Oppedal, F., Utne, I., and Schjølberg, I. (2015). Exposed aquaculture in norway. 1–10. doi:10.23919/OCEANS.2015.7404486.

Bloecher, N., Olsen, Y., and Guenther, J. (2013). Variability of biofouling communities on fish cage nets: A 1-year field study at a norwegian salmon farm. *Aquaculture*, 416-417, 302 – 309. doi:https://doi.org/10.1016/j.aquaculture.2013.09.025. URL http://www.sciencedirect.com/science/article/pii/S0044848613004833.

Chen, H. and Moan, T. (2004). Probabilistic modeling and evaluation of collision between shuttle tanker and fpso in tandem offloading. *Reliability Engineering & System Safety*, 84(2), 169 – 186. doi:https://doi.org/10.1016/j.ress.2003.10.015.

Holen, S., Utne, I., Holmen, I., and Aasjord, H. (2016). Occupational safety in aquaculture. part 1 and 2. *Marine Policy (submitted)*.

Kruusmaa, M., Gkliva, R., Tuhtan, J.A., Tuvikene, A., and Alfredsen, J.A. (2020). Salmon behavioural response to robots in an aquaculture sea cage. *Royal Society Open Science*, 7(3), 191220. doi:10.1098/rsos.191220. URL https://royalsocietypublishing.org/doi/abs/10.1098/rsos.191220.

Olsen, S. (2020). Mistet undervannsdrone til 100.000 kroner: Skjermen ble plutselig bare helt svart. URL https://ilaks.no/mistet-undervannsdrone-til-100-000-kroner-\skjermen-ble-plutselig-bare-helt-svart/.

Rist-Christensen, I. (2016). Autonomous robotic intervention using ROV.

SINTEF (2012). Merdrov - utvikling av rov som verktøy for automatiserte simultane notoperasjoner i oppdrettsmerd. URL https://www.sintef.no/prosjekter/merdrov-utvikling-av-\rov-som-verktoy-for-automatis/.

Solem, A.J.L. (2017). Analysis of current rov operations in the norwegian aquaculture - reducing risk in exposed aquaculture operations.

The Norwegian Ministry of Agriculture and Food (2009). Animal welfare act. URL https://www.regjeringen.no/en/dokumenter/animal-welfare-act/id571188/.

Thorvaldsen, T., Holmen, I.M., and Moe, H.K. (2015). The escape of fish from norwegian fish farms: Causes, risks and the influence of organisational aspects. *Marine Policy*, 55, 33 – 38. doi:https://doi.org/10.1016/j.marpol.2015.01.008. URL http://www.sciencedirect.com/science/article/pii/S0308597X15000196.

Utne, I., Schjølberg, I., and Holmen, I. (2015). Reducing risk in aquaculture by implementing autonomous systems and integrated operations. doi:10.1201/b19094-481.

Østen Jensen, Tim Dempster, E.B.T.A.F. (2010). Escapes of fish from norwegian sea-cage aquaculture:causes, consequences, and prevention. *Aquaculture Environment Interactions*, 1(1), 71–83. doi:10.3354/aei00008.

Appendix B

Autonomous Aquaculture: Implementation of an autonomous mission control system for unmanned underwater vehicle operations

Henning Ødeby Karlsen[1], Herman Biørn Amundsen[2], Walter Caharija[2], Martin Ludvigsen[1]

[1] Norwegian University of Science and Technology - Department of Marine Technology

[2] SINTEF Ocean AS; Trondheim, Norway

Introduction

Remotely operated vehicles (ROVs) are vital for the aquaculture industry and are used for inspection, maintenance, and repair (IMR) operations such as net inspections, mooring inspections, net cleaning, and biomass monitoring. It is well documented that industry workers at fish farms are exposed to risks, and as the industry grows and fish farms are moved further offshore, the risk for the people involved increases [1]. More autonomous ROVs can help mitigate this risk by performing some of the most dangerous underwater operations autonomously. Efficiency could also be improved with autonomy. Per today most ROVs have some autonomous abilities, but none can perform complete aquaculture IMR tasks autonomously. ROVs operating in aquaculture net cages face several unique challenges compared to most other oceanic industries, such as manoeuvring inside dynamic structures and operating in the wave zone.

This paper proposes a system architecture for autonomous mission control of ROVs operating in the aquaculture domain. We also present an analysis of which IMR operations in aquaculture have the potential to be conducted by autonomous ROVs, including their implementation requirements for an autonomous mission control system. The different risks connected to the use of autonomous ROVs in aquaculture are also analysed.

Materials and methods

Motion control systems for autonomous marine vehicles are usually constructed of three independent blocks, *guidance, navigation,* and *control,* which together makes up a GNC system [2]. However, when the complexity of the vehicle mission and the number of possible mission states increases, these three blocks can become tightly coupled and there may exist functionalities which a GNC-system may not describe.

The GNC architecture is therefore seen in context with classical robot control architectures such as deliberative, reactive, hybrid, and behaviour-based architectures for mission management [3]. The proposed mission control system consists of a hybrid approach building on the GNC-architecture, such that an autonomous ROV can both react to unplanned events and keep an overall plan of the task it is performing. Furthermore, the mission control system is divided into different layers for implementation. The layers proposed are a planning layer, an executive layer, and a behavioural control layer. The planning layer handles high level control and deliberation, such as keeping track of the mission progress and moving from point to point. The planning layer is implemented in SINTEF Ocean's GUI for ROV control, Aqueous [4]. The behavioural layer consists of lower-level control behaviours such as dynamic positioning, net following and path following. It also ensures reactivity by handling exceptions in execution of the low-level behaviours. The behavioural layer is implemented in the simulation framework FhSim [5]. The executive layer is divided between Aqueous and FhSim, where FhSim handles reactive transitions while Aqueous handles deliberative and manual transitions. For visualizing and understanding the conditions and transitions between the different behaviours a finite state machine (FSM) is utilized.

Results

The proposed mission control system has been implemented on an Argus Mini ROV and verified in simulations using the simulation framework FhSim. Furthermore, in a proof-of-concept experiment, elements of the autonomous mission control system were successfully tested at SINTEF ACE, a full-scale industrial fish farm laboratory located off the coast of mid-Norway. The trial targeted aquaculture inspection operations and during the experiments the ROV performed an autonomous inspection of a net pen using a Doppler velocity log (DVL) [6]. The autonomous mission control system was able to handle loss of sensor measurements through reactive behaviour. Furthermore, using deliberative behaviour the mission control system tried to keep track of and manage the autonomous net inspection task. The results are promising, even though they show that better localization of the net pen is required.

Discussion

The implemented system architecture for autonomous mission control allows the ROV to perform both deliberate and reactive behaviours. The results show that the proposed GNC-based architecture utilizing a layered hybrid approach can be expanded and improved for increased autonomy in the future, for instance, by adding new behaviours to the autonomous mission control system.

Conclusion and further work

An analysis of ROV operations in aquaculture that have the potential to be autonomized have been conducted. Furthermore, a proposed mission control system for ROVs operating autonomously in aquaculture net cages has been implemented and tested in a full-scale experiment. The results are promising and demonstrates the potential in autonomous ROV operations in aquaculture. Further work includes the implementation and field validation of more autonomous functionality, as well as improving the net inspection tracking performance.

References

[1] T. Thorvaldsen, T. Kongsvik, I. M. Holmen, K. Størkersen, C. Salomonsen, M. Sandsund and H. V. Bjelland, “Occupational health, safety and work environments in Norwegian fish farming - employee perspective,” *Aquaculture,* p. 735238, 2020.

[2] T. I. Fossen, Handbook of Marine Craft Hydrodynamics and Motion Control, Wiley, 2011.

[3] B. a. K. O. Siciliano, Springer Handbook of Robotics, Springer Publishing Company, Incorporated, 2016.

[4] E. Dahlen, S. A. Aubert, H. Grønbech, H. Backer, T. S. Gunstad, A. Knutsen and H. H. Syverinsen, “TDT4290 Aqueous,” November 2019. [Online]. Available: https://github.com/eirikdahlen/TDT4290_Aqueous. [Accessed April 2020].

[5] B. Su, K. Reite, M. Føre, K. Aarsæther, M. Alver, P. Endresen, K. D. H. J. W. Caharija and A. Tsarau, “A Multipurpose Framework for Modelling and Simulation of Marine Aquaculture Systems,” in *Proceedings of the ASME 2019 38th International Conference on Ocean, Offshore and Arctic Engineering. Volume 6: Ocean Space Utilization*, Glasgow, Scotland, UK, 2019.

[6] P. Rundtop and K. Frank, “Experimental evaluation of hydroacoustic instruments for ROV navigation along aquaculture net pens,” *Aquaculture,* pp. 143-156, 2016.

Bibliography

[1] AKVA group. AKVA FNC8 2.0. https://www.akvagroup.com/pen-based-aquaculture/net-cleaning/akva-fnc8, 2019.

[2] H. B. Amundsen. Robust nonlinear ROV motion control for autonomous inspections of aquaculture net pens. Master's thesis, NTNU, 2020.

[3] H. B. Amundsen and W. Caharija. Interviews with people experienced within ROVs in aquaculture, 2021.

[4] H. B. Amundsen, W. Caharija, and K. Y. Pettersen. Autonomous ROV inspections of aquaculture net pens using DVL. *IEEE Journal of Oceanic Engineering (submitted)*, 2020.

[5] Argus Remote Systems AS. Argus Mini. https://www.argus-rs.no/argus-rovs/11/argus-mini, 2019.

[6] P. J. Ashley. Fish welfare: Current issues in aquaculture. *Applied Animal Behaviour Science*, 104(3):199 – 235, 2007. Fish Behaviour and Welfare.

[7] J. Bannister, M. Sievers, F. Bush, and N. Bloecher. Biofouling in marine aquaculture: a review of recent research and developments. *Biofouling*, 35(6):631–648, 2019. PMID: 31339358.

[8] M. Ben-Ari and F. Mondada. *Finite State Machines*, pages 55–61. Springer International Publishing, Cham, 2018.

[9] M. S. Branicky. *Introduction to Hybrid Systems*, pages 91–116. Birkhäuser Boston, Boston, MA, 2005.

[10] M. Carreras, J. Batlle, P. Ridao, and G. Roberts. An overview on behaviour-based methods for auv control. *IFAC Proceedings Volumes*, 33(21):141 – 146, 2000. 5th IFAC Conference on Manoeuvring and Control of Marine Craft (MCMC 2000), Aalborg, Denmark, 23-25 August 2000.

[11] M. Colledanchise and P. Ögren. *Behavior trees in robotics and AI: An introduction*. CRC Press, 2018.

[12] E. Dahlen, S. A. Aubert, H. Grønbech, H. Backer, T. S. Gunstad, A. Knutsen, and H. H. Syverinsen. TDT4290 Aqueous. https://github.com/eirikdahlen/TDT4290_Aqueous, Nov 2019.

[13] F. Dukan. ROV motion control systems. 2014.

[14] FAO. The state of world fisheries and aquaculture 2020. sustainability in action. *Food and Agriculture Organization of the United Nations*, 2020.

[15] T. I. Fossen. *Handbook of marine craft hydrodynamics and motion control*. John Wiley & Sons, 2011.

[16] T. O. Fossum, M. Ludvigsen, S. M. Nornes, I. Rist-Christensen, and L. Brusletto. Autonomous robotic intervention using ROV: an experimental approach. In *OCEANS 2016 MTS/IEEE Monterey*, pages 1–6. IEEE, 2016.

[17] A. Girault, B. Lee, and E. A. Lee. Hierarchical finite state machines with multiple concurrency models. *IEEE Transactions on computer-aided design of integrated circuits and systems*, 18(6):742–760, 1999.

[18] H. Grønbech, H. B. Amundsen, W. Caharija, and W. Azam. Imc for fhsim aquaculturerobotics. Technical note, restricted, version 3, 2021.

[19] P. Gullestad, S. Bjørgo, I. Eithun, A. Ervik, R. Gudding, H. Hansen, R. Johansen, A. Osland, M. Rødseth, I. Røsvik, et al. Effektiv og bærekraftig arealbruk i havbruksnæringen–areal til begjær. *Rapport fra et ekspertutvalg oppnevnt av Fiskeri-og kystdepartementet. ISBN*, pages 978–82, 2011.

[20] B. O. A. Haugaløkken. *Autonomous Technology for IMR Operations in the Norwegian Aquaculture*. PhD thesis, NTNU, 2020.

[21] S. Holen, I. Utne, I. Holmen, and H. Aasjord. Occupational safety in aquaculture. part 1 and 2. *Marine Policy (submitted)*, 2016.

[22] J. E. Hopcroft, R. Motwani, and J. D. Ullman. Introduction to automata theory, languages, and computation. 2006.

[23] Ø. Jensen, T. Dempster, E. Thorstad, I. Uglem, and A. Fredheim. Escapes of fish from norwegian sea-cage aquaculture: causes, consequences, and prevention. *Aquaculture Environment Interactions*, 1:71–83, 08 2010.

[24] M. Kruusmaa, R. Gkliva, J. A. Tuhtan, A. Tuvikene, and J. A. Alfredsen. Salmon behavioural response to robots in an aquaculture sea cage. *Royal Society Open Science*, 7(3):191220, 2020.

[25] A. Kurt and Ü. Özgüner. Hierarchical finite state machines for autonomous mobile systems. *Control Engineering Practice*, 21(2):184–194, 2013.

[26] LSTS. IMC: Laboratório de sistemas e tecnologia subaquática. `https://lsts.pt/software/63`, 2020.

[27] Ministry of Agriculture and Food. Norwegian animal welfare act. `https://www.regjeringen.no/en/dokumenter/animal-welfare-act/id571188/`, Jul 2009.

[28] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Hähnel, T. Hilden, G. Hoffmann, B. Huhnke, D. Johnston, S. Klumpp, D. Langer, A. Levandowski, J. Levinson, J. Marcil, D. Orenstein, J. Paefgen, I. Penny, and S. Thrun. Junior: The stanford entry in the urban challenge. pages 91–123, 01 2009.

[29] S. J. Ohrem, E. Kelasidi, and N. Bloecher. Analysis of a novel autonomous underwater robot for biofouling prevention and inspection in fish farms. In *2020 28th Mediterranean Conference on Control and Automation (MED)*, pages 1002–1008. IEEE, 2020.

[30] M. Olsson. Behavior trees for decision-making in autonomous driving, 2016.

[31] Ö. Özkahraman and P. Ögren. Combining control barrier functions and behavior trees for multi-agent underwater coverage missions. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 5275–5282. IEEE, 2020.

[32] Ö. Özkahraman and P. Ögren. Combining control barrier functions and behavior trees for multi-agent underwater coverage missions. In *2020 59th IEEE Conference on Decision and Control (CDC)*, pages 5275–5282. IEEE, 2020.

[33] M. Powers, D. Wooden, M. Egerstedt, H. Christensen, and T. Balch. The sting racing team's entry to the urban challenge. 01 2012.

[34] S. Rabin. *Game AI pro: collected wisdom of game AI professionals.* CRC Press, 2017.

[35] K.-J. Reite, M. Føre, K. Aarsæther, J. Jensen, P. Rundtop, L. Kyllingstad, P. Endresen, D. Kristiansen, V. Johansen, and A. Fredheim. FHSIM — time domain simulation of marine systems. *Proceedings of the International Conference on Offshore Mechanics and Arctic Engineering - OMAE*, 8, 06 2014.

[36] P. Rundtop and K. Frank. Experimental evaluation of hydroacoustic instruments for rov navigation along aquaculture net pens. *Aquacultural Engineering*, 74:143–156, 2016.

[37] S. S. Sandøy. *Probabilistic Localization and Mapping using Unmanned Underwater Vehicles for Aquaculture Operations.* PhD thesis, NTNU, 2020.

[38] D. Saul and I. Tena. BP's AUV development program, long term goals-short term wins. In *OCEANS 2007*, pages 1–5. IEEE, 2007.

[39] M. L. Seto. *Marine robot autonomy.* Springer Science & Business Media, 2012.

[40] B. Siciliano and O. Khatib. *Springer Handbook of Robotics.* Springer Publishing Company, Incorporated, 2nd edition, 2016.

[41] A. J. L. Solem. Analysis of current ROV operations in the norwegian aquaculture-reducing risk in exposed aquaculture operations. Master's thesis, NTNU, 2017.

[42] Sperre AS. Fish net sewing machine. `https://sperre-as.com/portfolio/fish-net-sewing-machine/`.

[43] C. I. Sprague, Ö. Özkahraman, A. Munafo, R. Marlow, A. Phillips, and P. Ögren. Improving the modularity of auv control systems using behaviour trees. In *2018 IEEE/OES Autonomous Underwater Vehicle Workshop (AUV)*, pages 1–6. IEEE, 2018.

[44] N. Standard. Remotely operated vehicle (ROV) services. *U-102, Rev 3*, 1, 2016.

[45] B. Su, K.-J. Reite, M. Føre, K. G. Aarsæther, M. O. Alver, P. C. Endresen, D. Kristiansen, J. Haugen, W. Caharija, and A. Tsarau. A multipurpose framework for modelling and simulation of marine aquaculture systems. In *International Conference on Offshore Mechanics and Arctic Engineering*, volume 58837, page V006T05A002. American Society of Mechanical Engineers, 2019.

[46] The Norwegian Directorate of Fisheries. Statistics for aquaculture. `https://www.fiskeridir.no/English/Aquaculture/Statistics`, 2019.

[47] United Nations. Goal 2: Zero hunger – united nations sustainable development.

[48] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, et al. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, 25(8):425–466, 2008.

[49] C. Urmson, J. A. Bagnell, C. Baker, M. Hebert, A. Kelly, R. Rajkumar, P. E. Rybski, S. Scherer, R. Simmons, S. Singh, et al. Tartan racing: A multi-modal approach to the darpa urban challenge. 2007.

[50] I. Utne, I. Schjølberg, and I. Holmen. Reducing risk in aquaculture by implementing autonomous systems and integrated operations. 09 2015.

Henning Ødeby Karlsen

Towards Autonomous ROVs in Aquaculture

NTNU
Norwegian University of
Science and Technology