Master's thesis

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

Tore Øystese

Underwater Path Planning using Sonar Imaging Data

Master's thesis in Marin Teknikk
Supervisor: Martin Ludvigsen
June 2019

Tore Øystese

Underwater Path Planning using Sonar Imaging Data

Master's thesis in Marine Technology
Supervisor: Martin Ludvigsen
June 2019

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

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

MASTER THESIS MARINE CYBERNETICS

SPRING 2019

FOR

STUD. TECH. Tore Øystese

Title: Underwater Path Planning using Sonar Imaging Data

Work Description

Remotely Operated Vehicles (ROVs) are commonly used for subsea inspection and light intervention task as they have high manoeuvrability and payload capacity. Today, all operations are carried out by the control of an operator. By using increasing the level of autonomy, one can reduce the reliance on a surface operator, reducing operation costs and increasing the complexity of the missions. As the ROVs often operate in areas of subsea installations, the ROVs are prone to manoeuvre around objects on the sea floor. To be able to manoeuvre in complex areas, there is a need for a set path. Since each area can be different and each start and goal locations differs on different missions at different locations. To solve this problem, a path planning algorithm can be used. A map of the current area can be created, and a path can be planned. To minimize mission time, a shortest path planning algorithm can be implemented.

Scope of Work

1. Literature study on and Remotely Operated Vehicles, focusing on:
    a. Use of sonar for obstacle detection and map generation of underwater environment
    b. Path planning in underwater environment.
    c. Underwater control and navigation
    d. Path optimization algorithms
2. Literature study of the Voronoi path generation
3. Development work;
    a. Implement an algorithm for path planning based of sonar imaging.
    b. Implement a shortest path algorithm.
4. Experimental Work
    a. Testing of algorithms in simulations (LabView).
    b. Field testing in Trondheimsfjorden.

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 11th, 2019.

Supervisor: Martin Ludvigsen

Abstract

Remotely Operated Vehicles are widely used in the offshore industry in different applications such as interventions, exploration and observation. To reduce the reliance on surface ships and remote operators, there could be value in increasing the autonomous capabilities of ROVs. This thesis aims to explore the possibility of using sonar imaging data from a Forward Looking Multibeam Sonar as the basis of a underwater path planning scheme using Voronoi Diagrams and Dijkstra Algorithm. The contribution of this thesis is utilizing these aspects together, combing feature extraction from sonar imaging and path planning using the Voronoi Diagrams. The input for this path planning is a sonar video stream containing information about the underwater area in front of the underwater vessel, in this case a ROV. The output is a path for the ROV to follow through areas with obstacles.

The input of the path planning scheme is the a sonar image. The image is processed using different image processing techniques. To obtain the best results possible, several different filtering and thresholding techniques were applied. The results from the filtering showed that applying a mean filtered with 5×5 kernel size and a binary threshold with a threshold value of 127 yielded the best results. When processed, several feature detector were tested. The feature detector yielding the best results were the ORB detector, which utilizes a version of the FAST detection algorithm. The feature detector returned a set of keypoints. To transform the detected keypoints to an obstacle map, a conversion were needed. This were done by transforming the detected features of the image to a obstacle map, using the range indicators in the sonar image. The obstacles were used as an input to a Voronoi Diagram, which created a graph tree of feasible paths. To select the shortest path, a Dijkstra algorithm were implemented.

To test the feasibility of the paths generated, the ROV following the created path were simulated. To do this, the ROV were simulated using the control system and simulator created by the AUR-lab in the graphical programming language LabVIEW. The ROV were path following were simulated using a constant jerk guidance model with waypoints as the input. The paths were simulated and the results were presented.

Four simulations were performed, using four different sonar images as input. The ROV were in general able to follow the created paths, with small deviation between the estimated and desired state. Exceptions occurred when the ROV where commanded to follow paths with hard turns, i.e large differences in path direction. The maximum deviation were measured at 7 meters from the path.

From the results obtained, it appears that the concept of underwater path planning using sonar imaging can have some promise, but not without its problems. It turns out that there are difficulties with confirming that all the obstacles are detected, and thus difficulties confirming the feasibility in regards of collision avoidance for the path planning. However, in a scenario were there are multiple smaller obstacles in the area in front of the ROV, the use of Voronoi path planning using sonar imaging could be feasible.

Sammendrag

Fjernstyrte undervannsfartøy (ROV) er mye brukt i offshore-industrien til forskjellige oppgaver og gjøremål som tekniske inngrep, observasjoner og utforskning. For å redusere avhengigheten av overflateskip og piloter som fjernstyrer fartøyet, kan det være muligheter i å utvide de autonome egenskapene til ROVer. Denne avhandlingen tar for seg å undersøke mulighetene for å bruke sonarbildedata fra en Forward Looking Multibeam Sonar som grunnlag for en undervanns baneplanlegging ved hjelp av Voronoi diagram og Dijkstra algoritmen. Resultatet av denne oppgaven er å utnytte disse aspektene sammen, ved å kombinere informasjonen utvunnet fra sonardata med baneplanlegging ved hjelp av Voronoi Diagram. Grunnlaget for baneplanleggingen er informasjon fra en sonarvideo, som inneholder informasjon om området foran undervannsfartøyet, som i dette tilfellet er en ROV. Resultatet fra baneplanleggingen er en trygg bane ROVen kan følge gjennom områder med hindringer.

Grunnlaget til baneplanleggingsalgoritmer er et sonarbilde. Bildet blir prosessert ved hjelp av forskjellige bildebehandlingsteknikker. For å sikre de beste resultatene ble det anvendt flere forskjellige filtrerings- og bildeterskel-teknikker. Resultatene fra filtreringen viste at bruk av et median filtrer med 5 5 kjernestørrelse og en binær terskel med en terskelverdi på 127 ga de beste resultatene. Etter bildebehandlingen, ble flere detektorer testet. Detektoren som gav det beste resultatet, var ORB-detektoren, som bruker en versjon av FAST deteksjonsalgoritme. Resultatet fra detektoren var et sett med nøkkelpunkter markert på sonarbildet. For å transformere de detekterte nøkkelpunktene til et kart over hindringer, trengtes en konvertering. Dette ble gjort ved hjelp av rekkeviddeindikatorene i sonarbildet. Hindringene i kartet ble brukt som grunnlaget til et Voronoi diagram, som lager et graftre av mulige baner. For å velge den korteste banen, ble Dijkstra-algoritmen implementert, med banelengde som avgjørende faktor.

For å teste om banene er gjennomførbare, ble ROVen simulert og styrt gjennom den foreslåtte banen. ROVen ble simulert ved hjelp av et kontrollsystem og en simulator laget av AUR-laben i det grafiske programmeringsspråket LabVIEW. Banene ble simulert og resultatene fra simulering ble presentert.

Fire simuleringer ble utført ved å bruke fire forskjellige sonarbilder som utgangspunkt. Generelt klarte ROVen og følge de gitte banene, med små avvik mellom estimert og ønsket posisjoner. Unntakene var i situasjoner der ROVen måtte følge baner med harde svinger, dvs. stor forskjell i baneretning. Maksimalt avvik ble målt til 7 meter fra den gitte banen.

Fra de oppnådde resultatene ser det ut til at konseptet undervannsplanlegging ved hjelp av sonarbilder har potensialet, men dog ikke uten problemer. Det viser seg at det var vanskeligheter med å bekrefte at alle hindringene var oppdaget, og at det dermed er vanskeligheter med å bekrefte at man unngår kollisjon i løpet av banen. I et scenario der det er flere hindringer i området rundt en ROV, kan bruk av Voronoi-baneplanlegging ved hjelp av sonarbilder imidlertid være mulig.

Preface

This master thesis is the final work in the master program of Marine Technology at the Department of Marine Technology, a part of the Faculty of Engineering. The master thesis was written and created with the basis of the specialization Marine Cybernetics, and the work done has been conducted in the spring of 2019. This thesis is the result of the accumulation of knowledge gained throughout my time at this master's program, with some new aspects earlier foreign to me.

The topic of the thesis was established through cooperation with my supervisor Martin Ludvigsen and the work done during the Project Thesis written in the Fall of 2018. The topic of sonar imaging and path planning fitted well with my interest in both computer science and underwater robotics. The work done has been both challenging and rewarding and has left me with valuable experience.

I want to thank Martin Ludvigsen for his guidance and support through the work with this thesis. I would also like to show gratitude to Stein Nornes for his advice and help in the LabView environment.

[illegible]

Tore Øystese, 9th of June

Contents

List of Figures

List of Tables

Abbreviations

AUR	Applied Underwater Robotics
AUV	Autonomous Underwater Vehicles
BODY	Body fixed Coordinate Frame
CB	Constant Bearing Guidance
CMROV	Consumer Market Remotely Operated Vehicle
CO	Center of Origin
CPM	Control Plant Model
DOF	Degree(s) of Freedom
DoG	Difference of Gaussian
DoGSS	Difference of Gaussian Scale Space
DVL	Doppler Velocity Log
FAST	Features from Accelerated and Sements Test
FLS	Forward Looking Sonar
GNG	Growing Neural Gas Algorithm
GUI	Graphical User Interface
HIL	Hardware in the Loop
HiPAP	High Precision Acoustic underwater Positioning
ILOS	Integral Line of Sight Guidance
IMU	Inertial Measurement Unit
KF	Kalman Filter
LBL	Long Base Line
LOS	Line of Sight
MIMO	Multi-Input Multi-Output

NED	North-East-Down
NTNU	Norwegian University of Science and Technology
OCROV	Observation Class Remotely Operated Vehicle
ORB	Oriented Brief Rotated Fast
PID	Proportional-Integral-Derivative
ROV	Remotely Operated Vehicle
SIFT	Scale-Invariant Feature Transform
SNR	Signal-to-Noise power Ratio
SSBL	Super Short Base Line
SURF	Speeded up Robust Features
TCP	Transmission Control Protocol
UAV	Unmanned Aerial Vehicle
USBL	Ultra Short Base Line
WCROV	Working Class Remotely Operated Vehicle

Chapter 1

Introduction

This thesis aims to further develop the autonomous capabilities of the Remotely Operated Vehicles (ROV) operations. The topic is based on implementing an underwater sonar path planning scheme, combining sonar imaging, Voronoi Diagrams and Dijkstra algorithm to create paths for the ROV.

1.1 Motivation

The capitalization of offshore resources is an old industry ranging from offshore fishing to offshore hydrocarbon extraction and to rather new ventures such as offshore seabed mineral extraction. The industry has developed a lot, and new technologies has helped solving and facing new challenges. The growing need for hydro carbons in the world has pushed the offshore Oil and Gas industry to harsher environments, with colder weather and deeper oceans. The need for subsea installations has increased as a result of this and along side the need for operation and maintenance for these underwater installation. Furthermore, the ocean is vast and is largely unexplored, and the research on the biological life in large depths unavailable for human divers are needed.

The use of Unmanned Underwater Vehicles can tackle many of these challenges. This includes both Autonomous Underwater Vehicles (AUV) and Remotely Operated Vehicles(ROV). AUVs are largely applied in the mapping and exploration of the under water environment and functions autonomously without human intervention. The use of Remotely Operated Vehicles is today applied in the industry for tasks in environments unreachable for humans. The use of ROVs ranges from exploration of seabed points of interests, research of underwater biology systems, maintenance and operation of underwater hydrocarbon plants, drilling wells and more. As indicated by the name, the ROV are operated by a pilot stationed in a surface ship, which is responsible for launching and retrieving the ROV. The ROV is tethered to the ship so that the pilot is able to operate it. Although necessary for the current technologies, the surface ships operation can be expensive. The cost of operating a surface ship can be in the range of tens of thousands of dollars[9]. These costs directly influences the cost of the use of the ROVs, and makes operations such as exploration and maintenance more costly. In addition, the use of an sea-surface operator has its limits. The operation relies heavily on the experience, knowledge and capability of the operator, which can be compromising for the integrity of the operation.

The possibility for applying autonomy in the scope of Unmanned Underwater Vehicles is fueled by said possibilities and limits. This can be applied for both AUVs and ROVs. The AUVs, although autonomous, is facing challenges still, and is not developed to its fullest capability. The ROVs already used by to days industry, can be further developed to do tasks autonomously. With autonomous algorithms for the different stages of the ROV mission, the operations can rely less on the operator, and if developed successfully, result in safer and more consistent operations.

1.2 Objectives

The motivation of this thesis is based on further develop the autonomous capabilities of a ROV in mission. Thus, this thesis explore the possibilities in combining sonar image data from a multibeam forward looking sonar and Voronoi Diagrams in a path planning problem in an underwater environment. The path is then simulated using a constant jerk guidance algorithm to ensure successful tracking of said paths. The objective for the thesis work is

1. Test feature extraction from sonar imaging data
2. Develop an algorithm for detection of obstacles from a sonar imaging video stream
3. Develop an algorithm that utilizes the detected obstacles from the sonar imaging video to create a Voronoi Diagram
4. Develop an interface for simulations in LabVIEW utilizing the AUR-lab control and simulation environment.
5. Simulate the output paths in the AUR-lab control and simulation environment.

The objectives are conducted using Python programming language and the LabVIEW simulation suit developed by the AUR lab at the Institute of Marine Technology. Further, the thesis gives an overview of the dynamics of the ROV, the control systems created by the AUR-lab, digital image features and the basis of path planning and Voronoi diagram generation. The algorithm created should take a sonar image or sonar video stream input and output a set of waypoints used for tracking for the ROV. The simulations are performed under the assumption that the global start and goal points are defined, making the algorithm limited to local path planning.

1.3 Limitations

During the work on the thesis several limitation were discovered. The availability of sonar imaging data were very limited, resulting in the work done on the sonar imaging data actually acquired were not tested on other data sets. In addition, the sonar data used for this thesis is not from the same model as that attached to the ROV.

Further, a sea trail were to be conducted, but were cancelled due to technical difficulties with both the newly acquired ROV Minerv II and the research vessel Guneriues, which were submitted to a dry dock during the winter for maintenance and further installations.

1.4 Background

The vessels described under the term Unmanned Underwater Vehicles is largely divided into two categories. The two categories consist of the Autnomous Underwater Vehicles and the Remotely Operated Vehicles. The AUV is un-tethered and fully autonomous, while the ROV is tethered to a surface vessel and is controlled by an on board operator. Figure 1.1 shows the classification of the underwater vehicles.

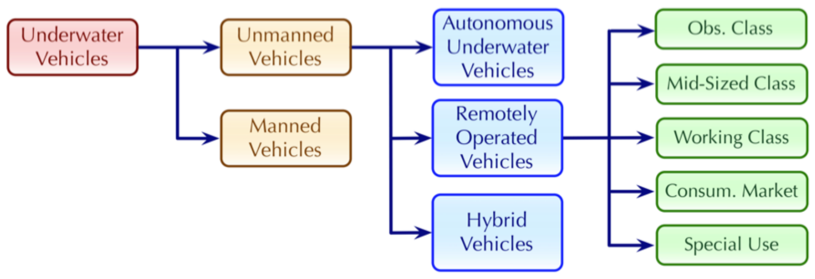


Figure 1.1: Classification of Under Water Vehicles [1]

Remotely Operated Vehicles is the collective term for the under water vessels operated through an umbilical cable connecting the vehicle to a support vessel. The ROVs are often characterized by their geometrical shape and their abilities. The main field of operation for the ROV is monitoring, mapping, intervention, sampling and investigation of areas of interests. The ROV moves, in general, slower than AUVs, and due to the umbilical connection, the ROV often has smaller spacial capabilities. However, the ROVs are rewarded by their versatility in their application, their high payload capacities regarding sensors and their lack of power restrictions. The ROV is used for a wide set of applications and is classified in regards of their ability. The classification is based largely on the size of the vessels and their accompanying ability to intervene and carry payloads. The ROVs are generally, based on their class, able to perform observation, mapping and exploration, cargo transportation and interventions. The definition of the different ROV classes are established in [10], and can be divided into five different classes: Observation Class, mid-sized class, working class, consumer market and special use. The classes are defined as follow.

- **Observation Class:** The OCROV Is the smallest class of industrial ROV and is typically weighing less than 100 kg. The observation class is mainly used in shallower waters. Their payload capability is limited to cameras small manipulators, and is thus used to inspection and mapping.
- **Mid-Sized** The MSROV shares similar aspects of the smaller observation class but is in all larger, with higher payload capacity and is applicable to larger ocean depths. The manipulator is generally hydraulic based and can therefor be used to light interventions and general inspection.
- **Working Class** WCROV is the largest and heaviest ROV class. Through the umbilical the working class ROV receives higher voltage power. The thrusters are hydraulic based and the manipulator capable of six degrees of freedom intervention, capable of heavy loads.
- **Consumer Market** The CMROV is the definition of the ROVs sold to end consumer, often used as a recreational activity. Small enough to carry, they often include on-board batteries and camera for underwater exploration and video capture.
- **Special Use** The special use vehicles include all ROV not categorized by the for-mentioned classes, which would include special design made to carry out special task

The branch of hybrid vehicles denoted in Figure 1.1, regards the vehicle not categorized in the two other branches. ROVs today is, indicated by their name, still mostly controlled and operated from the support surface vessel. The hybrid category includes ROVs with autonomous features, reducing the need for operator intervention. Although tethered, control algorithms applied to the ROVs has the ability to make them perform parts of the operation autonomously. They are not strictly classified, but is regarded as Autonomous Remotely Operated Vehicles (AROV). With the governing goal of making all ROVs fully autonomous, or rather, make all the today's ROV operation done autonomously with out surface support, this branch is beginning to grow. The autonomy of different systems can largely be discussed. There are many different interpretations of the term autonomy, and the subsequent degrees of autonomy. The Committee on Autonomous Vehicles in Support of Naval Operations [11] introduced a definition, which was later modified by [12]

Level 1: Manual Operation
The human operator directs and controls all mission functions, while the vehicle still flies autonomously.

Level 2: Management By Consent
The system automatically recommends actions for selected functions and prompts the operator at key points for information or decisions.

Level 3: Management By Exception
The system automatically executes mission-related functions. The operator is alerted to function progress and exceptions, and may override or alter parameters and cancel or redirect actions within defined time lines.

Level 4: Fully Autonomous
The operator is alerted to function progress, while the system automatically executes mission related functions when response times are too short for operator intervention.

The marine ROV operation can be divided into different states. The first state is launching, where the ROV is put in the water and a control of the system is performed. The second state is decent where the vehicle descends into the sea column and reaches it desired depth. The next state is transit, where the vessel moves in the horizontal frame to reach the general location of the object of interest. The last two steps are camera tracking to reached the desired position and the inspection or intervention. Figure 1.2 shows the states of a ROV mission.

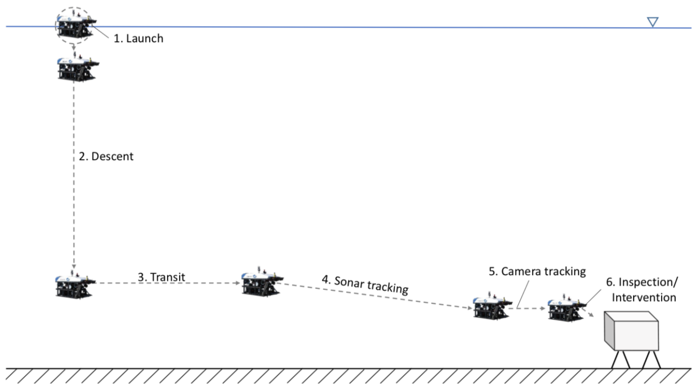


Figure 1.2: The different states of a ROV mission [2]

To be able to navigate known or unknown environments by generating way points, and to successfully be able to follow said paths is important in steps to autonomy. To be able to perform autonomous operations, a significant part of the problem is to generate paths based on sensor data. Autonomous Path planning is a well known problem and has been well reviewed for robotic application. As the electromagnetic waves does not propagate through water, the underwater environment the lack of communication with the Global Navigation Satellite System. This toughens the problem of navigation and path planning. In order to perform this, other senors data needs to be applied. In addition, the lack of maps of the underwater environment often results in a unknown local areas.

1.5 ROV Minerva 2

The Applied Underwater Robotics Lab (AUR Lab), a part of the Institute of Marine Technology at the NTNU, owns and operates several underwater vehicles, including multiple ROVs. The newly acquired ROV is called *Minerva 2*. *Minerva 2* is a working class vessel at 1500 kg, and is rated for operation at depths up to 3000 meters below sea surface. The ROV is situated with base line system, Doppler Velocity Log and Inertial Measurement Unit sensors used for positioning. ROVs are generally slightly positive buoyant, so that in case of accidents or unprepared loss of control the vessel will reemerge by itself. ROVs are generally capable of translations in 3 degrees of freedom along with control of heading. Roll and pitch motions are left uncontrolled, but are by design stable. However, the newly acquired ROV of the AUR lab at the department of Marine Cybernetics is capable of control in all 6 degrees of freedom, due to its larger thruster capacity.

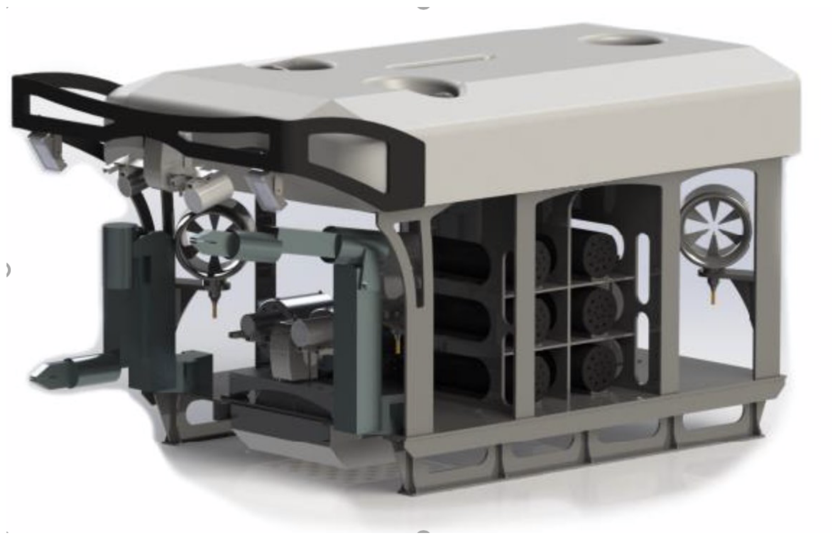


Figure 1.3: CAD-model of Minerva II

The ROV is fitted with a set of sensors used for navigation, where different sensors gives a measurement for different states of the ROV. The sensors used for navigation includes a transponder, DVL, pressure sensors and cameras. The ROV is fitted with a transponder, which is part of the USBL (Ultra Short Base Line), SSBL (Super Short Base Line), and LBL (Long Base Line) system. The transponder is connected on the ROV and receives acoustic signals, which carries a specific direction and face, from transducers connected to either a support vessel or at the seafloor. The base line systems are based of triangulation and is used for positioning of the ROV. By knowing the positions of the transducers, and using triangulation, an measurement of the ROV position is retrieved.

The DVL (Doppler Velocity Log) is a sensor used for velocity measurements. The DVL systems sends an array of four sonar beams in a downward direction towards the sea bed, and by measuring the frequencies of the echo of the sonar beams, the velocity of the ROV relative to the sea floor can be measured.

The Inertial Measurement Unit is a collection of sensors, consisting of accelerometer, gyros and magnetometer. The accelerometer measures the acceleration along the three body axis, xyz. The gyro measures the turning rate of the vehicle about the three axis xyz. The magnetometer is used as a heading measurement, and measures the direction of the ROV in relation to the magnetic north pole, which in turn by knowing the location of the ROV and the relation between the magnetic and geographical north pole, gives a compass reading relative to North from the NED-frame.

The pressure sensor is used as a depth measurement. The pressure sensors measure the ambient pressure of the surrounding of the ROV, and from that calculates the depth of the ROV relative to the sea surface. The salinity of the water and latitude of the ROV influences these readings, but can be compensated if the variables are known.

The ROV is also equipped with a forward-looking wideband multibeam Sonar from Norbit. The sonar is mounted in the bow of the ROV, used for retriving information about the area in front of the vessel. The sonar works on a frequency of 400 kHz with a 7°-180° horizontal viewing angle. The multibeam sonar returns a spazial map of the surroundings infront of the ROV, based on the intensity of the feedback of the sonar acoustic impulses that return to the sonar head. The NORBIT WBMS sonar has a max range of 250 meters with a nominal range of 100 meters at the operating frequency of 400 kHz.

1.6 Previous Work

1.6.1 Path planning

The path planning problem is well known challenge for robotics and autonomous systems. It has been widely studied for different applications. The problem of path planning is twofold, know the position of the vehicle, the goal position of the vehicle, and the location of obstacles that may hinder the path. For applications on land, the problem of robot location, goal location and obstacles is often well accounted for. In robotics in general, the solution to these problems is often based off of which sensor data is readily available. For robotic applications above sea surface, a wide number of accurate and fairly easily available sensors and sensor data is present. Example of this includes Lidar, positioning using the Global Navigation System and good availability of accurate information of the area of interest, either through inspection or pre-produced maps.

In the subsea environment most of the available sensors and methods used on land is not longer available, so the path planning problem is further toughens the problem. If information about the robot environment can be made available through sensors, different approaches to the path planning problem can be presented.

In the doctoral thesis of Candeloro [13], the use of Voronoi Diagrams in the path-planning problem is proposed. Along with Professors Lekkas, Sørenesen and Fossen, Candeloro developed a path planning algorithm combining Voronoi diagrams and Fermat's spirals in making the curvature-continuous, where the Fermat's spirals were used to smooth the paths [14]. In the article the problem of path planning in a static environment with a-priory knowledge of the are were proposed. Further development, in a cooperation with Professor Sørensen, Professor Lekkas and Postdoctoral Fellow Hedge, included an extension for 3D dynamic path-planning using Voronoi Diagrams for UUVs [15]. This included a re-planning phase which allowed for dynamic environments and collision avoidance. The final result were presented in a collaboration with Professor Lekkas and Professor Sørensen in [16]. In the paper the path planning concerns the use of LOS guidance for under actuated vehicles, such as autonomous surface vessels and AUVs.

1.6.2 Sonar image feature extraction

In the article "Underwater vehicle path planning using a multi-beam forward looking sonar" [17] by D. Lane and Y. Pelliot, introduces an algorithm that uses the sonar imaging data to produce a path through obstacles in an underwater environment. The algorithm proposed in the article uses the image as a input and outputs a path. The image is processed and when the obstacles are identified, the obstacles are described by constructive solid geometry method. From the information of the obstacles, the path is generated by solving a nonlinear search of feasible paths in the constructed map.

When the image is input in the algorithm, several steps is made to identify the obstacles. Firstly, the image is segmented. Generally, sonar images contain a lot of noise, and this is necessary to be able to produce sufficient clarity for the obstacle extraction. The segmentation is done by performing first performing a 7x7 mean filter on the image, reducing the noise at a lower computational cost. The second step is thresholding of the image. The threshold contributes to a clearer difference between the objects in the frame compared to the background, or more generally between difference shades in the image. After segmentation, the paper introduces a method for feature extraction of the scanned image, to find the obstacles and the object suited for tracking. The obstacles is labeled with relevant meta data, constructed with the constrictive solid geometry method. For the path planning algorithm, the obstacles are presented as elliptical objects. The path planning is then reduced to finding the shortest path without crossing said elliptical objects.

In the publication by Calceran and Djapic et al. titled "A Real-time Underwater Object Detection Algorithm for Multi-beam Forward Looking Sonar" [18], the feature extraction of a multi beam forward looking sonar is furthered explored. As with Lane's and Pelliot's publication, the article uses different levels of image processing to ensure success in feature extraction. Firstly, the a part of the scanned sonar image is chosen as the area of interest. With the smaller image, the image is represented by as integral image. The main purpose of this is to reduce the computational time of the image processing, as the representation allows for lower computational time than that of the original image, when calculating the background, i.e the image representation of what potential obstacles is compared to. The background image is found by finding the average pixel value of the integral image. By computing an average of each image to find the corresponding background, the algorithm can be applied to other images at areas where the seabed and thus the background is different.

In the thesis written by Tsz Kit Chiu, titled "Sonar tracking and obstacle avoidance for navigation of ROV"[19] a further work with object detection is proposed. The thesis aims at developing a system for detecting a object and tracking it. To detect the object, a sonar image is used. He proposes four different methods for detecting the object, namely Harris Corner detection, SIFT, SURF and ORB. The four methods is tested on a sonar scanned image, and the SURF algorithm yields the best result. The SURF algorithm is a local feature detector that is built on the SIFT algorithm, but has been proven as faster and more accurate. Further in the thesis the detected object is used in a tracking algorithm, with the sonar image used as a map.

In the paper titled "A Voting-Based Approach for Fast Object Recognition in Underwater Acoustic Images" [20] by G.L. Foresti and C.S. Regazzoni et al., a voting based approach for object recognition is proposed. The voting approach yielded good results in regards of acoustic images, an were used as a basis of estimating the vehicle attitude. The voting algorithm were based on the edge discontinuities of the underwater acoustic images. In spite of the often undesired effects in underwater acoustic imagery such as blurring, speckle noise and distortions, the voting approach yielded robust results.

In the article "Feature Tracking in Video and Sonar Subsea Sequences with Applications"[21] by Trucco Petillot, describes a way of locating and tracking obstacles with the help of a multi beam sonar sequence, fast enough to support real-time ROV and AUV operations. The algorithm is built upon a two level segmentation of each sonar image where the first segmentation smooths the picture and lowers noise levels, where as the second level of segmentation chooses areas of interest in the image and does segmentation on these areas alone. The algorithm proved good results, and allowed for online computation of objects in frame.

1.6.3 Use of sonar in path planning

”Development of an Obstacle Detection and Avoidance System for ROV” [22] author Ø. Grefstad introduces a obstacle detection system based of raw sonar data. The thesis aims at developing an algorithm for detecting underwater obstacles using sonar data and then to develop a map and path planning scheme. The map is constructed as a 2 dimensional occupancy grid and is used to form a feasible path.

1.7 Thesis Contribution

The contribution of this thesis is focused on the testing of using sonar imaging data as a basis of a Voronoi Diagram based path planning solution. The thesis focuses on the use of different image processing techniques for a sonar imaging data set to establish a set of obstacles. The obstacles are then used for the creation of a Voronoi diagram. Further, through the thesis, a set of simulation is performed to test the feasibility of the paths created. In addition, an algorithm were developed to perform the path planning autonomously, creating a Python programming code to process the image and a input system in LabVIEW to ensure a automatic transformation between the python code and the LabVIEW environment used for simulation and control of the ROV.

1.8 Thesis Outline

The thesis is divided into eight further chapters. The first chapters regards the different theories applied in the thesis, while the later regards the applied methods, simulations and results.

Chapter 2 gives an introduction to the theory behind the sonar application in underwater environments along with the different applications of sonar technology in the under water environment. Chapter 3 gives an overview of the theory behind the theory of the ROV dynamics and control system. Chapter 4 regards the theory behind digital images, as well as the theory behind the image processing techniques applied through the thesis.

Chapter 5 gives a overview of the path planning problem, and specifically how the utilization of the Voronoi diagram aims to solve the path planning problem. Further, an introduction to the Dijkstra algorithm is given. Chapter 6 gives a overview of the development and methods applied in the thesis, and how the simulations were conducted. Lastly Chapter 7 presents the results found in the work with this thesis, which is discussed in Chapter 8. Chapter 9 presents a conclusion to the thesis and its results, and proposes further work that can further improve the solution and better its integrity.

Chapter 2

Sonar

Sonar, which is an abbreviation of Sound Navigation And Ranging, is a technological principal used for communication and collecting information of surroundings by utilizing acoustic waves. Unlike signal waves used in radar and electro-optical systems, sonar utilizes the use of propagating acoustic waves in the medium it operates. As it does not rely on sending signals through the medium, but utilizing the medium itself, it can be applied as a substitution or addition to electromagnetic waves in wide set of applications.

There are generally two types of sonar systems, namely passive and active sonar. The former is as the name suggest a passive system, used for listening to acoustic waves of its surroundings. The latter on the other consists of a transmitter and a receiver, and actively sends out acoustic signals and in turn reads the returning signals.

Common application for sonar systems include navigation, fish and marine detection, localization and tracking of underwater objects, communication, passive monitoring and research of underwater biological and geological areas.

2.1 Underwater acoustics

The basis of the sonar technology is the acoustic waves. An acoustic wave can be described as a sound waves in a unspecified medium. The waves propagates as pressure differences in the medium, causing the mediums molecules to form a wave motion. The speed of propagation, i.e the speed of which the next molecule in the wave direction is disturbed can be described, as with electro-magnetic waves as

$$c = \lambda f \tag{2.1}$$

where c is the speed of the propagating wave, λ is the wave height and f is the frequency of the wave motion, given as the number of complete motions is performed by the wave per time. The speed of propagation in the acoustic wave is much slower than that of electro-magnetic waves but with the speed of 1500 m/s in water. The speed is affected by a set of physical traits of the medium, and one can describe the speed as a function of these traits.

$$c = c(T, p, S) \tag{2.2}$$

where T is the temperature, p is the pressure and specifically for water S is the salinity of the medium. The pressure in water is directly correlated with the operating depth of the sonar.

2.2 Sonar Equations

There are basically two types of sonar technology systems used, namely the passive and the active sonar. As the name suggests, the former actively sends out signals and listens to the return echo signal, while the passive listens for ambient signals from the environment. The following section explains the theory behind them.

In general, the sonar system is explained by an equation describing the different physical phenomenons that occur while employing the sonar technology.

2.2.1 Passive sonar

As mentioned, the passive sonar listens to the ambient sound in an environment. The application for this differs between listening to marine mammals exerting acoustic waves as a way of communication, listening to acoustic waves produced by other vessels, and in particular for this case in localization of vessels, both surface and subsea, during warfare.

The sonar equation is a method of combining all the parts of the sonar process [23]. The passive sonar equation is introduced to be able to define the terms and the notation for the sonar equation . The main part of the sonar equation is the signal-to-noise power ratio (SNR). SNR can be described as

$$S/N = \frac{Signal}{PropLoss}\frac{ArrayGain}{TotalLoss} \tag{2.3}$$

which, when converted to a dB scale would become, from Urick (1996) [24].

$$SNR = SL - TL - NL + DI \tag{2.4}$$

where SNR is the signal-to-noise ratio, SL is the radiated signal, TL is the transmission or propagation loss, NL is the total noise and DI is the directivity index or array gain. Equation 2.3 states that a radiated signal originating at a target is diminished by transmission or propagation loss as it travels to the receiver [23]. The noise level can deteriorate the signal while the the array gain at the receiver can improve the SNR. Equation 2.4 represents the same equation but transformed to a decibel gain scale, represented by 2.5, where a is the output in dB and b is the input from equation 2.3.

$$a = 10log(b) \tag{2.5}$$

2.2.2 Active Sonar

As with the passive sonar equation, the active sonar equation also calculates the signal-to-noise ratio. As opposed to the passive sonar, the active sonar emits a signal. The signal travel from the transmitter, hits the object and reflects back. This leads to the conclusion that the propagation loss term in the equation is doubled. Further, a new term is introduced, namely the target strength. Thus the active sonar equation can be described as

$$SNR = SL - 2TL + TS - NL + AG \tag{2.6}$$

where AG is the array gain and TS is the target strength. The array gain takes into account directional and correlation characteristics of the various components of the noise field. The total noise, NL is everything but the echo signal of which is undesired, such as ambient noise and reverberation and is explained further later in the section.

The radiated signal SL is in the active sonar the signal emitted from the sonar, while in the passive sonar equation its the source signal level that the sonar is listening to. For an omnidirectional sound source, the source level (dB) can be described as

$$SL = 10log\frac{I}{I_0} \tag{2.7}$$

where I is the intensity of source the source, and I_0 is the intensity in water, both given in W/m^2

The target strength TS is the ability of a target to return an echo. The return noise not exerted from the target, which hinders the target strength, is the noise term reverberation. The target strength can be described as

$$TS = 10log\frac{I_r}{I_i} \tag{2.8}$$

where I_r is the reflected acoustic intensity at the reference distance, while I_i is the incident acoustic intensity.

2.2.3 Sonar losses

Transmission loss

The transmission loss is the loss of signal integrity and strength when traveling through a medium. It is defined as the loss of intensity between a point of interest and a reference distance. It can be described as

$$TL = -10log\frac{I(R,D)}{I_0} \tag{2.9}$$

where $I(R,D)$ is the acoustic intensity at a range R and a depth D and I_0 is the acoustic intensity at the reference distance. As the ambient pressure of the medium the sonar signal is traveling through is dependant on the submersion depth, the transmission loss can be said to be dependant on the ambient pressure at a range R and a depth D

The transmission loss is dependant on several factors, such as the spreading of the signals and the absorption or attenuation of the signal. The latter describes the absorption of the signal by the medium, here ocean. In the ocean, there are three mechanisms that cause absorption . The mechanisms are shear viscosity, volume viscosity and the ionic relaxation for boric acid and magnesium sulfate. This causes the attenuation to be temperature, salinity, pH, depth and pressure dependant. [23]

Surface and sea bottom

The transmission losses can also include the interaction with boundaries to the operating environment. In an ocean environment this includes the interaction between the sea surface and the sea bottom. As the sea surface both is a dynamic boundary and is a boundary between the two mediums water and air, the losses can be significant. Since the ROV missions typically is done with a significant distance to the surface, the losses in this case can be neglected.

For the interaction with the sea bottom on the other hand can not, as most ROV missions happen at or near the sea bottom. The sea bottom also shifts medium with the water, but is a solid object. Significant energy penetrates through the sea bottom and the return signal is dependant on the ocean bottom characteristics.

Reverberation

The reverberation is the return signals echoed from the other objects than that of the object of interest. The sources of reverberation in the ocean environment include the boundaries ocean surface and the sea bottom and the volume of the medium. The volume reverberation includes marine animals, bubbles and abnormalities in the water and other debris and particles.

Ambient Noise

Ambient noise is a term to describe the noises associated with the environment of operation. In general, these noises are out of the control of the system designers, but needs to be accounted for in the model. Ambient noise include noise from turbulence, shipping vessels, wave motion, thermal agitation of the medium, underwater and subsurface seismic disturbances, surface weather, including wind and rain, marine life and the motion of ocean ice. The ambient noise is defined as the occurring noise when identified sources are excluded.

2.3 Underwater Acoustics Application

As briefly mentioned, the sonar technology has a wide set of use, depending on how the sonar technology is implemented. There are several different ways to characterizes the different sonar technologies. One is the for mentioned division between the active and the passive. For the most part, this section will discuss the active sonar. There are also other ways of applying the underwater acoustics signals such as in base line localization and communication.

2.3.1 Single Beam Sonar

As the the active sonar is reliant on a transducer, the number of transducer used in a sonar system can determine the type of sonar systems. The single beam sonar uses one single transducer. Often, the same transducer is used for both signal transmission and signal reception. The transducer consists of a hydrophone and a projectors, which is equivalent to the speak and microphone used in the air, which converts acoustic energy into electrical energy and vice versa. The transducer can have several possible configuration, namely fixed, mechanically rotating or mechanically translating. These division determines if and how the beam is moved, thus determining its use case. For the fixed single beam sonar is typically used for depth assertion. The mechanically rotating and translating sonar uses a motor to broaden their field of view, but are restrained by the output time, the distance traveled by the output signal before the transducer position and rotation can be changed. The side-scan sonar is an implementation of single beam sonar transducer.

2.3.2 Side-scan Sonar

Side-scan sonars are first and foremost considered as visualisation tools, providing acoustic images of the seabed [4]. The sidescan sonar are usually installed on a tow unit behind ships or on a ROV/AUV. The principle of the side scan sonar is the emission of a narrow signal in horizontal direction and which is generated by two side mounted sensor. The linear translation of the tow vessel or underwater vessel allows the sidescan to sweep over the area during the scan.

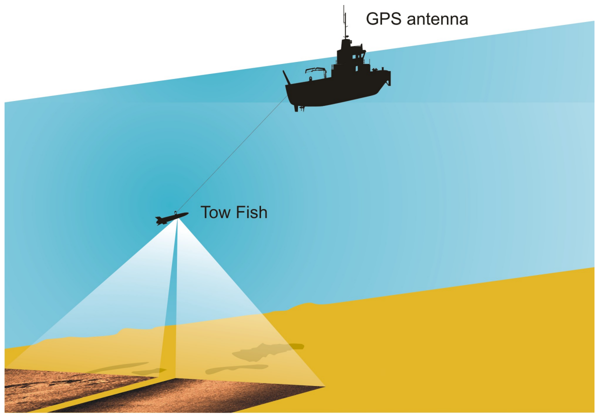


Figure 2.1: Sidescan Principle [3]

The echo received along time will represent the bottom reflectively along the swathe, and returns information about irregularities along the sea bottom. The system structure is the same as for single-beam echo sounders, with frequencies in the scale of hundreds of kHz.

2.3.3 Multi Beam Sonar

The multi beam echo sounder consists of a set of single beam transducer that operates cooperatively. Instead of transmitting and receiving a single beam, the multi beam sounder transmits and receives a fan of smaller beams, with typically widths of 1-3°. This in turn gives higher fidelity and covered area, but is more computational intensive and the equipment generally more expensive.

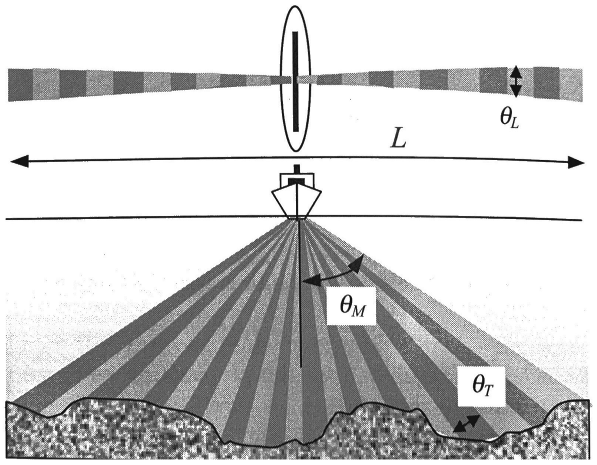


Figure 2.2: Multibeam Echo Sounder Principle [4]

2.3.4 Sonar Imaging

To be able to ekstract information directly from a sonar data set, sonar imaging is used. The principle of sonar imaging is the same for both multibeam sounders and sidescan sonars. The signal returned from the seabed is recorded as a function of time. The intensity of the return signal represents the characteristics of the scanned seabed. The return signals is then constructed as an image in regards of its intensity. The intensity can be highlighted with difference in black/white levels or colors. Figure 2.3 shows a sonar image from a forward looking multibeam sonar.

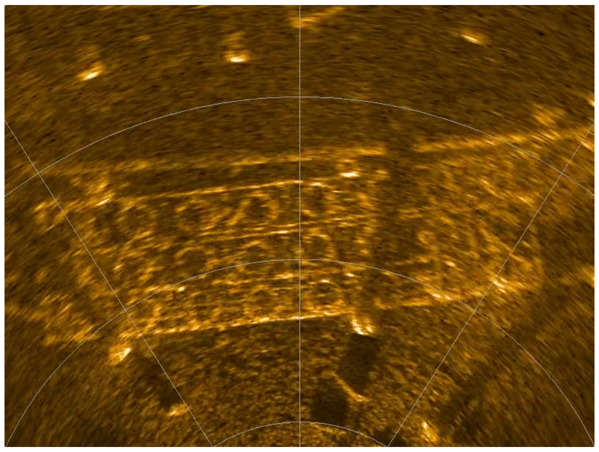


Figure 2.3: Sonar Image

2.3.5 Base line localization

Underwater acoustics can, in addition to sonar technology, be used as a tool for underwater localization as well. The basis for this is a system in a area consisting of stationary transmitters and receivers. The system of receivers and transmitters uses the idea of triangulation to determine the position of a moving vessel, typically, a ROV. The different baseline systems can be described by the difference in the distance between the transducers which is the basis of the triangulation.

Long Baseline acoustic systems

The long baseline acoustic system is based of transducers fixed in position on offshore structures or on the ocean floor. Being a permanent solution, this system is used in areas of offshore activity. The larger distance between the transducers gives better accuracy, with the disadvantage of a fixed solution.

Short Baseline Acoustic system

The short base line system is generally used with vessels. The transducers are placed on the vessels submerged body, and communicates with the underwater vessel. The distance between the transducers ranges from 5 to 20 meters.

Super-Short Baseline Acoustic systems

The super-short baseline system is an integrated solution, needing one module with several transducers. This means that the transducer distance is less than 25 cm. The ease of use is counter with worse accuracy than both the long and short baseline systems.

Chapter 3

ROV Control System

3.1 ROV Dynamics

The ROV is based on operating below surface. The control system applied to the ROV determines the action of the ROV and controls the desired motion and counters the undesired motion caused by external loads. To control the ROV, the dynamics of the ROV needs to be modeled. The dynamics of the ROV consists of rigid body dynamics, kinematics of the frames on which the dynamics is founded, hydrodynamic forces acting on the vessel as well as external forces such as commanded forces from the thrusters, forces from the umbilical cord connecting the ROV to the surface ship, forces from the manipulator of the ROV and external forces from floating ice and propagating waves. In this section, the different aspects of the ROV dynamics will be presented, and denoted on the form of Fossen's Robot-like Vectorial Model to in detail, develop a process plant model for the ROV.

The vessel is able to move in 6 degrees of freedom, namely the translations surge sway heave, and the rotations roll, pitch yaw. To express the positions, velocities, and accelerations of the vessel, frames of reference needs to be created. For marine vessels, operating in local environments, the convention is to use the inertial frame *North-East-Down*(NED). The NED frame is usually defined as a tangent plane on the surface of the Earth. It consist of three axis $\{n\} = [x_n, y_n, z_n]$ where x_n points north, y_n points east and z_n points downwards normal to the tangent plane and earth surface. The NED coordinate frame is used to describe the position of the vessel. To describe the forces acting on the vessel as well as the velocity and acceleration, the *Body-fixed-frame* is used. The BODY frame is fixed to the vessel, with axis $\{b\} = [x_b, y_b, z_b]$. The origin is usually set in the center of gravity of the vessel, with x_b direction directed from aft to fore, y_b directed to starboard and z_b directed from top to bottom.

Kinematics

Since the underwater vessels operate in six degrees of freedom, rotations and angel rates is also necessary to describe along side position and velocity. The rotation and position is describe in the vector $\boldsymbol{\eta}$, defined in the NED frame, and the velocity and angular rate is described by the vector $\boldsymbol{\nu}$ defined in the body frame. In sum, the vectors $\boldsymbol{\eta}$ and $\boldsymbol{\nu}$ is defined as

$$\boldsymbol{\eta} = [x \ y \ z \ \phi \ \theta \ \psi] \tag{3.1}$$

$$\boldsymbol{\nu} = [u \ v \ w \ p \ q \ r] \tag{3.2}$$

With the 6 degrees of freedom defined for both position and velocity, the dynamics of the vessel can be described. As there are forces defined and acting in both frames, there needs to be a translation between the two frames. To do so, there is defined a relationship between the derivative of the position in the NED frame and the Body frame velocity. This relation can be described as

$$\dot{\boldsymbol{\eta}} = \mathbf{J}_{\Theta}(\boldsymbol{\eta})\boldsymbol{\nu} \tag{3.3}$$

where $\mathbf{J}_{\Theta}$ is the transformation matrix based on Euler Angles. In matrix form 3.3 can be denoted as

$$\begin{bmatrix} \dot{\mathbf{p}}^{n}_{n/b} \\ \dot{\boldsymbol{\Theta}}_{nb} \end{bmatrix} = \begin{bmatrix} \mathbf{R}^{n}_{b}(\boldsymbol{\Theta}_{nb}) & \mathbf{0_{3x3}} \\ \mathbf{0_{3x3}} & \mathbf{T}(\boldsymbol{\Theta}_{nb}) \end{bmatrix} \begin{bmatrix} \mathbf{v}^{b}_{b/n} \\ \boldsymbol{\omega}^{b}_{b/n} \end{bmatrix} \tag{3.4}$$

where $\mathbf{R}^{n}_{b}(\boldsymbol{\Theta}_{nb})$ is a rotation matrix based on the principle rotation around the coordinate axis zyx specified by the Euler angles $[\phi, \theta, \psi]$. $\mathbf{T}(\boldsymbol{\Theta}_{nb})$ is the angular transformation matrix which describes the relation between the body fixed angular velocity vector $\boldsymbol{\omega} = [p, q, r]^{T}$ and the Euler rate vector $\boldsymbol{\Theta} = [\dot{\phi}, \dot{\theta}, \dot{\psi}]$.

Rigid Body Kinetics

With the kinematics of the vessel described, the dynamics can be found. For a vessel operating in an underwater environment, there are several forces acting on the body. By defining the body as a rigid body, the dynamics of the vessel can be found. By assuming rigid body one assumes that the vessel can not be deformed by forces acting on the body. This extends to that forces acting, in any direction of the body, the forces causes the body to experience either an acceleration or a reacting force. To fully describe the kinetics of the body, rotation needs to be accounted for. To do so, the Newton Euler equations can be used. The Newton Euler equation is based on a conservation of both forces and moments acting on the body, chosen at an arbitrary center CO. The rigid body forces and moments acting on the ROV can be formulated as

$$\mathbf{M}_{RB}\dot{\boldsymbol{\nu}} + \mathbf{C}_{RB}(\boldsymbol{\nu})\boldsymbol{\nu} = \boldsymbol{\tau} \tag{3.5}$$

where $\mathbf{M}_{RB}$ is the rigid body mass matrix, $\mathbf{C}_{RB}\boldsymbol{\nu}$ is the centripetal Coriolis matrix and τ is the external force and moment vector defined as $\boldsymbol{\tau} = [X_N, Y_N, Z_N K_N, M_N, N_N]$ working on the vessel. The mass matrix consists of the mass and the moments of inertia for the vessel and is found as

$$\mathbf{M}_{RB} = \begin{bmatrix} m\mathbf{I}_{3x3} & -m\mathbf{S}(\mathbf{r}^{b}_{g}) \\ m\mathbf{S}(\mathbf{r}^{b}_{g}) & \mathbf{I}_{b} \end{bmatrix} \tag{3.6}$$

where $S(\cdot)$ is a skew matrix, $\mathbf{r}^{b}_{g}$ is the vector of center of gravity in body frame and $\mathbf{I}_b$ is the moment of inertia around the CO. The full matrix form of $\mathbf{M}_{RB}$ can then be found as

$$\mathbf{M}_{RB} = \begin{bmatrix} m & 0 & 0 & 0 & mz_g & -my_g \\ 0 & m & 0 & -mg_z & 0 & mx_g \\ 0 & 0 & m & my_g & -mx_g & 0 \\ 0 & -mz_g & my_g & I_x & -I_{xy} & -I_{xz} \\ mz_g & 0 & -mx_g & -I_{yx} & I_y & -I_{yz} \\ -my_g & mx_g & 0 & I_{zx} & -I_{zy} & I_z \end{bmatrix} \tag{3.7}$$

The second term in the rigid body equation $\mathbf{C}_{RB}$ denotes the rigid body centripetal Coriolis matrix, which is the result of the rotation of the earth. The resulting force is proportional to the vessel velocity and is derived in [25] as

$$\mathbf{C}_{RB} = \begin{bmatrix} \mathbf{0}_{3x3} & -m\mathbf{S}(\boldsymbol{\nu}_1) - m\mathbf{S}(\mathbf{S}(\boldsymbol{\nu}_2)\mathbf{r}_n^b) \\ -m\mathbf{S}(\boldsymbol{\nu}_1) - m\mathbf{S}(\mathbf{S}(\boldsymbol{\nu}_2)\mathbf{r}_n^b) & -m\mathbf{S}(\mathbf{S}(\boldsymbol{\nu}_2)\mathbf{r}_n^b) - \mathbf{S}(\mathbf{I}_b\boldsymbol{\nu}_2) \end{bmatrix} \tag{3.8}$$

Restoring Forces

The ROV operates in the ocean environment, meaning that there are hydro-mechanics forces acting on the vessel, often denoted as hydrodynamic and hydrostatic forces. The latter is found as the buoyancy of the vessel. In addition to gravity, there is a counter force acting on the vessel from the buoyancy. The buoyancy is defined as the mass of the displaced water multiplied with the gravitational acceleration working on the body. In marine crafts, the convention is to gather the buoyancy and gravitational forces and moments to a restoring term. The gravitational forces act on the center of gravity while the buoyancy forces act on the center of gravity of the displaced water mass, which corresponds to the center of buoyancy of the vessel. If these two points coincide, there will be a net sum of zero moments from these forces. If not, there will be moments working on the body. The vector of gravitational force and vector of buoyancy force is defined as

$$f_g^n = \begin{bmatrix} 0 \\ 0 \\ mg \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ W \end{bmatrix} \tag{3.9}$$

$$f_b^n = -\begin{bmatrix} 0 \\ 0 \\ \rho g \Delta \end{bmatrix} = -\begin{bmatrix} 0 \\ 0 \\ B \end{bmatrix} \tag{3.10}$$

If one lets the the center of gravity be defined as $r_g^b = [x_g, y_g, z_g]$ in the body frame and the center of buoyancy be defined as $r_b^b = [x_b y_b z_b]$ in the body frame, the restoring term $\mathbf{g}(\boldsymbol{\eta})$ can be defined as

$$\mathbf{g}(\boldsymbol{\eta}) = -\begin{bmatrix} \mathbf{f}_g^b + \mathbf{f}_b^b \\ \mathbf{r}_g^b \times \mathbf{f}_g^b + \mathbf{r}_b^b \times \mathbf{f}_b^b \end{bmatrix} = -\begin{bmatrix} \mathbf{R}_b^n(\boldsymbol{\Theta}_{nb})^{-1}(\mathbf{f}_g^b + \mathbf{f}_b^b) \\ \mathbf{r}_g^b \times \mathbf{R}_b^n(\boldsymbol{\Theta}_{nb})^{-1}\mathbf{f}_g^n + \mathbf{r}_b^b \times \mathbf{R}_b^n(\boldsymbol{\Theta}_{nb})^{-1}\mathbf{f}_b^n \end{bmatrix} \tag{3.11}$$

The rigid body equation of motion can now be set as

$$\begin{aligned} \mathbf{M}_{RB}\dot{\boldsymbol{\nu}} + \mathbf{C}_{RB}(\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{g}(\boldsymbol{\eta}) &= \boldsymbol{\tau_{ext}} \\ \dot{\boldsymbol{\eta}} &= \mathbf{J}_{\Theta}\boldsymbol{\nu} \end{aligned} \tag{3.12}$$

Equation of Motion

In total the vectorial model dynamics can now be expanded, with all the relevant hydrodynamic forces established along side the rigid body dynamics and kinematics. The external forces is now reduced to true external forces acting on the vessel. The first and foremost is the commanded control force from the ROVs actuators, normally thrusters. In addition, environmental forces such as forces from ice in the water is included along side wave excitation, if operating in the wave zone. ROVs are often tethered, which causes the umbilical to act on the vessel with a force. The 6 DOF dynamic equation of motion for the ROV can be described as

$$\begin{aligned} \dot{\boldsymbol{\eta}} &= \mathbf{J}_{\Theta}\boldsymbol{\nu} \\ \mathbf{M}_{RB}\dot{\boldsymbol{\nu}} + \mathbf{C}_{RB}\boldsymbol{\nu} + \mathbf{M}_{A}\dot{\boldsymbol{\nu}} + \mathbf{C}_{A}\boldsymbol{\nu} + \mathbf{D}_{L}\boldsymbol{\nu} + \mathbf{D}_{NL}|\boldsymbol{\nu}|\boldsymbol{\nu} + \mathbf{G}(\boldsymbol{\eta}) &= \boldsymbol{\tau}_{ext} \end{aligned} \tag{3.13}$$

where τ_{ext} is the external forces and is described by a linear superposition of the forces

$$\boldsymbol{\tau}_{ext} = \boldsymbol{\tau}_{thrust} + \boldsymbol{\tau}_{current} + \boldsymbol{\tau}_{wave} + \boldsymbol{\tau}_{ice} + \boldsymbol{\tau}_{thet} \tag{3.14}$$

The forces from current can be included in the hydrodynamic radiation terms. This is done by introducing the velocty $\boldsymbol{\nu}_r$ which is the relative velocity between the current and the vessel velocity. The relative velocity is denoted as

$$\boldsymbol{\nu}_r = \boldsymbol{\nu} - \boldsymbol{\nu}_c \tag{3.15}$$

where $\boldsymbol{\nu}_c$ is the velocity of the current in body frame. The relative velocity can now be implemented in 3.13. As the forces acting on the body from the current directly correlates with the hydrodynamic forces, the relative velocity is only applied to the hydrodynamic terms $\mathbf{M}_A$, $\mathbf{C}_A$, $\mathbf{D}_L(\boldsymbol{\nu})$ and $\mathbf{D}_{NL}(\boldsymbol{\nu})$. By introducing this velocity, the forces from the current can be removed and the resulting equation of motion in 3.16 is established.

$$\begin{aligned} \dot{\boldsymbol{\eta}} &= \mathbf{J}_{\Theta}\boldsymbol{\nu} \\ \mathbf{M}_{RB}\dot{\boldsymbol{\nu}} + \mathbf{C}_{RB}\boldsymbol{\nu} + \mathbf{M}_{A}\dot{\boldsymbol{\nu}}_r + \mathbf{C}_{A}\boldsymbol{\nu}_r + \mathbf{D}_{L}\boldsymbol{\nu}_r + \mathbf{D}_{NL}|\boldsymbol{\nu}_r|\boldsymbol{\nu}_r + \mathbf{G}(\boldsymbol{\eta}) &= \boldsymbol{\tau}_{ext} \\ \boldsymbol{\tau}_{ext} &= \boldsymbol{\tau}_{thrust} + \boldsymbol{\tau}_{wave} + \boldsymbol{\tau}_{ice} + \boldsymbol{\tau}_{thet} \end{aligned} \tag{3.16}$$

Equation 3.16 is based as the process plant model for the entire ROV dynamics.

3.2 ROV Control System

The control system for the ROV has been developed by the AUR-lab at NTNU. The development and implementation of the control system is largely explained in [1] and [26]. The purpose of this section is to give the reader an input of how its implemented and the main parts of the control scheme is explained.

For the ROV to be used in the multi-DOF environment multiple separate control systems has been developed. The two primary control systems is the velocity and position control. This calculates the desired velocity and position for the ROV in the horizontal plane, used for transit and movement of the ROV. The second is the altitude or depth controller. This controls the ROV to the desired altitude above the seabed. The control scheme of the control system of the ROV is presented in Figure 3.1. The control system is based on a standard setup with a state observer estimating the states, a guidance module, feeding the desired states in to the controller along side with the estimated state from the observer. The controller calculated the desired force vector and the thrust allocation block calculates the desired shaft speed of the different thrusters.

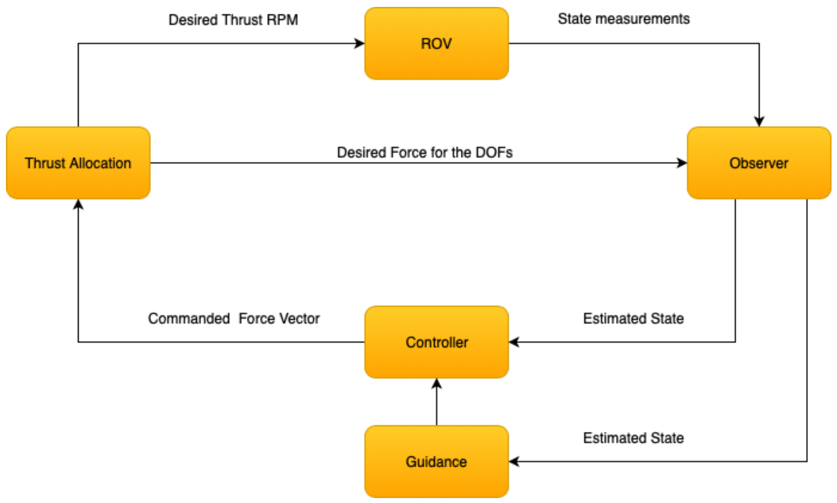


Figure 3.1: Simplified Control Scheme of the ROV

3.2.1 Control Plant Model

When developing control systems and running simulations for control systems, there is often a need for a reduction of the Process Plant Model(PPM). The PPM describes as accurately as possible the dynamics of the system, often with nonlinear terms and relations. The result of this is that the computational power needed for simulation is very high, as well as the complexity is hard to model. This causes the need for a Control Plant Model (CPM). The CPM is used for simulation and testing of control systems, and is used in the development for both the observer and the controller. The CPM derives only the most essential from the PPM, and gets rid of complex terms that affects the dynamics to a lesser extent. The CPM for the ROV can be described as

$$\begin{aligned} \dot{\boldsymbol{\eta}} &= \mathbf{R}_b^n \boldsymbol{\nu} \\ \mathbf{M}\dot{\boldsymbol{\nu}} + \mathbf{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{D}(\boldsymbol{\nu})\boldsymbol{\nu} + \mathbf{g}(\boldsymbol{\eta}) &= \boldsymbol{\tau} + \mathbf{R}_n^b(\psi)\mathbf{b} + \mathbf{w}_m \\ \dot{\mathbf{b}} &= -\mathbf{T}_b^{-1}\mathbf{b} + \mathbf{w}_b \end{aligned} \tag{3.17}$$

where **b** is the bias term representing the un-modeled dynamics of the system. In general, the CPM is used in the design and development for the different parts of the control system, in addition to simulation. The simulation based on the CPM is less computational intensive and can be simulated faster than real time. The PPM on the other hand is mostly used for high fidelity simulations running at or below real time, and Hardware In the Loop testing (HIL).

3.2.2 Controller

Based on the desired state provided by the guidance module, the controller calculates the needed force and moments for the vessel to obtain the desired state. The estimated states from the motion control system. There are several control algorithms applicable to drive the ROV to the desired state. The controller is divided into sub modes, a position control, controlling heading and depth, and a speed control, speed in surge and sway. The motion control is a Multiple Input Multiple Output(MIMO) nonlinear Proportional-Derivative-Integral-control(PID-control) combined with a feed-forward term, and can be described as

$$\boldsymbol{\tau} = \boldsymbol{\tau}_{PID} + \boldsymbol{\tau}_{FF} \tag{3.18}$$

The nonlinear PID is described by

$$\boldsymbol{\tau}_{PID} = -\mathbf{J}^T(\boldsymbol{\eta})(\mathbf{K}_p\tilde{\boldsymbol{\eta}} + \mathbf{K}_p\mathbf{J}(\boldsymbol{\eta})\tilde{\boldsymbol{\eta}} + \mathbf{K}_i\int_0^t \tilde{\boldsymbol{\eta}}dt) \tag{3.19}$$

where $\mathbf{K_p}, \mathbf{K_i}, \mathbf{K_d}$ denotes the proportional, derivative and integral matrices of the PID. The nonlinear part is the transformation matrix $\mathbf{J}(\boldsymbol{\eta})$. The feed forward term is defined as

$$\boldsymbol{\tau}_{FF} = \mathbf{M}\mathbf{a}_d + \mathbf{C}(\boldsymbol{\nu}_d)\boldsymbol{\nu}_d + \mathbf{D}(\boldsymbol{\nu}_d)\boldsymbol{\nu}_d \tag{3.20}$$

3.2.3 Thruster

For ROVs in general, thrusters are the only actuators. The thrusters is capable of producing a thrust force in constant or variable directions, depending on if they have an axis of rotation or are fixed. The force of which the thrusters are capable of producing is generally dependant on the torque and speed of the shaft of the thrusters. Minerva 2 is situated with 7 thrusters, four of which are placed in the horizontal xy-plane in the body frame, and three placed in the vertical zx-plane in the body frame [27]. All thrusters are fixed, but tilted slightly to avoid water being flushed through the ROV. In order to find the force needed by each thruster to satisfy the commanded force and moments from the controller, an thrust allocation algorithm is used. The main goal of the thrust allocation algorithm is to translate the force vector, including forces and moments, to the desired shaft speed and direction of each thruster, combined with optimizing the solution in regards of energy consumption. Figure 3.2 shows a block diagram over the thrust allocation problem.

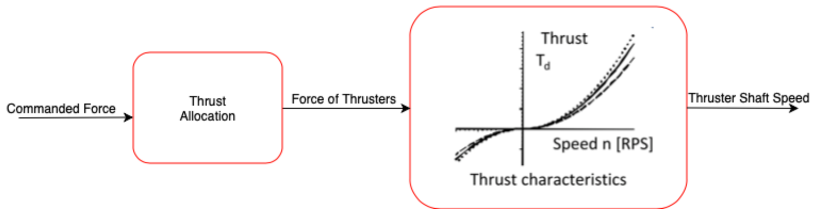


Figure 3.2: Relation between Commanded Force and Thrust Shaft Speed

The thrust allocation can be described as

$$\boldsymbol{\tau} = \mathbf{T}(\boldsymbol{\alpha})\mathbf{K}\mathbf{u} \tag{3.21}$$

where $\boldsymbol{\tau}$ is the commanded force vector, $\mathbf{T}$ represents the geometrical configurations of the thrusters which accounts for thrust direction, $\mathbf{K}$ is the thrust coefficient matrix translating shaft speed to force and $\mathbf{u}$ is the thrust shaft speed. Since the thrusters are fixed, the angle α is not variable, but instead constant and independent for each thruster. The geometrical configuration can be expressed as

$$\mathbf{T}(\boldsymbol{\alpha}) = \mathbf{T} = \begin{bmatrix} f(\boldsymbol{\beta}_{x,i}) & ... & f(\boldsymbol{\beta}_{x,n}) \\ f(\boldsymbol{\beta}_{y,i}) & ... & f(\boldsymbol{\beta}_{y,n}) \\ f(\boldsymbol{\beta}_{z,i}) & ... & f(\boldsymbol{\beta}_{z,i}) \\ f(\boldsymbol{\beta}_{x,i}, \mathbf{x}) & ... & f(\boldsymbol{\beta}_{x,n}, \mathbf{x}) \\ f(\boldsymbol{\beta}_{y,i}, \mathbf{y}) & ... & f(\boldsymbol{\beta}_{y,n}, \mathbf{y}) \\ f(\boldsymbol{\beta}_{z,i}, \mathbf{z}) & ... & f(\boldsymbol{\beta}_{z,n}, \mathbf{z}) \end{bmatrix} \tag{3.22}$$

where i denotes the seven thruster $n = 7$ and β denotes the direction of the thruster in regards to each axis xyz. $[x, y, z]$ denotes the arm from the CO to the thruster location resulting in $T \in \Re^{7x6}$, for 6 DOFs and 7 thrusters. The K matrix describes the thrust coefficients and is related to the block to the right in Figure 3.2. The matrix can be set as

$$\mathbf{K} = \begin{bmatrix} \mathbf{K}_1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \mathbf{K}_2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \mathbf{K}_3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \mathbf{K}_4 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \mathbf{K}_5 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \mathbf{K}_6 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \mathbf{K}_7 \end{bmatrix} \tag{3.23}$$

The u is the thurster speed input and is established as

$$\mathbf{u} = \begin{bmatrix} u_1 & u_2 & u_3 & u_4 & u_5 & u_6 & u_7 \end{bmatrix}^T \tag{3.24}$$

To translate the commanded force vector from the controller to how much thrust each of the actuators should produce one need to solve the thrust allocation problem. Thrusts are often restricted in multiple factors. The first is the max capability of the thrusters. The max force a thrusters is capable of producing is limited by the propeller size, the motors torque capability and how much the power there is available. Since ROVs tends to be tethered, the power capabilities is often the least governing restriction. The second restriction is the rate of which the power can be delivered, i.e the time it takes from commanded max shaft speed to the thruster is capable to deliver the set shaft speed. Due to a number of different configuration of the thrusters can solve a set commanded force vector, the thrust allocation problem turns into an optimization problem. The optimization problem can often be formulated as to minimize power consumption, restrained by the aforementioned restriction and still meeting the requirements from the control output.

3.2.4 Observer

The observer for the control system estimates the position and attitude of the ROV based on the available measurements and the control output from the controller.The observer in the control system of the ROV is implemented as a Kalman filter.[1] The Kalman Filter is a widely used estimator. Its based on the premise of that with full model knowledge, as well as knowledge of the measurement noise, it results in an optimal state estimator.

Given the state space model as

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{Ax} + \mathbf{Bu} + \mathbf{Ew}\\ \mathbf{y} &= \mathbf{Hx} + \mathbf{v}\end{aligned} \tag{3.25}$$

the discrete Kalman filter algorithm can be modeled as 3.26 to 3.28. Equation 3.26 initializes the state propagation variable and state estimation error co-variance matrix.

$$\begin{aligned}\bar{\mathbf{x}}(0) &= x_0\\ \mathbf{P}(0) &= P_0\end{aligned} \tag{3.26}$$

Equation 3.27 updates the state estimates, along with the error co-variance matrix and the Kalman-gain matrix $\mathbf{K}$, which is calculated as

$$\begin{aligned}\mathbf{K}(k) &= \bar{\mathbf{P}}(k)\mathbf{H}^T(k)[\mathbf{H}(k)\bar{\mathbf{P}}(k)\mathbf{H}(k)^T + \mathbf{R}]^{-1}\\ \hat{\mathbf{x}}(k) &= \bar{\mathbf{x}}(k) + [\mathbf{y}(k) - \mathbf{H}(k)\bar{\mathbf{k}}(k)]\\ \hat{\mathbf{P}}(k) &= [\mathbf{I} - \mathbf{K}(k)\mathbf{H}(k)]\bar{\mathbf{P}}(k)[\mathbf{I} - \mathbf{K}(k)\mathbf{H}(k)]^T + \mathbf{K}(k)\mathbf{R}(k)\mathbf{K}^T(k)\end{aligned} \tag{3.27}$$

3.28 propagates the filter values and updates the next iteration.

$$\begin{aligned}\bar{\mathbf{x}}(k+1) &= f(\hat{\mathbf{x}}(k), \mathbf{u}(k))\\ \bar{\mathbf{P}}(k+1) &= \mathbf{\Phi}(k)\hat{\mathbf{P}}\mathbf{\Phi}^T(k) + \mathbf{\Gamma}(k)\mathbf{Q}(k)\mathbf{\Gamma}^T(k)\end{aligned} \tag{3.28}$$

with $\mathbf{Q}(k)$ and $\mathbf{R}$ as the tuneable process co-variance and measurement co-variance matrices, designed for the set state space model.

3.2.5 Guidance

The task of the guidance block is to transform the desired path, trajectory or target to a controllable parameter to the system. The guidance module can be used to solve the path following problem. The guidance module is typically given a set of waypoints, or a path, and based on the set guidance law, ensures that the path is reached by letting the controller bring the state to the desired state. For marine applications, the guidance laws are heavily influenced of the actuator historically available to the ship, as well as the controllable states of the ship. For ships in general, in transit maneuvers at least, the available actuators are a fixed thruster for forward speed control and a rudder angle, for heading control. The guidance laws presented in [25] and [28] are largely based on this, along with under-actuated scenarios of missiles and other one-propulsion vehicles.

An alternative to a guidance module based on the actuators available is a reference model. A reference module is a function that creates a desired trajectory based on the reference position or velocity [1]. In the situation of over actuated vessels the use of a reference model can created the same desired performance as a pure guidance law, which is especially true for vehicles that is not prevented for translation or rotation in any DOF, such as a ROV.

The reference model constant jerk is based on the premise that the change in acceleration, i.e. jerk, is held constant or zero through out the translation. This results in a linear reference mode for the acceleration and a low order polynomial reference mode for both position and velocity. The results of keeping the jerk constant is shown in Figure 3.3.

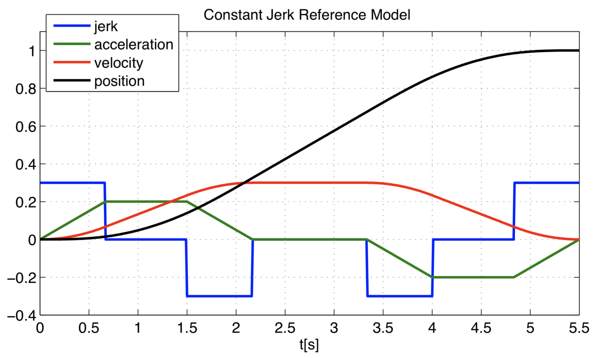


Figure 3.3: Constant jerk and resulting acceleration, velocity and position [1]

The constant jerk reference model can be described as

$$j = j_c\delta(t) \tag{3.29}$$

$$a(t) = \int_{t_0}^{t} a(\tau)d\tau + a_0 \tag{3.30}$$

$$v(t) = \int_{t_0}^{t} v(\tau)d\tau + v_0 \tag{3.31}$$

$$p(t) = \int_{t_0}^{t} p(\tau)d\tau + p_0 \tag{3.32}$$

where j is the jerk motion, δ is a step function and a, v and p is the acceleration, velocity and position, respectively.

Chapter 4

Image Recognition

As the identification of obstacles is here dependant on the use of sonar imaging, a critical part of the obstacle detection is the use image processing. The sonar data, when translated into sonar imaging represents the basis of this sonar processing. To understand the basis of the processing, a brief introduction to the digital image is presented.

4.1 Digital Image

A digital image can be seen as a array of data sets, where the size and dimensions are determined by a set of different factors. A digital image consists of a given number of pixels. One pixel represent one point of data, which can be further divided into bits. The number of bits for one pixel is determined by the size of information, or accuracy, in each pixel. The resolution of the image is determined by the number of pixels, and is distributed spatially in size of $N_w \cdot N_h$. Different types of images can be binary images, one bit per pixel, greyscale image, 8 bits per pixel, and color images, which contain 24 or 32 bits per image.

A binary image, is there for built up by a set of $N \cdot M$ number of pixels, determining its size, and one bit per pixel, determined if the pixel is black or white. The same applies for cray scale and color, but instead of one bit determining if its on or off, 0 or 1, there is a scale of 8 bits, or greyscale, giving 8^2 number of white and black combinations. For the color image in particular, it has 3 basis colors of RBG, or red blue and green, which when combined, can represent

$$2^8 \cdot 2^8 \cdot 2^8 = 16777216 \tag{4.1}$$

different colors. For the 32 bit example, there is a third set of bits, called the alpha channel, determining the transparency of the each pixel. Therefore, a digital image can be represented by a matrix of pixels, with dimensions of the matrix determined by the fidelity of the image, which is termed the spacial distribution of an image.

The colors of a digital image is based of color models. The color image can be constructed with several different color models, the RGB color model is the most common. The RGB model is based of that color can be viewed as a vector function of three coordinates for each position with in the image [29]. Another notation for this model can be the additive model, based on the fact that the image is obtained by adding the components of the primary colors, red, blue and green. The idea behind the color model, is that with the basis of three colors, and their intensity, matched together can reproduce close to all colors perceptible to humans.

Another commonly used system is the YUV system. It consists of three components, luminance (y) and two chrominance components (U and V). The YUV is obtained from the RGB model as [29]

$$Y = 0.299R + 0.587G + 0.114B \tag{4.2}$$

$$U = 0.564(B - Y) \tag{4.3}$$

$$V = 0.713(R - Y) \tag{4.4}$$

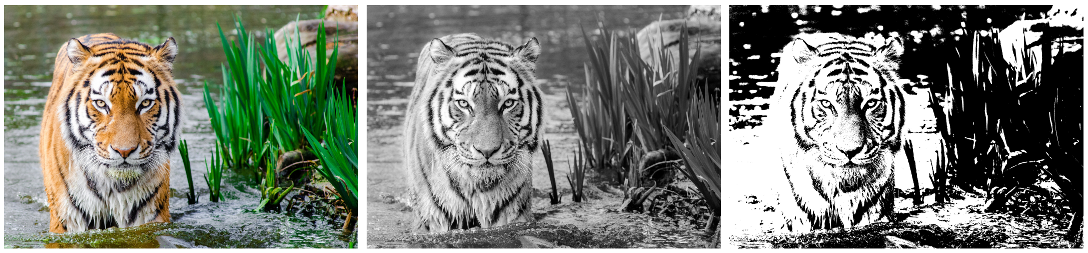


Figure 4.1: Image with color, greyscale and binary representation

Figure 4.1 shows the picture with 24 bit color representation. When transformed to binary and greyscale, the color nuances is lost and when transformed to binary representation, only black and white is left.

4.2 Image Processing

4.2.1 Image Greyscale

In the processing of images, it can be beneficial to transform a colored image into greyscale. This allows to narrow down the fidelity of the image and represent the image as a scale of grey instead of a scale and combination of colors and their intensities. The basis of greyscaling is to extract the dark and light colors of the image, and shade it with a scale of grey, as seen in Figure 4.2. A common usage of this is based on the standard Rec.601 which uses the luma or luminosity of an image to transform the color image to greyscale. If you have a color image based of the RGB color it is therefor necessary to transform the image to YUV color model, represented by equations 4.2 to 4.4. The luminance, Y is then the basis of the grey scale image, as it contains the information about the black and white levels of the images. The colors of the luminace is weighted in favor of green, as seen in the equation 4.2, the reason being that the green color is the color with largest space on the spectrum perceptible for humans and is the most occurring color in nature [30].

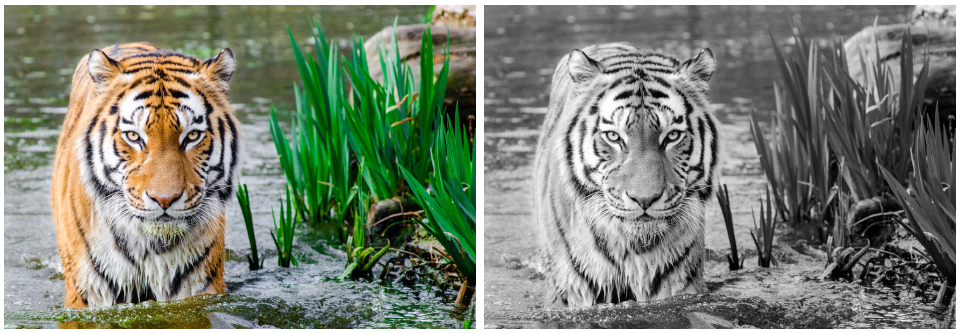


Figure 4.2: Color and greyscale

4.2.2 Image Threshold

A further form of image processing is that of image thresholding. In the image thresholding, one introduces a threshold value of the grey value in a picture. Consider a picture represented as a function $f(x, y$, where x and y denotes the pixels in the two dimensional data set. For each value of f, one can set a threshold value that determines if the pixel is a object point or background point. The result of the threshold, is reducing the image to a binary set of pixels, either black or white, that corresponds to their initial grey level.

$$g(x,y) = \begin{cases} 1 & \text{if } f(x,y) > T \\ 0 & \text{if } f(x,y) < T \end{cases} \tag{4.5}$$

The image threshold is a helpful tool when trying to identify objects that stands in contrast of the background[31]. To choose the threshold value, the use of an image histogram can be applied. The histogram shows a probability density function over the pixel intensities. Figure 4.3 shows the histogram of the greyscale picture of the tiger. The y axis represents the number of pixels, while the x-axis represents the grey level, thus the histogram shows the number of pixels per greyscale shade.

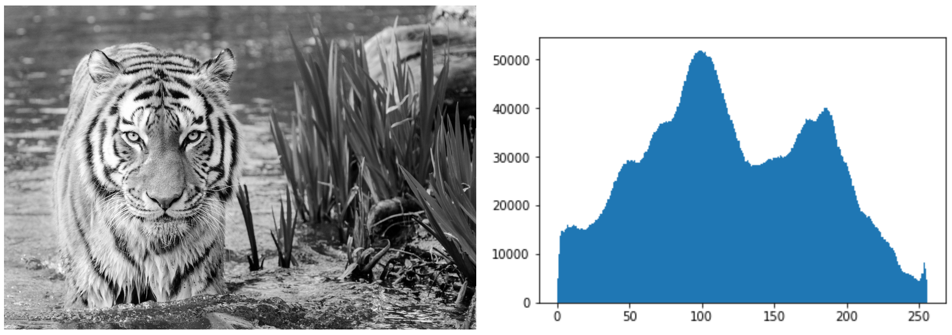


Figure 4.3: Greyscale image and corresponding histogram

Figure 4.4 shows the picture of the tiger of a global binary thresholding. Compared to the original greyscale, one can see that at a set greyscale level, the pixel is either black or white.

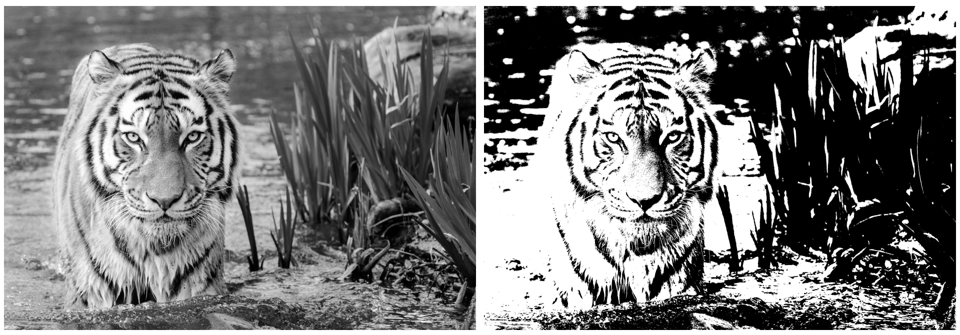


Figure 4.4: Greyscale image and binary image

In the case of the global thresholding, the threshold value is chosen arbitrarily. As there can be difficulties in choosing the right threshold value, several methods has been introduced to choose the right value. The Otus method is used to automatically choose the right threshold value. The Otus method is an algorithm that assumes that the image is defined by a background and an object in a bimodal picture. A bimodal picture suggests that there are two significant peaks in the histogram, showing that there are two dominant pixel intensity corresponding to the object and the background. The Otus method can be described as

$$\sigma_w^2 = q_1\sigma_1^2 + q_2\sigma_2^2 \tag{4.6}$$

where

$$\begin{array}{rr} \mathrm{q}_1(t) = \sum_{i=1}^{t} P(i) & \mathrm{q}_2(t) = \sum_{i=1}^{t} P(i) \\ \mu_1(t) = \sum_{i=1}^{t} \frac{iP(i)}{q_1(t)} & \mu_2(t) = \sum_{i=1}^{t} \frac{iP(i)}{q_2(t)} \\ \sigma_1^2(t) = \sum_{i=1}^{t} [i-\mu_1(t)]^2 \frac{P(i)}{q_1(t)} & \sigma_2^2(t) = \sum_{i=1}^{t} [i-\mu_2(t)]^2 \frac{P(i)}{q_2(t)} \end{array} \tag{4.7}$$

The Otus algorithm [32] tries to find a threshold value t which minimizes the weighted within-class variance in equation 4.6. Figure 4.5 shows the thresholding using the Otus algorithm.

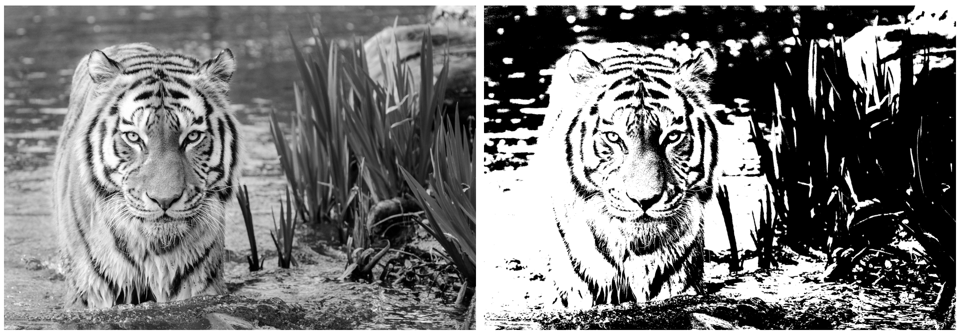


Figure 4.5: Greyscale compared to Otus Method threshold

Another thresholding technique is that of adaptive thresholding. The basis of the adaptive tresholding is if the image has difference levels of illumination for different parts of the image. The idea is then to do the tresholding locally, comparing each subregions pixels with another. This allows for success full thresholding for images whos histogram does not contain distinctive peaks, such as the tiger image, shown in Figure 4.3. Figure 4.6 shows the adaptive thresholding for the example tiger image. As one can see, the return image consists of more locally differences in black and white, giving the picture more details. One can also see that the image details the feature of both the tiger and the grass on its side. The method of selecting each subregion treshold value is based on the weighted sum of neighbourhood values where weights are a Gaussian window.

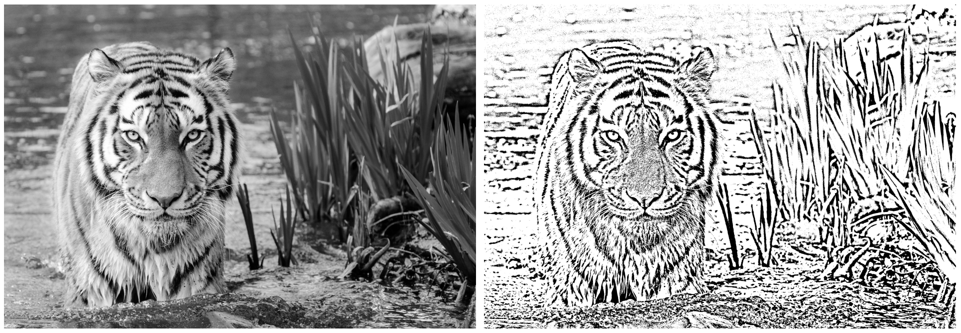


Figure 4.6: Greyscale compared to adaptive threshold

4.2.3 Image Smoothing

Image smoothing is also a relevant method in image processing. By performing a smoothing operation, one applies a lowpass filter kernel. In image processing, a kernel is a matrix of which one considers the pixels $[x, y]$ in an image. The application of a filter can be described as

$$g(x,y) = \omega \times f(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} \omega(s,t) f(x-s, y-t) \tag{4.8}$$

where $g(x,y)$ is the processed or filtered image, $f(x,y)$ is the input image, ω is the kernel. As seen from equation 4.8, each element of the kernel filter is considered by the set restrictions a and b.

Averageing

A type of filtering, or image smoothing is averaging. In this case the image is filtered by a box filter, described as 3×3 matrix on the form

$$\omega = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \tag{4.9}$$

which takes the average of all the pixels under the kernel area and replaces the central element, as shown in Figure 4.7. This is done by multiplying each element of the kernel with each corresponding element under the kernel of the original image, and then average by the total sum of the kernel elements.

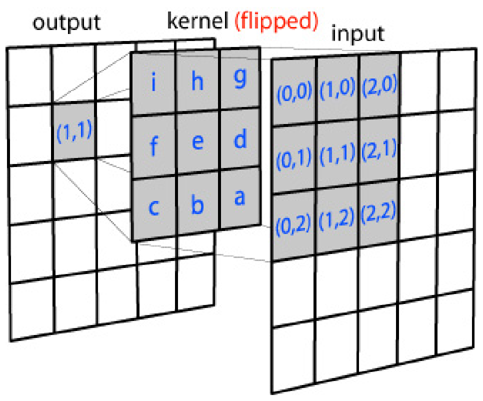


Figure 4.7: How the kernel is applied to a pixel

Each element of the kernel is multiplied with the corresponding pixels surrounding the target pixel, and then is averaged. As for the box filter containing elements of 1s, the target pixel is the average of the pixels surrounding, determined by the kernel size. Figure 4.8 shows a noisy image filtered by the box filter.

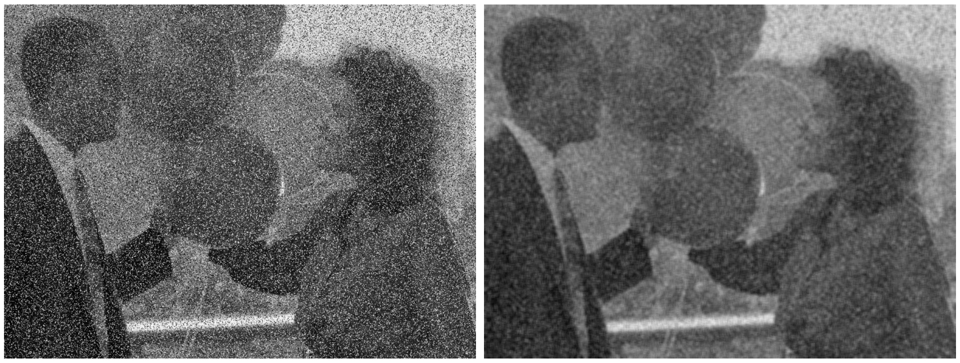


Figure 4.8: Averaging filter with original noisy image

Gaussian Blur

Gaussian blur or Gaussian filter is an other form of image smoothing. In stead of applying a box filter and then averaging each pixel, the kernel is made constructed by the use of a Gaussian distribution. In a 1-D Gaussian function can be described as

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \tag{4.10}$$

where x is the function variable and σ is the standard deviation. As the image is a 2D representation of pixels, the Gaussian function needs to represent two dimensions, and can be described as

$$G(x, y) = ce^{-\frac{x^2+y^2}{2\sigma^2}} \tag{4.11}$$

which is found by multiplying $G(x)$ with $G(y)$, and introducing a constant c as a scaling factor. The Gaussian kernel is then represented as the probability density function of the 2D Gaussian function on matrix form, where the expected value is the center value. Equation 4.12 shows a Gaussian kernel with $\sigma^2 = 2$ and a kernel size of 7×7 and c set to 90.

$$\mathbf{G}_{7\times7} = \begin{bmatrix} 1 & 3 & 7 & 9 & 7 & 3 & 1 \\ 3 & 12 & 26 & 33 & 26 & 12 & 3 \\ 7 & 26 & 55 & 70 & 55 & 26 & 7 \\ 9 & 33 & 70 & 90 & 70 & 33 & 9 \\ 7 & 26 & 55 & 70 & 55 & 26 & 7 \\ 3 & 12 & 26 & 33 & 26 & 12 & 3 \\ 1 & 3 & 7 & 9 & 7 & 3 & 1 \end{bmatrix} \tag{4.12}$$

The Gaussian filter is then applied to the pixels as described in Figure 4.7 and then normalized, resulting in a wieghtede average of the output pixel, weighted by the Gaussian distribution of the Gaussian kernel. Figure 4.9 shows the original greyscale image with a Gaussian filter applied.

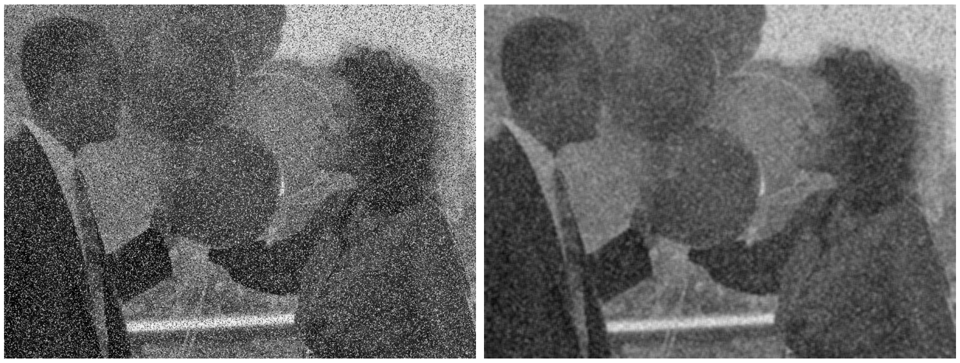


Figure 4.9: Gaussian filter with original noisy image

Median Filter

A third way of filtering is that of the median filter. The median filter Consider an image $I(x, y)$ represented as

$$I(x,y) = \begin{bmatrix} 1 & 2 & 3 \\ 9 & 2 & 7 \\ 10 & 0 & 3 \end{bmatrix} \tag{4.13}$$

To apply a median 3×3 filter to the central most pixel, represented by the number 2, the median of the matrix is calculated. The median of a data set I is the $I[n-1)/2]$ element of the data set when sorted after size. In the case of the data set in Equation 4.13, it can be desibed as

$$I(x,y) = \begin{bmatrix} 0 & 1 & 2 & 2 & 3 & 3 & 7 & 9 & 10 \end{bmatrix} \tag{4.14}$$

The $I[n-1)/2]$ would be 3, and the middle pixel would be changed to 3. This is done for all the pixels in an image.

Figure 4.10 shows that the median filter can smooth noisy pictures, while still maintaining structures and the boundaries between them. This is done because in a given kernel area the most occurring colors is that of the image objects and background, not of the noise. If there are more of the same number, there is a large probability of that number becoming the median value, and thus the output pixel. The backside of median filtering is higher computing time. When computing the median, the neighborhood values must be stored partially for the calculation of the median. However, in many image analysis task, the image enhancement is worth the computational cost [33].

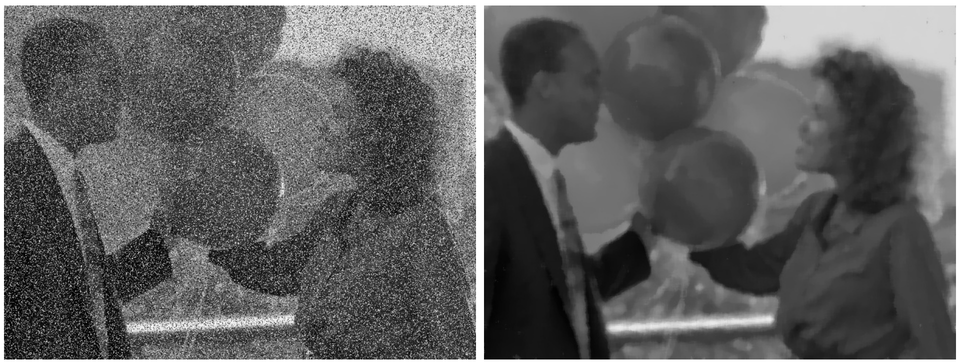


Figure 4.10: Median filter with original noisy image

4.3 Image Feature Detection

Image features can be determined as any object, structures or trends that occupy an image. The features can be edges, corners, circles, simple and complex patterns or things like faces. The act of extracting those features is that of finding the reoccurring trends of a specific feature one wants to detect. In this section, an introduction to some of the most popular image feature extraction methods and algorithms is done.

As features can be anything one self determines, a key part of the feature detection is to find what features one wants to detect. Features like edges and corners can both identify both human made objects in a nature picture, or can be helped in masking the object from the background of an image. This is done by defining an image descriptor.

The image descriptors is a kind of vector, used as the signature of a local image, i.e. a part of an image. The aim of this representation is to make the local image as distinctive as possible while maintain robustness to various kinds of scenarios.[7] As this thesis mainly utilizes feature detection, the focus will be aimed at the detection part of the algorithms.

4.3.1 Harris Corner Detection

The Harris Corner detection uses a corner descriptor in its feature extraction. A corner can be defined as a junction of two edges. The Harris corner detection is based of using a combination of partial derivatives, a Gaussian weighting function and the eigenvalues of a matrix representation of the equation.

The idea behind Harris Corner Detection is to analyse a specific part of the image, a window, move the window and look for a difference in the two snapshots of the image window. Consider the image in Figure 4.11. The image snapshot or window displays a clear difference when moved.

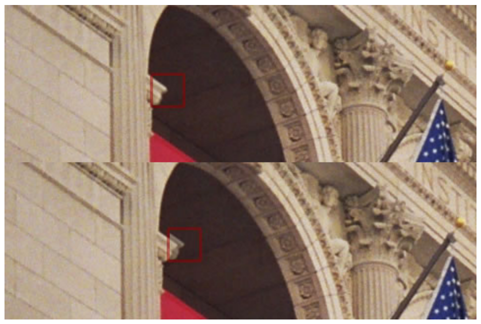


Figure 4.11: Window with variation of each snapshot[5]

The

$$E(u,v) = \sum_{x,y} [I(x+u, y+v) - I(x,y)]^2 \tag{4.15}$$

where E is the difference between the two windows, (u,v) is the movement of the image, $w(x,y)$ is the window at the position (x,y), I is the intensity of the image and the moved image. To detect the corners, the goal is to find positions where the difference is large, i.e. the value of E is high. The goal is therefor to maximize E. By expanding equation 4.15 in Taylor series yields

$$E(u,v) = \sum_{x,y} u^2 I_x^2 + 2uvI_xI_y + v^2 I_y^2 \tag{4.16}$$

where I_x and I_y is the partial derivative of the intensity function $I(x,y)$. Writing on matrix form gives

$$E(u,v) = \begin{bmatrix} u & v \end{bmatrix} \sum_{x,y} \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} \tag{4.17}$$

Replacing the sum of the matrix with M as

$$\mathbf{M} = \sum_{x,y} \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} \tag{4.18}$$

gives the notation

$$E(u,v) = \begin{bmatrix} u & v \end{bmatrix} \mathbf{M} \begin{bmatrix} u \\ v \end{bmatrix} \tag{4.19}$$

By elvauating the

$$R = det(\mathbf{M}) - k(trace((\mathbf{M}))^2$$

where det($\mathbf{M}$) = $\lambda_1\lambda_2$ and trace($\mathbf{M}$) = $\lambda_1 + \lambda_2$. The score of R determines if there is a corner or not, set if R supersedes a set value. To determine if there is a corner or not, one can evaluate the eigenvalues. Figure 4.12 shows the permissible area for a corner.

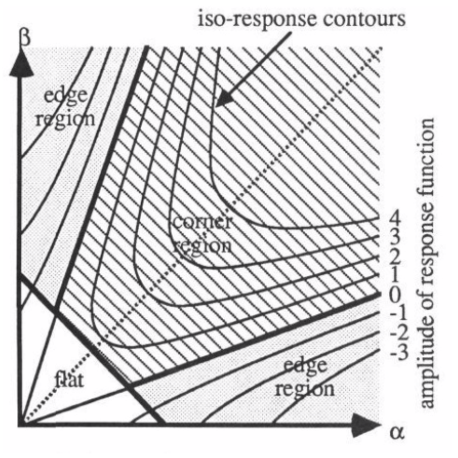


Figure 4.12: The scale of evaluating corners, plotted for the two eigenvalues[6]

As seen from the figure, if both the two eigenvalues are small, then the pixel is flat. If one eigenvalue is large while the other is not, there is a edge region. If both eigenvalues are large, then there is detected a corner.

The Harris Corner detection is rotate invariant, and can detect corners independently of rotation. It is however sensitive to the scaling of the image, as corner does not necessarily contain their features if up scaled or down scaled to a great extent.

4.3.2 Scale Invariant Feature Transform

SIFT, or Scale Invariant Feature Transform is a local image detection and descriptor used in various computer vision tasks such as classification, image retrieval, image registration and pose estimation [7]. SIFT utilizes a scale space extrema detection. SIFT is a scale invariant method while it also maintains the ability to detect features independently of rotation, blur illumination and noise. SIFT operates on a Gaussian scale space in order to detect keypoints and construct their local descriptors. A Gaussian scale space can be described as

$$L(x, y, \sigma) = I(x, y)G(\sigma) \tag{4.20}$$

where I is the image and G is a Gaussian filter with the standard variance of σ. The SIFT algorithm takes the extreme in DoG (Difference of Gaussian) scale space as the initial keypoints. The DoG can be defined as

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) \tag{4.21}$$

which is the difference two picture scale spaces, differentiated by a scale factor k. By applying the Gaussian filter a set number of times, a set of keypoints can be retrieved by the DoG.Images in the scale space is then transformed in to octaves, one octave for each for one point on the scale. In each octave its further divided into layers s. SIFT searches for the extrema along three dimensions for keypoint detection, which results that $s+3$ DoG images to cover a complete octave. In addition,

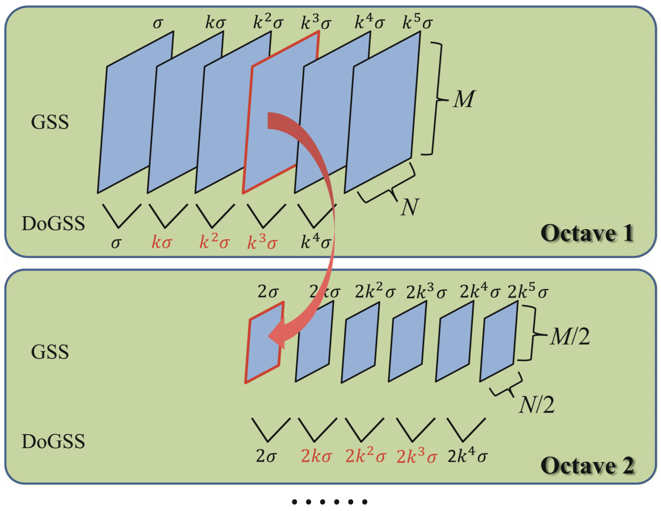


Figure 4.13: DOG scale space (DoGSS) is generated by subtracting adjacent images in the Gaussian scale space (GSS). The red values indicate the scales that is used for keypoint detection[7]

Keypoint Detection

To perform the keypoint detection SIFT utilizes the difference in the scale space between two images shown in 4.21. It finds the local maxima and minima of $D(x, y, \sigma)$ by comparing each pixel to the 8 surrounding pixels and the 18 pixels in the layer above and below. To accurately find a keypoint defined as $X = X(x, y, \sigma)$, a fitting to a 3D quadratic function around the local area of X_0 is done, and the keypoint is found by taking the interpolated position of its extremum. By introducing $X_0 = (x, y, \sigma)$ in 4.21 and using Taylor expansion one obtains

$$D(X) = D(X_0) + X^T \frac{\partial D}{\partial D} + \frac{1}{2} X^T \frac{\partial^2 D}{\partial^2 X} X \tag{4.22}$$

Further, one finds the extrema of $D(X)$ by finding the derivative of 4.22 and setting it equal to zero, which yields

$$\frac{\partial^2 D}{\partial X^2}(X_0)\Delta X = -\frac{\partial D}{\partial X}(X_0) \tag{4.23}$$

where ΔX is the offset of the extreme point to the original point X_0, so the refined keypoint is localized at $X = X_0 + \Delta X$. If the offset is larger than 0.5 in any of the dimensions it implies that extremum corresponds to the nearby point. If so, the extremum is changed to the neighbor point. This is repeated to until all the dimensions in the offset are found to be less than 0.5. By introducing $X = X_0 + \Delta X$ to 4.22, one obtains

$$D(X) = D(X_0) + \frac{1}{2}^T \frac{\partial D}{\partial X}(X_0) \tag{4.24}$$

To further narrow down the keypoints, a filtering is done. By evaluating the extremum $D(X)$ and finding them smaller than 0.03, it is discarded due to its low contrast. As the DoG function returns strong response along edges, extremas found with low responses usually have a large localization error and is unstable due to noise.

The curvature of the keypoint can be also be calculated to find the points on the edge. The keypoints that does not have extreme response in the DoG scale has a main curvature along an edge and a smaller perpendicular to that. The two curvatures are proportional to eigenvalue of a Hessian matrix H [7], which is found as

$$H = \begin{bmatrix} D_{XX} & D_{XY} \\ D_{YX} & D_{YY} \end{bmatrix} \tag{4.25}$$

where D_{ii} is the partial derivatives which are are found by taking the difference of neighboring sample points in the scale space. As the property we are interested in, the eigenvalues are not needed to be calculated, only the ratio between them. If one assumes that both eigenvalues are positive and they are defined as $\lambda_1 = \lambda$ and $\lambda_2 = r\lambda$ where $r \geq 1$, it follows that

$$trace(H) = \lambda + r\lambda = (1+r)\lambda$$
$$det(H) = \lambda \times r\lambda = r\lambda^2$$

With $r \geq$, one can obtain a ratio between the trace and the determinant of the matrix H, found as

$$\frac{trace(H)}{det(H)} = \frac{(r+1)^2}{r} \tag{4.26}$$

This results in the ability to set a threshold on the ratio to discard the points with large r, which would correspond to a large principal curvature and small corresponding perpendicular. A typical value for the implementation of SIFT is to set the threshold ratio to 10. As opposed to the Harris Corner detection, the usage of the eigenvalue ratio instead of the evaluation of the values of the eigenvalues, reduces computational cost.

4.3.3 Speeded up Robust Features

Speeded Up Robust Feature is based on the work on SIFT, but were introduced as a way to perform the algorithm faster by utilizing integral images. The algorithm structure is largely the same with using a scale space representation, and detect interest points in the scale space. To detect interest points, SURF uses Hessian Blob detector.

An integral image is defined as the sum from the start of the input image $(x = 0, y = 0)$ to the the location of a the integral image entry (x, y). Mathematically it can be expressed as

$$I_{\sum}(x, y) = \sum_{u=0}^{x} \sum_{v=0}^{y} I(u, v) \tag{4.27}$$

where $I(x, y)$ is the input image. The advantage with using integral images lies in the fast computation of the sum of the intensities for any rectangular region, as it only requires three additions to obtain its summed intensity, regardless of the region size [7].

As with SIFT, SURF utilizes the scale space. The difference in strategy comes as instead of reducing the image size step by step, the filter is repeatedly increased. Besides the scale normalized Laplacian of Gaussian LoG retrieved and approximated from the DoG, SURF also utilizes the Hessian matrix and the corresponding determinant. The Hessian is a point x in an image I at the scale σ is defined as

$$H(x, \sigma) = \begin{bmatrix} L_{xx}(x, \sigma) & L_{xy}(x, \sigma) \\ L_{yx}(x, \sigma) & L_{yy}(x, \sigma) \end{bmatrix} \tag{4.28}$$

where $L_{xx}(x, \sigma)$ is the convolution of the second order Gaussian derivative and σ is the standard deviation. The SURF algorithm is based on the integral image, which can be used to construct an estimated Hessian matrix $\hat{H}(x, \sigma)$. The SURF uses multiple box filters to approximate the Gaussian derivatives. Since the convolution to of the box filters to image can be computed with only several additions based on the integral image, the approximated Hessian matrix can be calculated extremely fast and is defined as

$$\hat{H}(x, \sigma) = \begin{bmatrix} D_{xx}(x, \sigma) & D_{xy}(x, \sigma) \\ D_{yx}(x, \sigma) & D_{yy}(x, \sigma) \end{bmatrix} \tag{4.29}$$

where $D_{ii}(x, \sigma)$ are the convolutions of the approximated filters with image I. The determinate associated with 4.29 can be review as the response of a blob point, a point that contains pixels sharply differing with the surrounding in lumination or color, which is approximated as

$$R(x, \sigma) \approx D_{xx}D_{yy} - 0.9D_{xy}^2 \tag{4.30}$$

Lastly, in order to localize interest points in the image and over scales, a non maximum suppression in a $3 \times 3 \times 3$ neighborhood is applied.

4.3.4 Oriented Brief Rotated Fast

Oriented Brief Rotated Fast is an alternative to the SIFT and SURF algorithms, that utilizes the FAST detector and the BRIEF descriptor. ORB performs as well as SIFT on the task of feature detection while being almost two orders of magnitude faster [34].

FAST

The FAST, or Features from Accelerated and Segments Test, detector is a corner detector. It is based on reviewing a pixel p in an image I with an intensity I_p. A threshold value is chosen as t, and a circle of 16 pixels around the pixel of interest p, as shown in Figure 4.14.

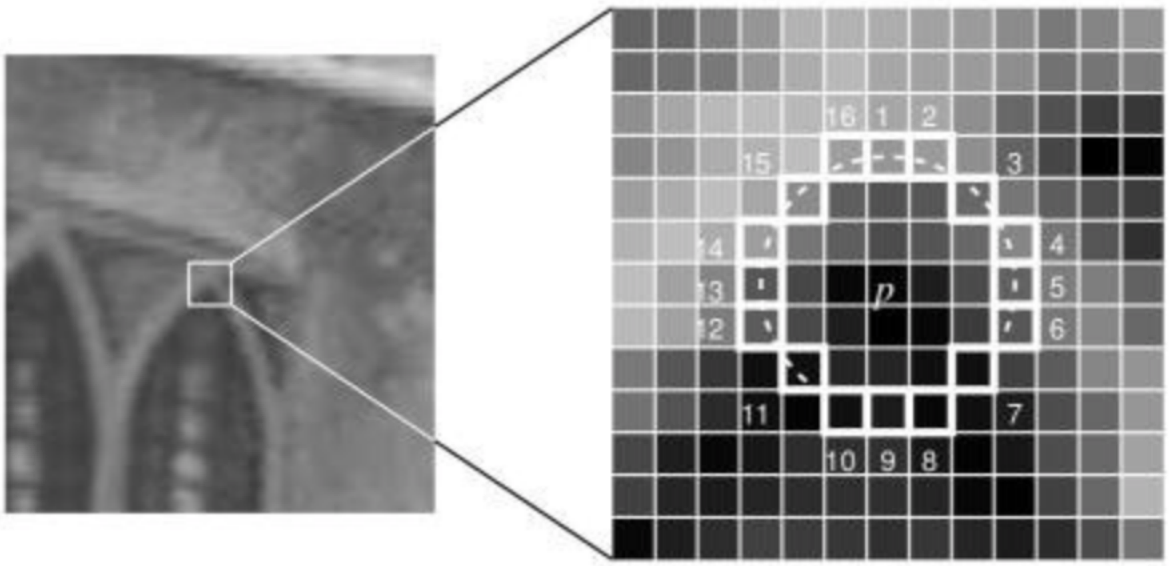


Figure 4.14: The selection of pixels around the pixel of interest

The pixel is chosen as a corner if there exist a set of pixels n which are all brighter than the original pixel if

$$I_{pi} > I_p + t \tag{4.31}$$

where I_{pi} is the pixel intensity of one of the surrounding 16 pixels. In the original publication detailing the algorithm, set the value of $n = 12$ as the number of pixels needed to comply with 4.31 to determine if the pixel represents a corner or not. The limitations to the algorithm is in the selection of the threshold value of n. With 16 reviewed pixels and the threshold value of $n = 12$, the number of detected keypoints becomes very large, as it is sensitive.

Chapter 5

Path Planning

The path planning problem is one of the crucial aspects of automatic and autonomous application in systems capable of change their position. The path planning problem can be defined as the action of defining the positions between the start point and end point for a change in spatial state. The path planning implies in its word that the path is predefined before the change in position happens. However, the term path re-planning can described continuous path planning through the mission. The definition of Tsourdos (2011)[35] for path planning for Unmanned Aerial Vehicles (UAV) is to provide structured mobility. To further explore the path planning problem, the definition of path needs to be stated.

A path can be described as a set of positions between a given start point and an end point. The path only describes the vessels positions, and unlike a trajectory, a path is not constricted by time or velocity requirements. The path can be described by a several different data points. A path can consist of a continuous set of points described, often denoted by a function. The most common way to describe a path is through waypoints, and defining what connects said waypoints. Connecting the waypoints can be done by straight lines, curves, or chosen functions. Depending on the requirements for the path, and restriction of the vessel, the waypoints and their implication of method of connection becomes the data set of the path. By defining straight lines as the connection, the path becomes piece-wise linear paths segments, as shown in Figure 5.1.

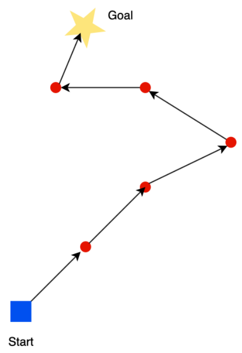


Figure 5.1: Piece-wise linear path

The essence of the path planning problem is to find a possible path between two points, whit out breaking the maneuvering restrictions of the vessel or defining the path unfeasible in regards of external factors such as objects and defined risk factors. The planning of the path can be described by the following mathematical expression [35]

$$P_{start} \xrightarrow{r(q)} P_{end} \tag{5.1}$$

where P_{start} is the starting position and P_{end} is the end position. The path is then described by the function r(q). As the start and finish points are often defined, the problem becomes defining the function which describes the path. By starting with the premise of that the path problem can be solved by defining a set of waypoints, the path problem can be formulated as

$$P_{start} = \xrightarrow{r(q)_i} \dots \xrightarrow{r(q)_{n+1}} P_{end} \tag{5.2}$$

where the number of waypoints n divides the problem in 5.1 into $n+1$ smaller problems of defining paths between n waypoints by finding $n+1$ number of paths $r(q)$. By increasing the number of functions describing more paths, the problem can easily become more difficult for the the path planning problem, but by assuming that each path is described the same way, the complication is reduced. The simplest form is to set each path segment as a linear path described by a straight line. Thus the path problem can be formulated as

$$R(q) = \sum_{i=1}^{n+1} r(q)_i \tag{5.3}$$

where each $R(q)$ denotes the total path, and the $r(q)$ is the path segments all described by lines, resulting in a piece wise linear path defined by the n waypoints.

By the predefined path segments, the path planning is now reduced to a waypoint generation problem, where the objective is not to define what position of the path, but defining the path from a set of waypoints. The number of waypoints affects both the problem complexity, and the fidelity of the path. By letting the number of waypoints go to do infinity, the same the path problem reoccurs, and by letting the number of waypoints go to zero, reduces the problem solution to a rudimentary solution of one line segment describing the entire path. Although, the line segment path by obvious geometrical features describes the shortest path, the feasibility in both regards to the vessels motion restriction and lack of obstacle consideration yields often none feasible solutions. The first problem is thus to find a suitable balance of number of waypoints, between infinity and zero, while still satisfying the environmental and system constrains.

The second problem is finding the waypoint positions. Since the path is defined through the waypoints, the position of the path relies on this, and the path problem reoccurs. To be able to qualitative assess the a waypoint position, one needs to be able to quantify what represents a good or optimal waypoint position, and thus what describes a good or optimal path.

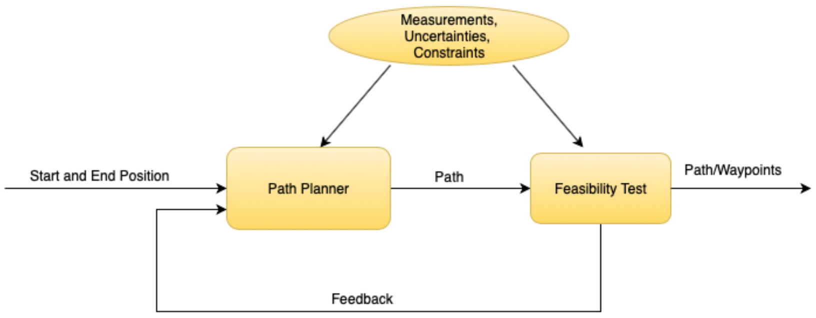


Figure 5.2: Path Planning Algorithm

As the energy and time consumption often is defined as the cost of an operation, which again often directly correlates with the monetary cost, in general, the path problem is based of finding the shortest possible path, taking the least amount of time. For local path planning problems however, several more pressing restrictions can be present. For the path to be considered better than an other, risk and complexity is often considered the more valuable parameters to reduce [12]. However, the purpose of the path planning is to find a admissible path between two points and the length of the path is still a valuable parameter.

An other concern when proposing a possible path is if the vessel can follow the path or not. If the proposed path is acceptable, there is still no guarantee that the path produced by the vessel corresponds perfectly with the proposed path, overshooting or not being able to reach the desired path through the movement. If this occurs, the movement of the vessel is not guaranteed the same risk, time, and length constrictions the path originally contained. This could result in higher risk, longer paths and potentially collision, due to the error between the path and the movement. An important constrain on the path proposal is therefor the path integrity, and the guarantee that, with the on board actuators and control system, that the vessel is able to keep the movement to the path. Places where this is likely to occur is in cases of rapid shifts in direction, and especially changes in direction where the vessel potentially is under actuated or geometrically restricted. Examples of this can be hard turns for ships without thrusters, where it relies on forward speed to have activate the rudder as an actuator.

There are several different approaches to the generation of waypoints. Different algorithms include the Growing Neural Gas Algorithm (GNG)[36] and waypoint generation using reinforcement learning [37]. The next section discusses the use of Voronoi Diagrams in waypoint generation and path planning.

5.1 Voronoi Based Path Planning

The Voronoi based path planning is laid on the the premise that there is places in the spatial area that or not available for motion. For practical application this would imply areas where there are obstacles. The Voronoi diagram shows the map as a grid of areas, where each area corresponds to one obstacle. The intersection between said areas are defined as the

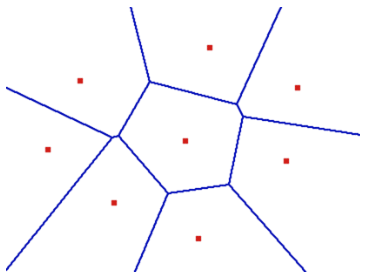


Figure 5.3: Voronoi Diagram [8]

The Voronoi diagram is a geometrical division of an Euclidean plane, limited to a set area based on objects or points in the diagram. Voronoi diagrams can be used in several applications, such as city planning, computer graphics, epidemiology, geophysics and meteorology. In mathematical terms, the Voronoi diagram splits a finite dimensional Euclidean space into set of convex polygons such that the polygons contains one and only one site from a set of n sites P that occupies the Euclidean Space. The result is a space with n cells, called Voronoi Cells.

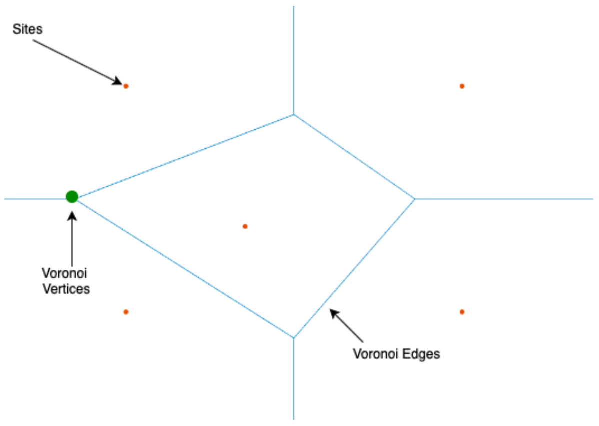


Figure 5.4: Voronoi Diagram with objects

The division of the space into the Voronoi Cells is based on the euclidean distance expressed as

$$d(q,p) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2 + ... + (q_n - p_n)^2} \tag{5.4}$$

such that for a an arbitrary point in a Voronoi cell based of a point p_i, the Euclidean distance $d(q,p_i) < d(q,p_j)$, for every point p_j in P [38]. The result in of this is a set of Voronoi edges, which defines the convex polygons resulting in the Voronoi cell. The Voronoi edges, where they meet, defines the Voronoi vertexes, as seen in Figure 5.4. Because of the distance between every point in the Voronoi Cell and the site is smaller than the distance to every other site, the edges shared by two sites, is positioned at the half way of the distance between two points, as illustrated in Figure 5.5

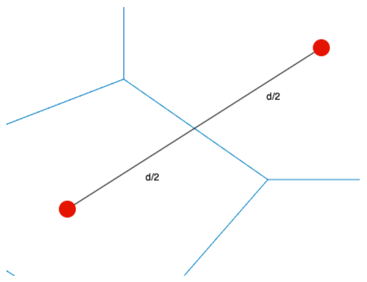


Figure 5.5: Distance between edge and site

The use of Voronoi diagrams in path planning is based on the premise that the vertices at all times, have the maximum distance from the obstacle present in the area. To transform the Voronoi diagram to a path the vertices are chosen as waypoints, and the edges connecting the vertices are the piece wise linear path segments. When complete, the Voronoi diagram consists of a set of paths, that guarantees longest distance from the obstacles as possible. However, by inspection it easy to see that there are paths that have longer distances from obstacles than other, and the length of each path is also varying.

5.2 Dijkstra

The Voronoi diagram is based on having location unfeasible to pass, or in other words location where there are physical obstacles. To chose the right path, a set of cost criteria needs to be introduced. For the Voronoi diagram, the most obvious is the path length combined with the distance to each obstacle along the path. The cost function needs to account for the possibility that although, mathematically the path is feasible, thresholds for the acceptable path needs to be set, based on the criteria. Examples of this could be the distance to the obstacle, the turning rate for the path and the path lengths.

The Dijkstra Algorithm is often used for the latter. It calculates the shortest path between to start and goal points, given a set of nodes and the length between each node. The lengths are set as the cost of each translation between each node. The Dijkstra algorithm were introduced in 1956 by the computer scientist E.W. Dijkstra. The algorithm, given an input vertex or point, calculates the distance to each other vertex in a graph tree. By doing so, the algorithm then chooses the shortest path to a indicated goal vertex.

Chapter 6

Method and Development

The basis of this thesis is to create a path planning scheme based on sonar imaging data from a forward looking multibeam sonar. The input is thus a sonar video stream. The sonar imaging data is that of Blueprint Subsea M1200d Dual-Frequency Multibeam Sonar, obtained at 1.2Mhz Low Frequency mode. The data can be found at `https://www.blueprintsubsea.com/pages/product.php?PN=BP01042`. The premise for problem faced are a path planning problem with an a-priory knowledge of the final goal point. The path planning is restricted to the scanned area of the sonar image data, as information of the are outside the scanned area is not available. The path planning is conducted using the information from the sonar imaging data and combining that with Voronoi diagrams and the Dijkstra algorithm. The obstacles are discovered using the sonar image, the Voronoi diagram creates a web of feasible path segments, and the Dijkstra algorithm calculates the shortest available path through the Voronoi Diagram.

To extract the obstacle knowledge from the sonar data, the use of the OpenCV library is important. To access the OpenCV library, Python code is used. Further, the obstacle information is passed through a TCP socket that connects the Python Program with the control system which is developed in LabVIEW, for the simulation of the paths. In this section a review of the different programs and programming language used will be conducted, along with the presentation of the methods used and the development of the python code and the LabVIEW environment.

6.1 Software

The software utilized to conduct the experiments for this thesis is based on the python programming language and the LabVIEW graphical programming environment. Python is a open source high level programming language widely used in the scientific community for its applicability and widely supported library and directories. To access the python programming suit, the package and compiler solutions developed by Anaconda were used. Anaconda is a package manager and development environment for Python.

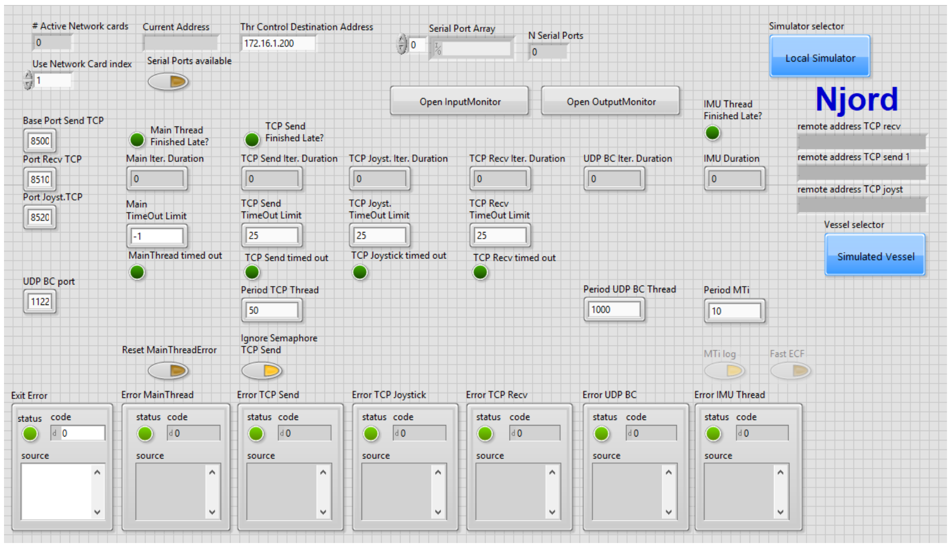
There are two Python versions supported at the moment, namely Python 2 and Python 3. The code written for this thesis are developed in Python 3. The Anaconda suit offers several integrated development environment, where the combined compiler and editor Spyder were utilized. Different packages and libraries were also utilized, of which the most significant were OpenCV.

OpenCV, is a computer vision library with a open source code. OpenCV was originally developed by Intel, but has since become open source. OpenCV has a broad variety of built in function in its library, many of which are utilized in this thesis. The notable ones are the different thresholding, filtering, image handling and image feature extraction algorithms.

To simulate the ROV dynamics, and to test the feasibility of the paths generated, LabVIEW were used. LabVIEW is a graphical programming

The The Applied Underwater Robotics (AUR) lab at NTNU has through different projects and theses developed a full suite of controller, simulator and graphical interface, mainly to use for software testing and sea trials of the ROVs. The main contributor is Fredrik Dukan, and his contribution is stated in his Ph.D. Thesis [1]. The software suit built in LabVIEW consists of three main parts, the simulator Verdandi, which simulates the ROV dynamics, the control system Njord and the Graphical User Interface (GUI) called Frigg.

Njord is the controller built for the ROV. The controller can be used both for simulations as well as controlling the ROV during sea trails.



# Active Network cards	0
Current Address	
Thr Control Destination Address	172.16.1.200
Serial Port Array	0
N Serial Ports	0
Use Network Card index	1
Simulator selector	Local Simulator
Vessel selector	Simulated Vessel
Base Port Send TCP	8500
Port Recv TCP	8510
Port Joyst.TCP	8520
UDP BC port	1122
Main Iter. Duration	0
TCP Send Iter. Duration	0
TCP Joyst. Iter. Duration	0
TCP Recv Iter. Duration	0
UDP BC Iter. Duration	0
IMU Duration	0
Main TimeOut Limit	-1
TCP Send TimeOut Limit	25
TCP Joyst. TimeOut Limit	25
TCP Recv TimeOut Limit	25
Period TCP Thread	50
Period UDP BC Thread	1000
Period MTi	10
Ignore Semaphore TCP Send	
remote address TCP recv	
remote address TCP send 1	
remote address TCP joyst	

	status	code	source
Exit Error	●	0	
Error MainThread	●	0	
Error TCP Send	●	0	
Error TCP Joystick	●	0	
Error TCP Recv	●	0	
Error UDP BC	●	0	
Error IMU Thread	●	0	

Figure 6.1: The controller Njord

Frigg is the GUI for the user interface of the controller. Figure 6.2 shows the Frigg GUI, with options for what motion control to choose such as Manual, tracking or Dynamic Position (DP).

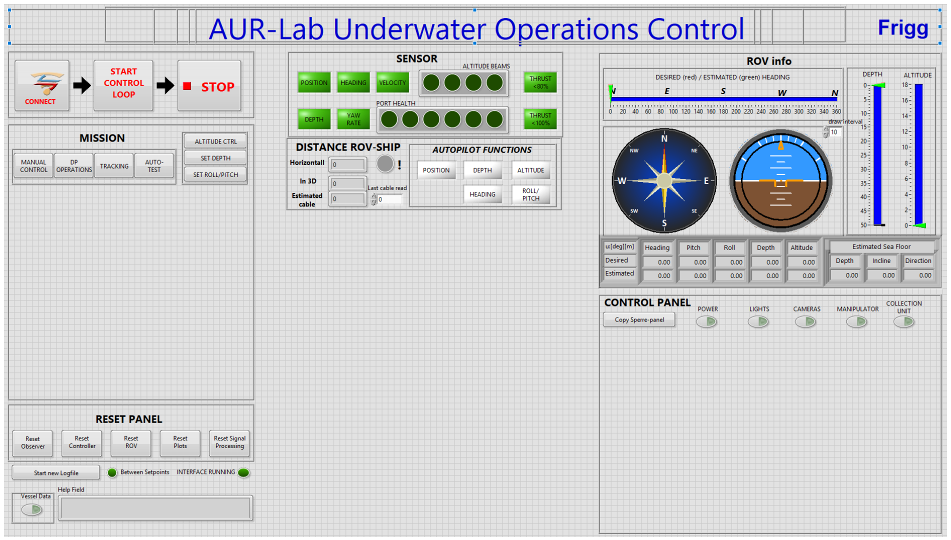
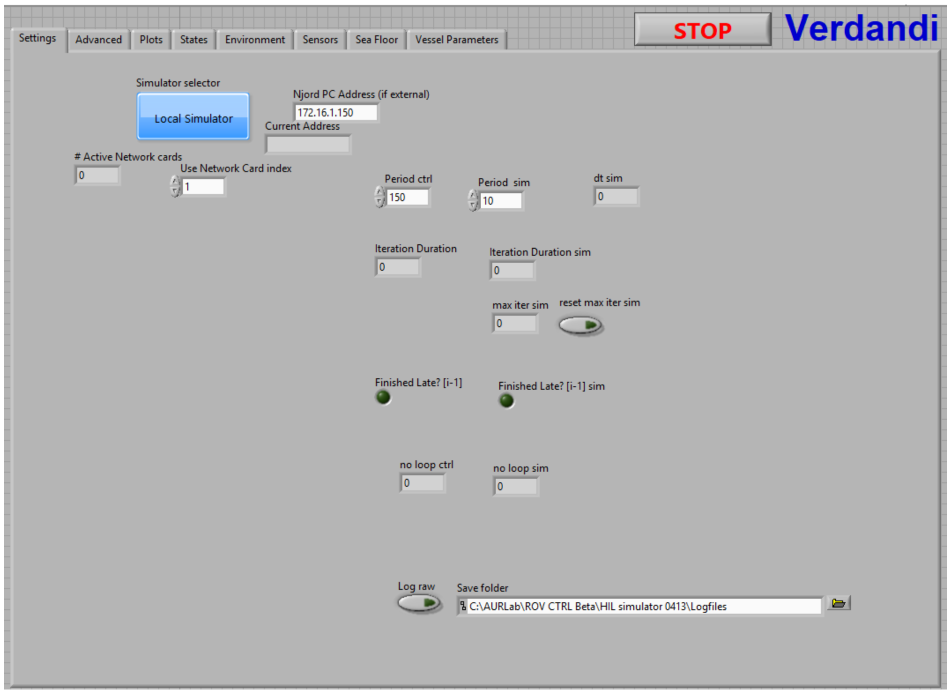


Figure 6.2: Graphical User Interface for the control system

The simulator built in the LabVIEW is called Verdandi. Verdandi contains all the ROV dynamics and is used for HIL-testing as well as simulator of the dynamics for offline simulations. The Verdandi interface can be seen in Figure 6.3



Simulator selector: Local Simulator
Njord PC Address (if external): 172.16.1.150
Current Address:
Active Network cards: 0
Use Network Card index: 1
Period ctrl: 150
Period sim: 10
dt sim: 0
Iteration Duration: 0
Iteration Duration sim: 0
max iter sim: 0
no loop ctrl: 0
no loop sim: 0
Save folder: C:\AURLab\ROV CTRL Beta\HIL simulator 0413\Logfiles

Figure 6.3: The simulator Verdandi

6.2 Method

6.2.1 Image Processing

To perform the feature detection on the sonar image input, several steps were performed. The first one is the extraction of image from the sonar video stream. This were done by a simple screen grab from the sonar image stream. The sonar image stream can be seen in Figure 6.4.

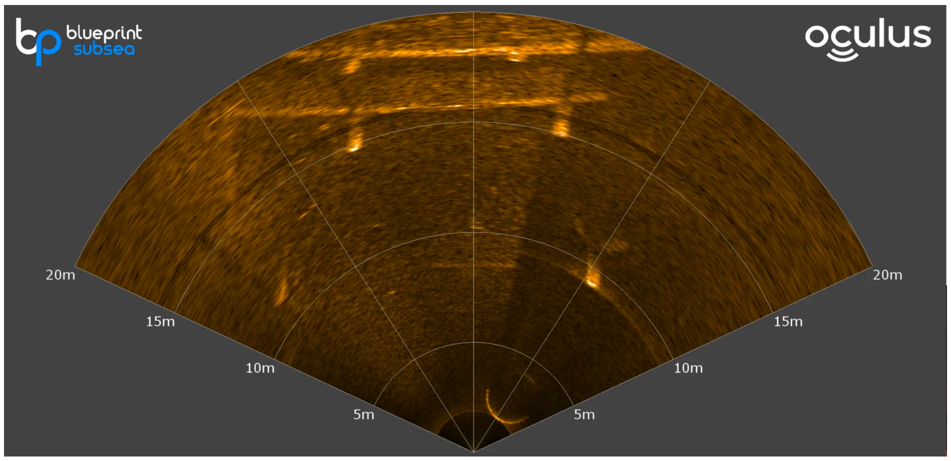


Figure 6.4: Sonar Video Stream

As the multi beam sonar scans a sector of the front of the ROV, the image needs to be cropped to only contain information from the sonar scan, and no the filler around. This were done by setting a set of variables to choose from which pixels in the original image to extract, resulting in Figure 6.5

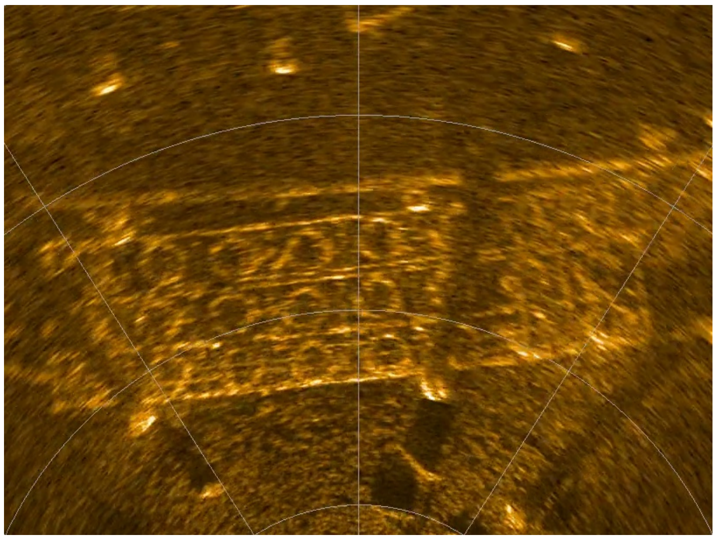


Figure 6.5: Sonar Image Cropped

The cropped sonar image now represents information solely on the scanned sector. The backside of this is the loss of information from the sector which does not fit with the squared frame.

Further, the image needed to be processed for feature detection. Several processing techniques were performed, to ensure the best possible results. This included filtering and thresholding. To ensure performance, several filtering and thresholding techniques were tested on sample images. The results from this were analyzed and the methods yielding the best results were chosen. The results of this is detailed in Chapter 7.

This leaves the image in a black and white representation, were the obstacles were represented by white spots, while the background were black. To find the location of the features, several detectors were tested, which returns keypoints in a specific pixel position in the image. The final result of this are presented in Chapter 7.

6.2.2 Path planning

Obstacle Map Generation

To transform the pixels in the image to a map, a conversion were needed. This were done by using the distance indicator on the image and using the number of pixels between the distance indicators to create a 2 dimensional map. The map were based on the image, with origin in the bottom left corner of the image, with ROV position in the middle. The map size were found to have a width of $W = 9.24$ meters and a height of $H = 14.1$ meters. As the cropped image no longer contains the origin of the sonar signals, an adoption were performed. The cropped image were chosen to align with the points from the original feed were the ROV scans 10 meters ahead. The angle of the sonar scanned sector were measured at 135°, which results in that the two bottom corners of the cropped image, represents the environment from

$$10 \cdot cos((180 - 135)/2) = 10 \cdot cos(22.5) \approx 3.83 \tag{6.1}$$

meters ahead of the ROV to the end of the map. The map width is determined from the number of pixels and the pixel to meters conversion ratio found and is approximately $W = 9.24$ This in turn means that the ROV start position is placed roughly in $(x, y) = [9.24, -3.84]$. The path were based on the generation of Voronoi Diagrams. Voronoi diagrams creates a web of feasible path segments around a set of input obstacles, and is there for reliant on a set of points of obstacles as input. To achieve this, a filter were applied to the detected keypoints from the detector. The filter were created as a grid with a set dimension, turning the map into a The filter set a minimal number of keypoints needed to consider the grid as containing an obstacle or not. The obstacle were placed in the middle of the grid.

Voronoi Diagram and Dijkstra Algorithm

The obstacles now present in the map were used as a input for the Voronoi generation. The Voronoi algorithm returns a set of path segments, and a set of points or vertices describing the joints between the Voronoi map. The Voronoi algorithm were retrieved from github.com [39], originally created by user *Yatoom*. The algorithm inputs are a set of points and a dimension for the diagram size, inside which the Voronoi diagram is created. The frame size were set to the total size of the scanned image, i.e the height and with of the original obstacle map. To implement the algorithm with the developed code in python, conversion between the different data point were concluded.

Dijkstra Algorithm

To successfully choose a path, an Dijkstra algorithm were used. The Dijkstra algorithm were retrieved from github.com [40], originally created by user *joyrexus*. The algorithm had a input of python dictionaries which contained information of each vertices connection between each other point and the length between. To transform the vertices from the Voronoi Diagram, each vertex needed information of which other vertices it was connected to. To do this, the edges from the Voronoi diagram, i.e the vectors between each vertex in the diagram. This were done along computing the weight of each connection, which would translate to the length of each vector. To extract a path, a start and goal point needed to be established. As the premise of the simulation were to navigate through the scanned area, the goal point were set on the upper limit of the scanned area, at $y = H$, where H is the obstacle map height. The vertex from the Voronoi diagram that were closes to the middle were chosen. If no vertex existed on the top of the diagram, the upper right corner were chosen. The start vertex were chosen similarly, but at the bottom of the obstacle map, at $y = 0$. This would result in the first waypoint in the path, and the start point for the Dijkstra algorithm.

When established, the dictionary containing the information of the connectivity of each node in the graph tree and their respectively length, along with the start and goal point, the Dijkstra algorithm could be called on. The Dijkstra algorithm then returned a set of points resulting in the shortest path. These points were then transformed back to coordinates and transformed to one single string variable to be transformed to the GUI of the ROV controller, Frigg.

6.2.3 Communication between Python and LabVIEW

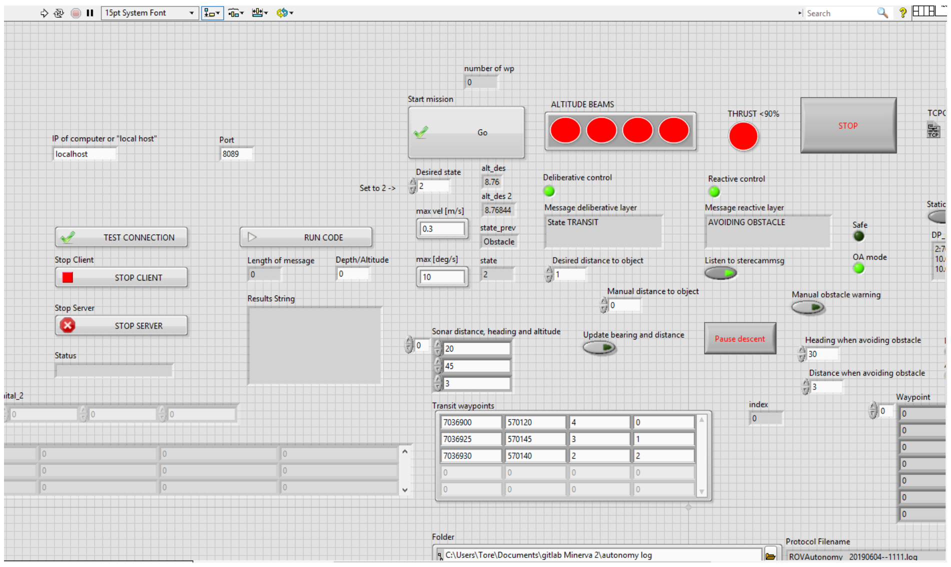
To establish communication between the ROV control interface and the python code, a Transmission Control Protocol (TCP) socket were created. The python program works as a server while LabVIEW acts as a client. The TCP connection only allowed for strings to be sent and received, so the return message from the client need to be a string variable, and the waypoints needed to be transformed from floating values to strings. The LabVIEW program sends messages to the running python script, indicating its action. When a call for waypoints are received, the python program starts and when completed, returns a set of of waypoints, describing the path. The server pseudo code is presented in Algorithm 1

Algorithm 1 Pseudo code serverLabVIEW.py

```
Result: Write here the result
create socket object
 create connection(address,port)
 while True do
    message = socket.receive
     if message = INIT then
        out = confirm connection
         send out
    else if message = RUN then
        out = imageReturn
         send out
    else if message = END then
        send close confirmation
         break
end
socket.close()
```

To receive the waypoints generated in the python script, a module were introduced to the LabVIEW

Autonomy Framework. The Autonomy Frame work is a setup for simulating the different stages of a ROV mission discussed in section 1.4. This was developed by I. Rist-Christensen in her thesis [2]. To simulate the ROV, a new module were introduced to the autonomy framework. This consisted mainly of the client in the client server relationship between the python script and LabVIEW. The Autonomy framework can be seen in Figure 6.6. Since the waypoints are



IP of computer or "local host": localhost
Port: 8089
number of wp: 0
Desired state: 2
alt_des: 8.76
alt_des 2: 8.76844
max vel [m/s]: 0.3
state_prev: Obstacle
max [deg/s]: 10
state: 2
Length of message: 0
Depth/Altitude: 0
Results String:
Status:
Message deliberative layer: State TRANSIT
Message reactive layer: AVOIDING OBSTACLE
Desired distance to object: 1
Manual distance to object: 0
Sonar distance, heading and altitude: 0 | 20 | 45 | 3
Heading when avoiding obstacle: 30
Distance when avoiding obstacle: 3
index: 0
Waypoint: 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0

ital_2

0	0	0

0	0	0
0	0	0
0	0	0

Transit waypoints

7036900	570120	4	0
7036925	570145	3	1
7036930	570140	2	2
0	0	0	0
0	0	0	0

Folder: C:\Users\Tore\Documents\gitlab Minerva 2\autonomy log
Protocol Filename: ROVAutonomy_20190604--1111.log

Figure 6.6: Part of the control panel for the autonomy window

6.3 Simulation

In the simulation test, four different sonar images were used as the basis of the path planning algorithm. The sonar images were cropped, and input to the image processing. There, a greyscale process were done to transform between the color image and the image represented as a scale of grey. This were done to focus on the different illumination levels on the image. Further, a median filter were applied to the image, smoothing the noise of the image. The median filter were applied with a 5x5 kernel. The image were then threshold at with a binary threshold with a threshold value of $min, max = [127, 255]$. This resulted in a pure black and white image. The image were then feature detected using the ORB feature detector, which are detailed in Section 4.3. The output keypoints were filtered and transformed into an obstacle map, which is the basis of the Voronoi Diagram. The path were selected by a Dijkstra Algorithm weighted by the path length.

The TCP socket receives the waypoint strings and transform them into a set of doubles. To transform from the local obstacle map reference frame to the NED, LabVIEW adds the x-position of the waypoint to the East value, and the y-position is added to the North direction. The waypoints in the Down positions were set constant at a constant altitude above sea floor of 7 meters. Since the control and guidance modules already developed in the AUR-lab, the ROV does not follow the state of 7 meters above the sea floor, but rather descends gradually through the path following simulation. This were not changed since this is a 2 dimensional path planning simulation.

The waypoints were based on a transit simulation using constant jerk reference model as the guidance module. The constant jerk guidance model is compatible with the planned path, as both are waypoint compatible. Environmental forces were set to 0, and the ROV dynamics and controls were simulated through the entire path, constructed by the input waypoints. The simulations were conducted on four different input images, resulting in 4 different paths and a corresponding trace of the paths. The results from the 4 simulations are presented in Section 7.3.

Chapter 7

Results

7.1 Results from image processing

The results found when choosing image processing techniques are presented in this section. The processing consists of image filtering and thresholding. This is done to find a balance between reducing noise and still being able to enhance information from the images for the detection process. Different detectors are also tried and the results are presented.

7.1.1 Filtering and Thresholding

As the first step is filtering, different filtering techniques were applied, with different properties. Figure 7.1 compares the three different filtering techniques median filter, average filter and Gaussian filter.

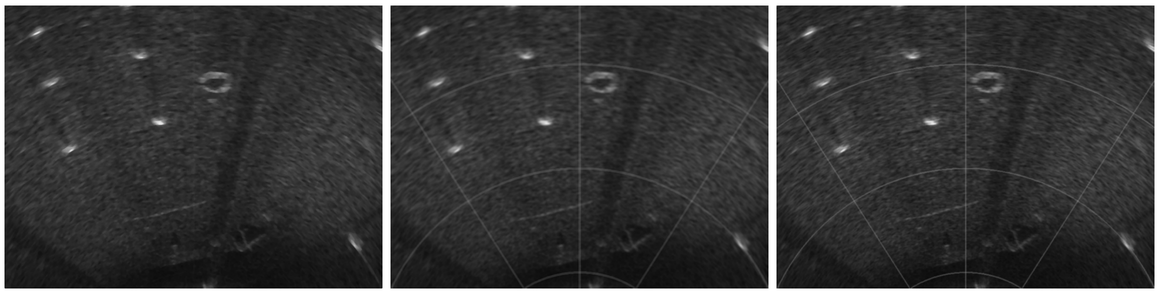


Figure 7.1: Median, Average and Gaussian Filtering with 5 × 5 kernel size

As can be seen in Figure 7.1, the average and Gaussian filter still displays the distance gauges in the images. This resulted in the median filter being chosen.

To determine the kernel size of the filter, some kernel sizes were compared. In Figure 7.2 the image is filtered with a kernel size of 3 × 3, 5 × 5 and 11 × 11, from left to right respectively.

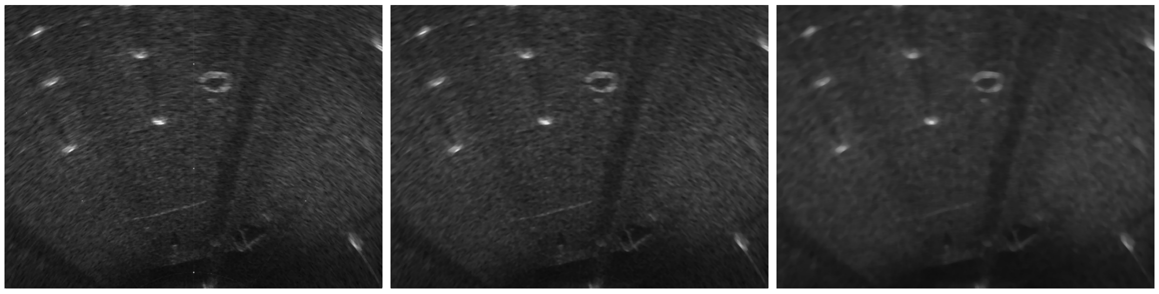


Figure 7.2: Median filtering with 3 × 3, 5 × 5 and 11 × 11 kernel size

As it appears from the figure above, the kernel size of 5 × 5 (in the middle) gives the best filtering results. The basis of this conclusion is the sufficient filtering without loosing to much details, like in the case for the kernel size of 11 × 11 (to the right), where the details are visibly distorted.

Further, the thresholding method were chosen. As can be seen from Figure 7.3 and 7.4, neither the Otus' thresholding or Adaptive thresholding yields a result were the obstacles are visible.

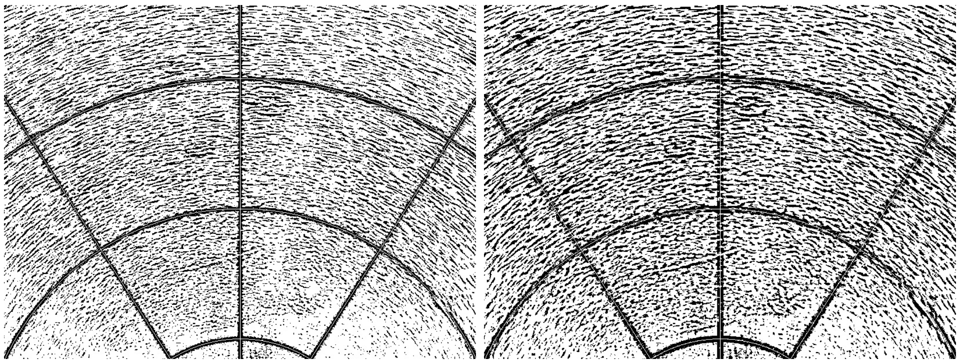


Figure 7.3: Gaussian and Median Adaptive Thresholding

Figure 7.3 shows both Gaussian and Median adaptive thresholding. Both yielded insufficient results.

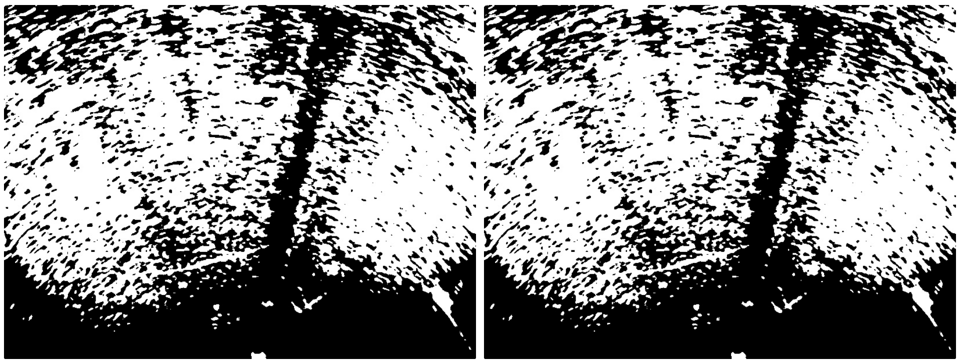


Figure 7.4: Otus' and Otus' + Binary Thresholding

Figure 7.4 shows Otus thresholding of the filtered image.

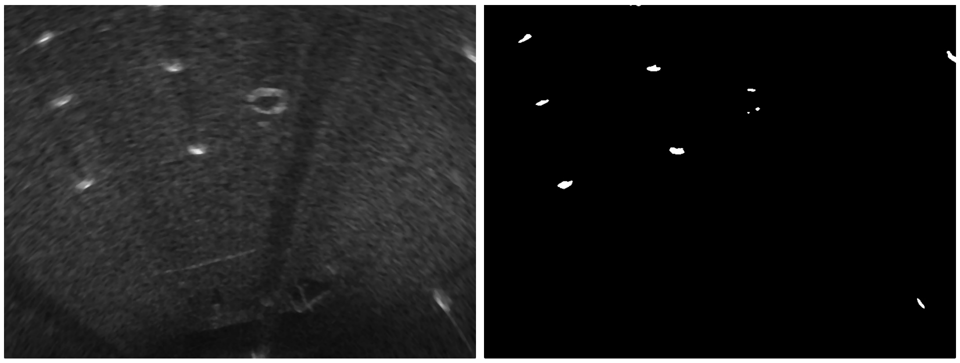


Figure 7.5: Binary Thresholding compared to the original median filtered image

Figure 7.5 shows the binary thresholding of the image. As can be seen, the thresholding did not lose the details of the features in the image, and enhanced the differences between the background and the features. The threshold limit were set to 127.

7.1.2 Detection

For the detection of features and keypoint generation, several algorithms were tested, namely FAST, SIFT, SURF, ORB and Harris Corner detection. The results from the FAST SIFT and SURF detection can be seen in Figure 7.6. As seen from the figure, FAST (left) yields a very low number of keypoints of the white spots. SIFT (middle) and SURF (right) yields some keypoints, but their accuracy is lower.

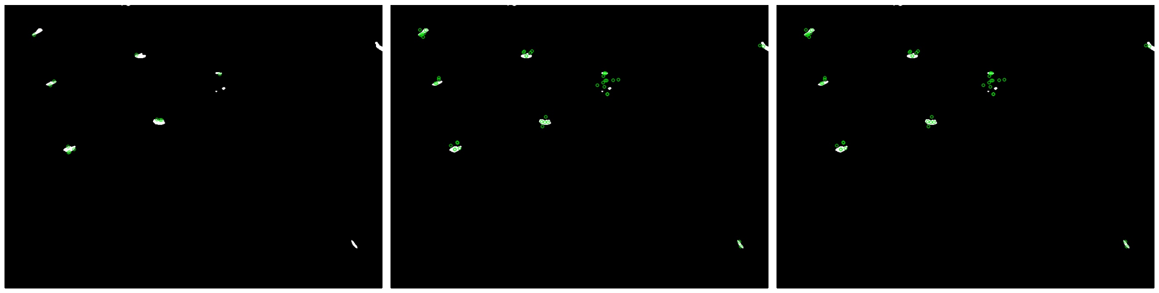


Figure 7.6: Detection with FAST (left), SIFT (middle) and SURF (right)

The two detectors that yielded the most keypoints are the Harris Corner detection (left) and the ORB detector (right), shown in Figure 7.7. As one can see, the Harris Corner detector yields the most keypoints out of the two, with similar accuracy between them. The ORB detector were chosen as the main detector due to the fact that the Harris Corner detector were to sensitive.

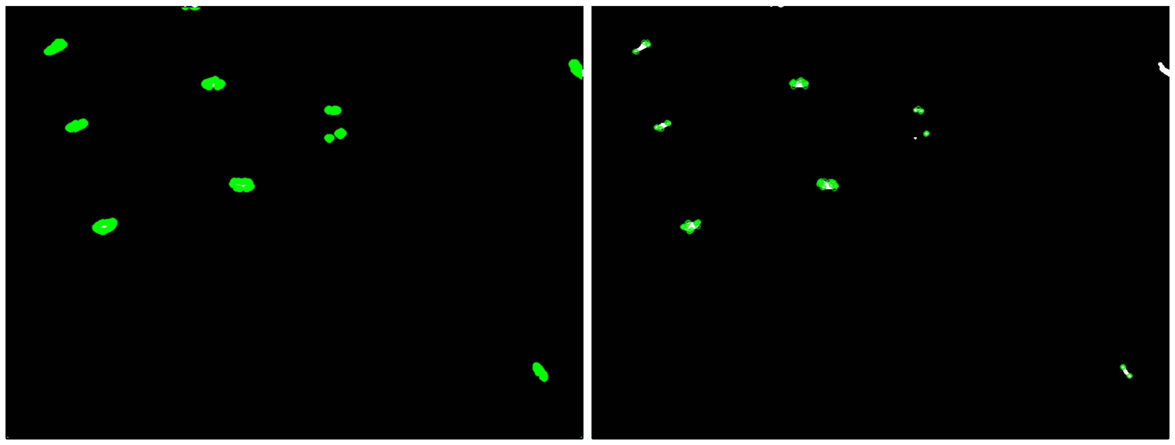


Figure 7.7: Detection with Harris Corner detection (left) and ORB (right)

7.2 Performance

The path planning process showed good performance in regards of computational time. Without plotting, the generation of a path from the input image, including processing, generation of keypoints, obstacle map creation and Voronoi diagram and path generation, the algorithm computational time were below 0.1 seconds. This implies that the algorithm are capable of online path planning.

7.3 Results from Simulation

With the results from the image processing obtained, the processing variable that yielded the best results were obtained. In the simulation, 4 different screenshots from the sonar video were tested. In this section the results from the path generation and the simulation of the ROV tracking these paths are presented.

7.3.1 Simulation 1

The first simulation were based on the screenshot from the sonar video seen in Figure 7.8, along with the corresponding features. The features captured shows recognition of the white areas which is the results from the thresholding.

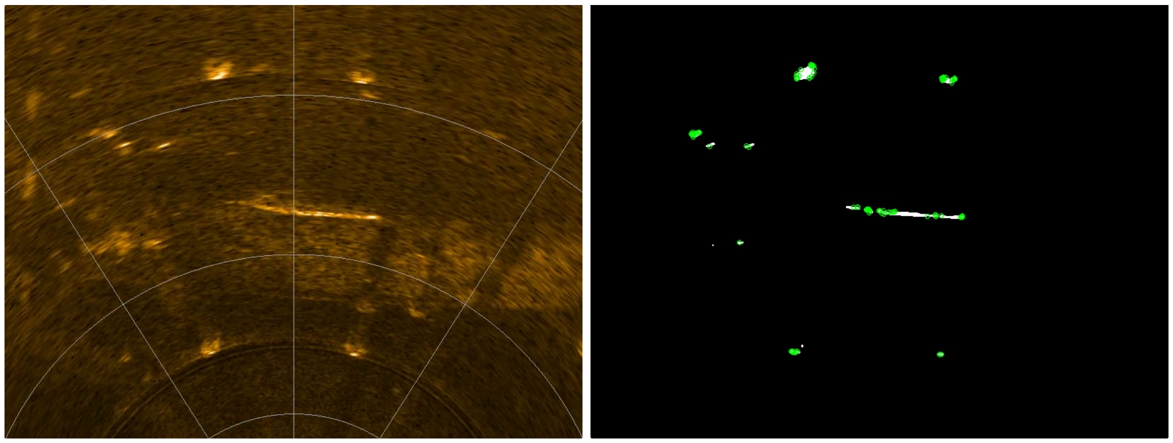


Figure 7.8: Simulation 1: Sonar frame with corresponding processed and detected features

To find the path, the features represented by keypoints in the figure above, were transformed into a detected feature map in a spacial domain. The detected keypoints are plotted in Figure 7.9

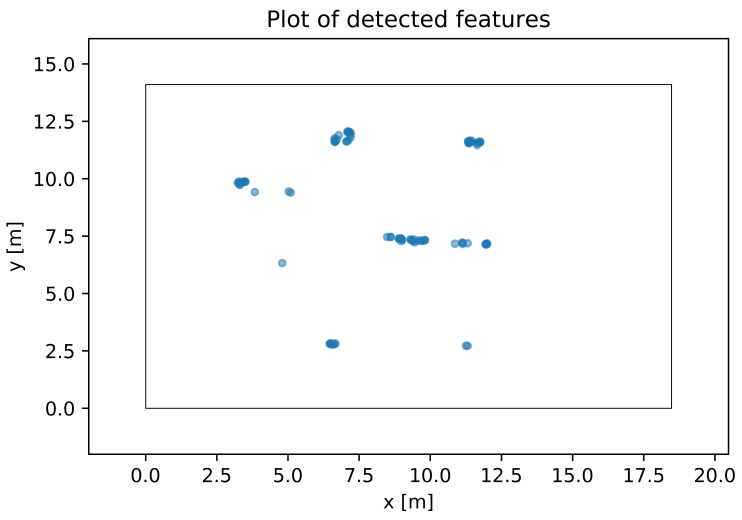


Figure 7.9: Simulation 1: Detected feature map

Figure 7.10 shows the filtered obstacles in a map.

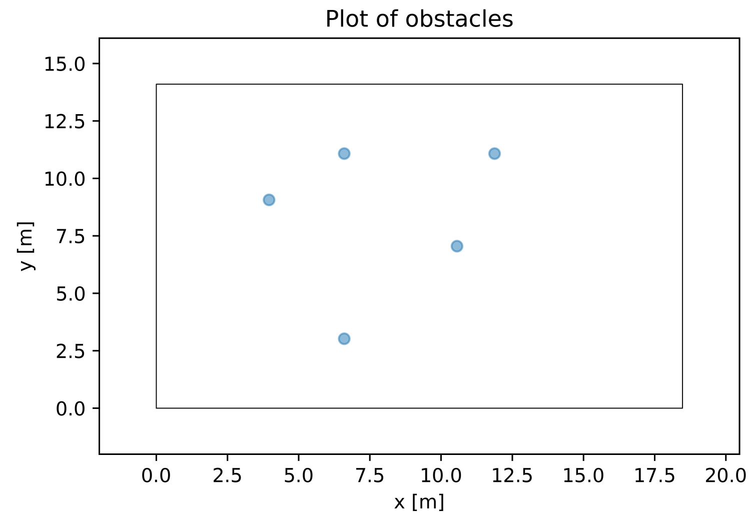


Figure 7.10: Simulation 1: Obstacle map

From the obstacles in Figure 7.10 the Voronoi diagram can be created. Figure 7.11 shows the output Voronoi diagram for the first simulation. The red points represents the obstacles, and the blue points is the Voronoi vertices.

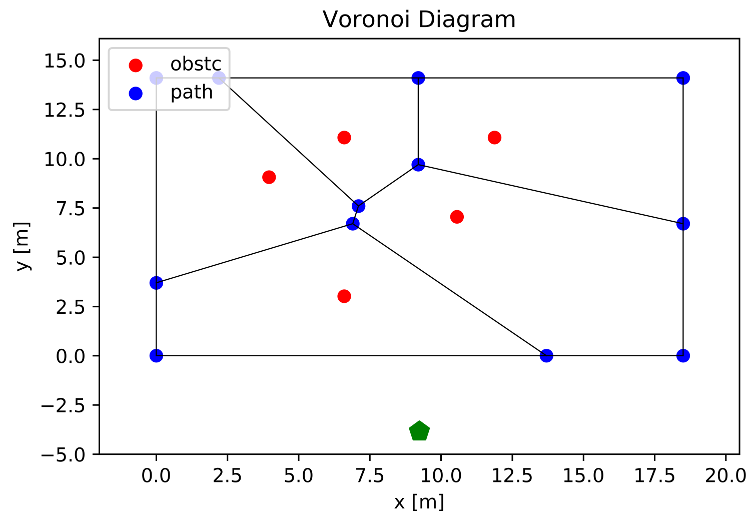


Figure 7.11: Simulation 1: Voronoi Diagram

Figure 7.12 shows the path generated from the Voronoi diagram. The path is indicated in blue.

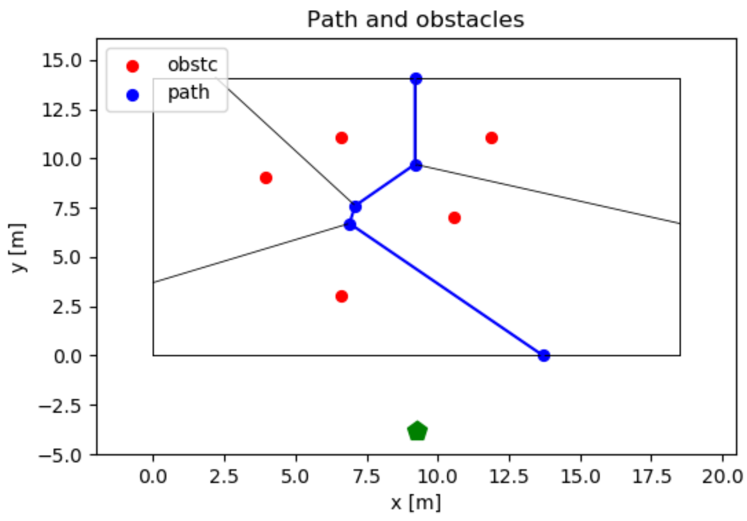


Figure 7.12: Simulation 1: Output path

To test the feasibility of the path, the ROV are fed the path from Figure 7.12. The start position of the tracking is the start position of the ROV, indicated by the green pentagon in Figure 7.12. The waypoint positions are indicated in Table 7.1. The basis of the waypoints are the start position of the ROV, which are $WP_{start} = [7036895, 570130]$, and the end position is the last WP in the path.

	North [m]	East [m]
Start	7036892	570128
Point 1	7036896.8	570132.5
Point 2	7036903.5	570125.7
Point 3	7036904.4	570125.9
Point 4	7036906.5	570128
Goal	7036910.9	570128

Table 7.1: Waypoints in North-East Frame

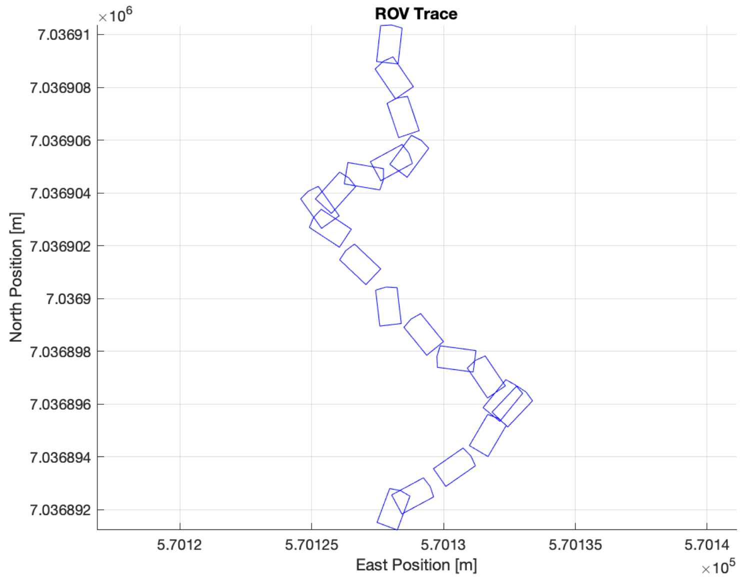


Figure 7.13: Simulation 1: Trace of the ROV

Figure 7.13 shows the trace of the ROV position during the simulation. The ROV starts at the established start point before moving to waypoint number 1. The ROV traces the waypoints until it comes close to the points 2 and 3, where the ROV moves over them both.

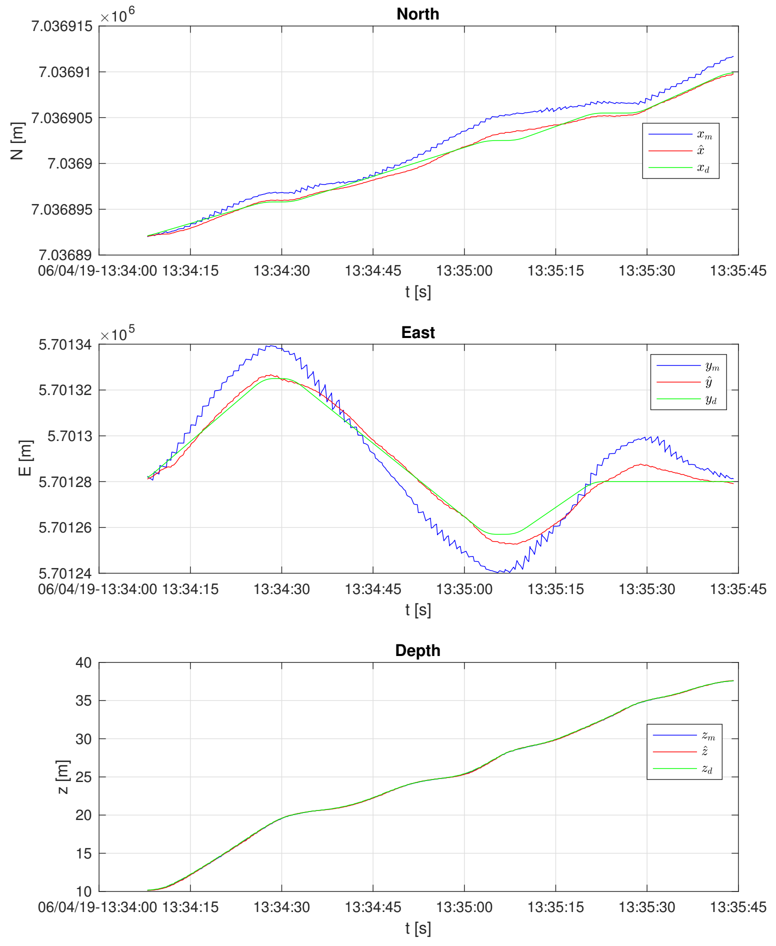


Figure 7.14: Simulation 1: Spacial states with corresponding desired states

In Figure 7.14 the states N-E-D are plotted over the time of the simulation. The red indicates the estimated position and the green indicates the desired position. The red and green more or less coincide through the simulation.

7.3.2 Simulation 2

Simulation 2 considers a new screenshot of the sonar imaging video stream. The detected features along with the original image are shown in Figure 7.15.

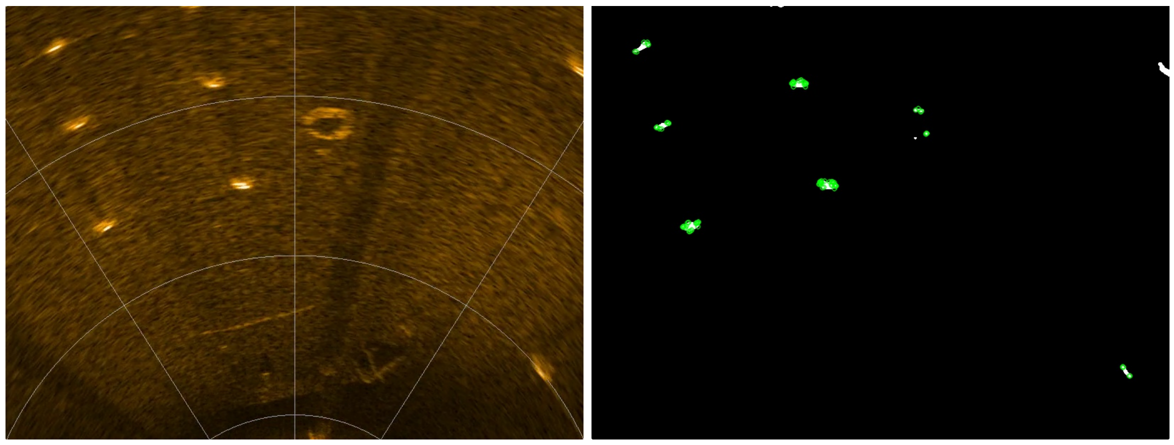


Figure 7.15: Simulation 2: Sonar frame with corresponding processed and detected features

Figure 7.15 shows the detected features of the image along with the original image.

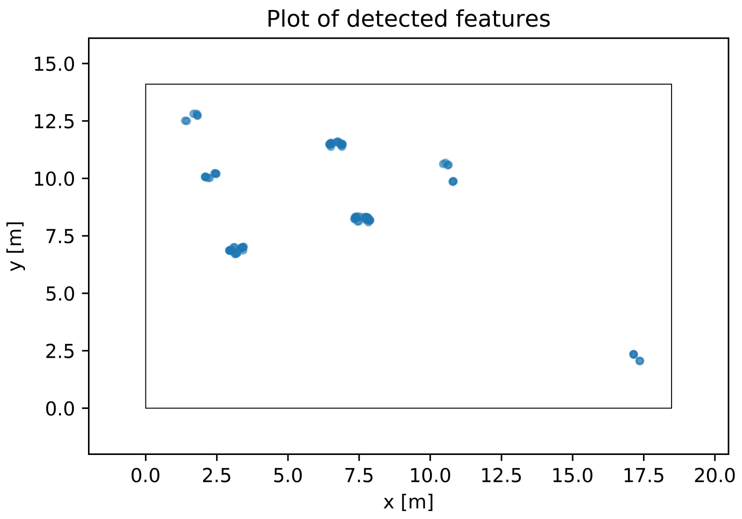


Figure 7.16: Simulation 2: Detected feature map

In figure 7.16 shows the detected features in a spacial map. The features are represented by the blue points. The inner frame indicates the size of the scanned area in the map.

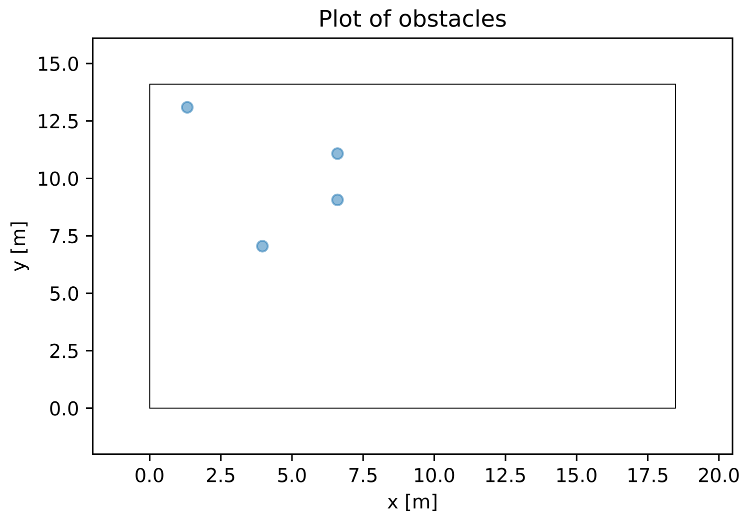


Figure 7.17: Simulation 2: Obstacle map

Figure 7.17 shows the filtered objects plotted in the same map. Comparing Figure 7.16 and 7.17 shows that several of the features are filtered out.

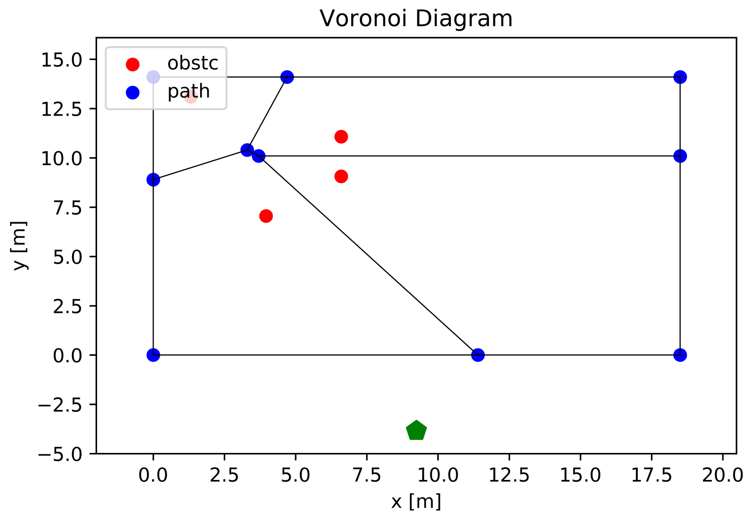


Figure 7.18: Simulation 2: Voronoi Diagram

From the obstacles plotted in the figure above, the Voronoi diagram is created. Figure 7.18 shows the resulting Voronoi diagram. The obstacles are closely position, resulting in a diagram with close vertices and fewer edges. The ROV position is indicated by the green pentagon in the bottom part of the plot.

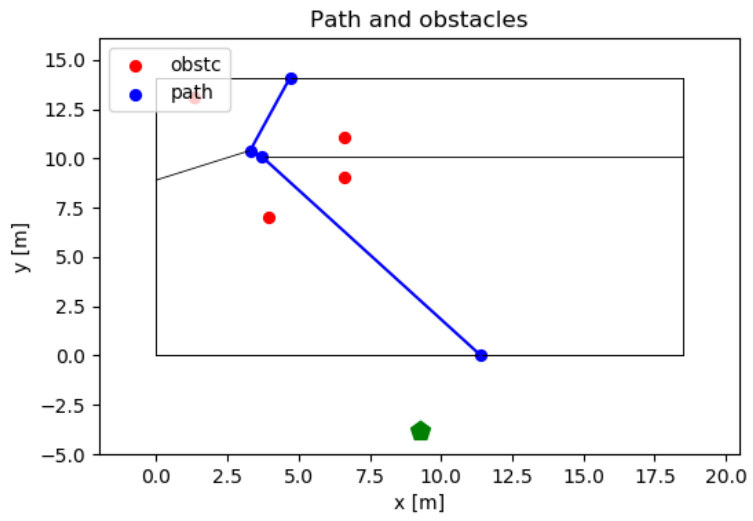


Figure 7.19: Simulation 2: Output path

Figure 7.19 shows the resulting path chosen by the Dijkstra algorithm. The path cuts right trough the obstacles, and follows the edges of the Voronoi diagram to the top.

Transformed to the NE-frame the waypoints are presented in Table 7.2. The start point, i.e the initial position of the ROV is found as $N, E = [7036892, 570130]$. Following are 4 waypoints including the goal waypoint.

	North [m]	East [m]
Start	7036892	570130
Point 1	7036895.8	570132.2
Point 2	7036905.9	570124.5
Point 3	7036906.2	570124.1
Goal	7036909.9	570125.5

Table 7.2: Simulation 2: Waypoints in North-East Frame

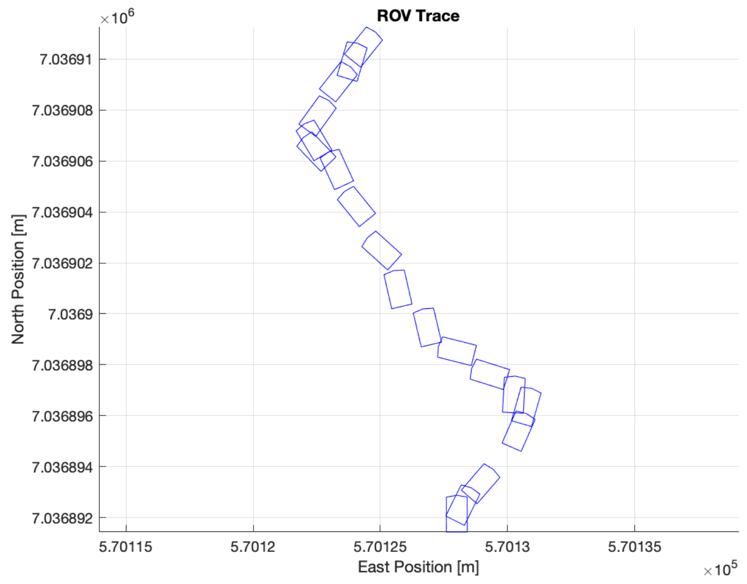


Figure 7.20: Simulation 2: Trace of the ROV

Figure 7.20 shows the path of the ROV in the NE frame. The ROV tracks the path, given in Table 7.2. The path consists of straight line segments, but traced path indicates overshooting of the given path.

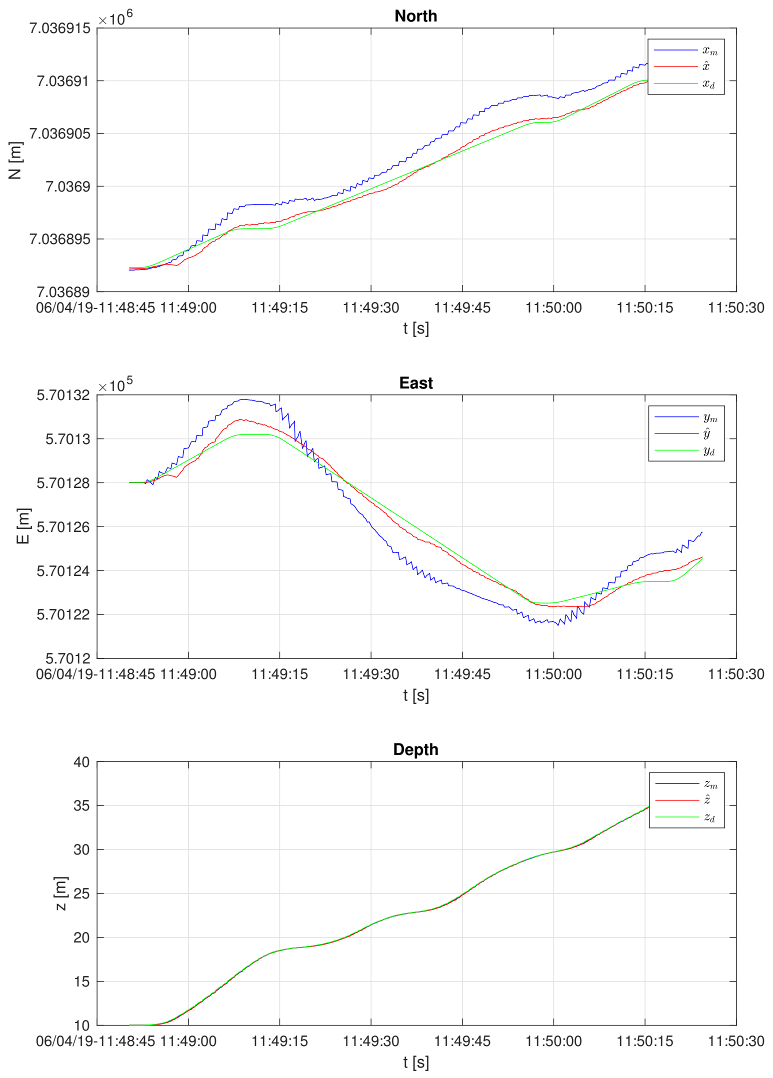


Figure 7.21: Simulation 2: Spacial states with corresponding desired states

The spacial states of the ROV is plotted in Figure 7.21. The green line indicates the desired state, which consists of straight line segments. The red line, indicating the estimated state follows the desired state, with some oscillations.

7.3.3 Simulation 3

In Figure 7.22, the third frame which are the basis of the third simulation are presented. The detected keypoints are indicated with the green points in the figure below. The corresponding image can be seen in the same figure.

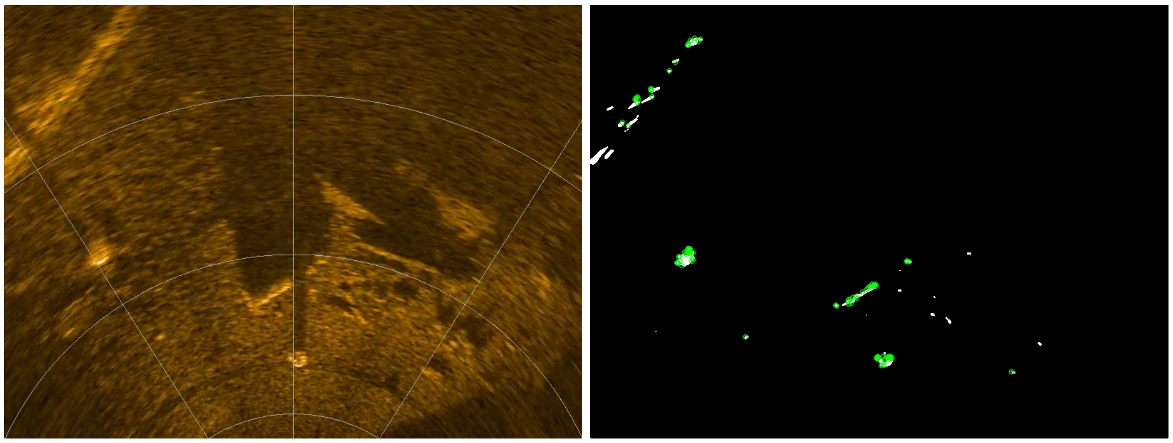


Figure 7.22: Simulation 3: Sonar frame with corresponding processed and detected features

Figure 7.23 shows the same keypoints plotted in a spacial map. The inner frame indicates the scanned area.

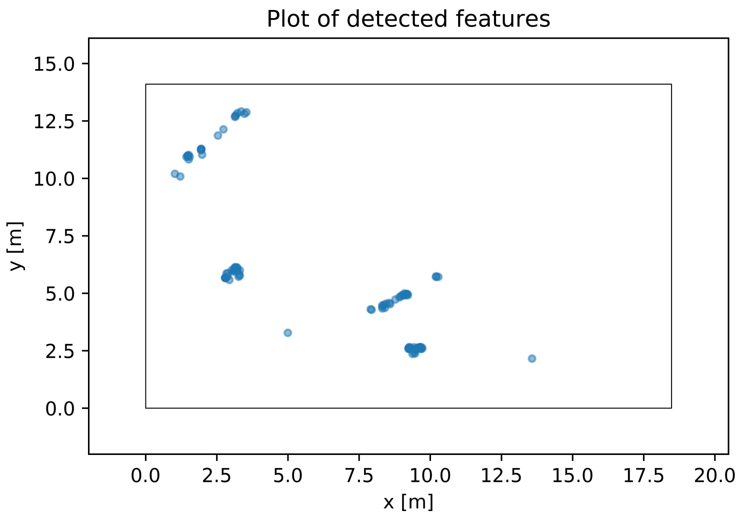


Figure 7.23: Simulation 3: Detected feature map

In Figure 7.24, the features are filtered and presented as obstacles in the spacial map. Comparing the two plots, one can see that some of the features with fewer keypoints detected is filtered, leaving fewer obstacles than detected keypoints.

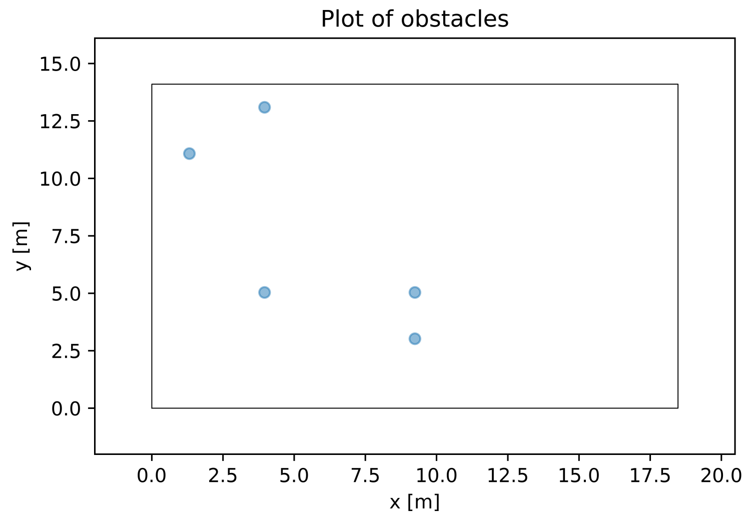


Figure 7.24: Simulation 3: Obstacle map

From the obstacles in Figure 7.24 the Voronoi Diagram is created. It can be seen in Figure 7.25. The path created are in turn based on the Voronoi diagram and is presented in Figure 7.26.

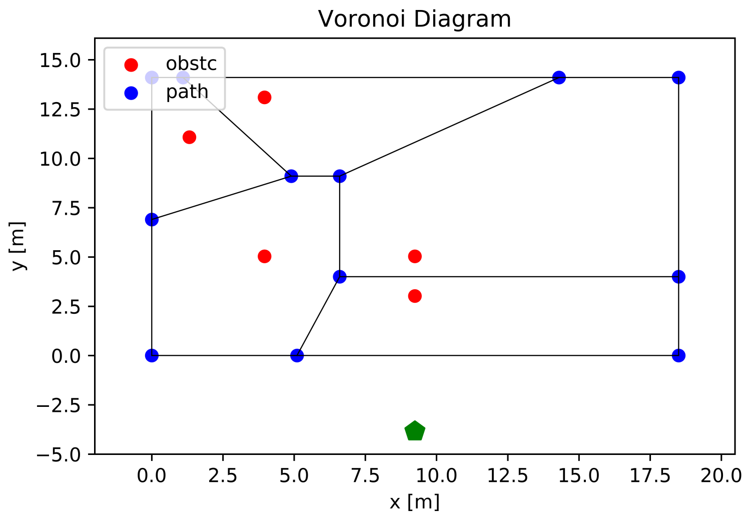


Figure 7.25: Simulation 3: Voronoi Diagram

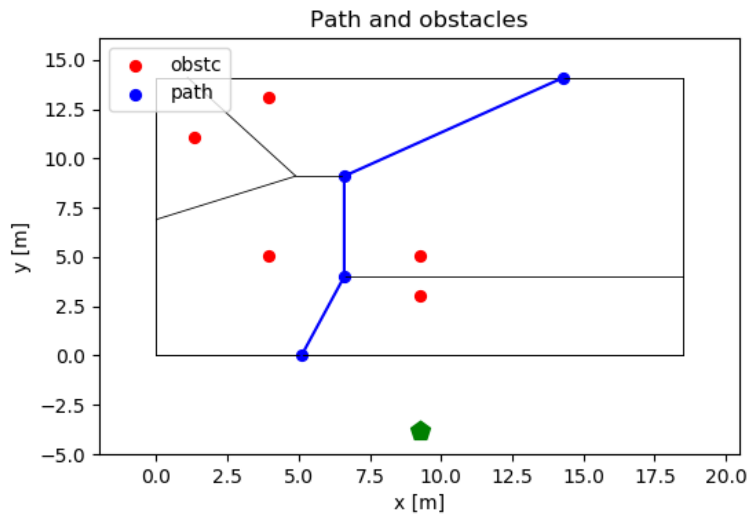


Figure 7.26: Simulation 3: Output path

In Table 7.3 the waypoints used in the simulation are presented. Starting waypoint indicate the ROV start position. The waypoints generated are based on the start position of the ROV and the relative position to the obstacles from Figure 7.26. The start point is located at the coordinate $N, E = [7035894, 570128]$ and the end point is found at $N, E = [7036911.9, 570128]$

	North [m]	East [m]
Start	7036894	570128
Point 1	7036897.8	570129.4
Point 2	7036899.1	570129.4
Point 3	7036900.8	570128
Goal	7036911.9	570128

Table 7.3: Simulation 3: Waypoints in North-East Frame

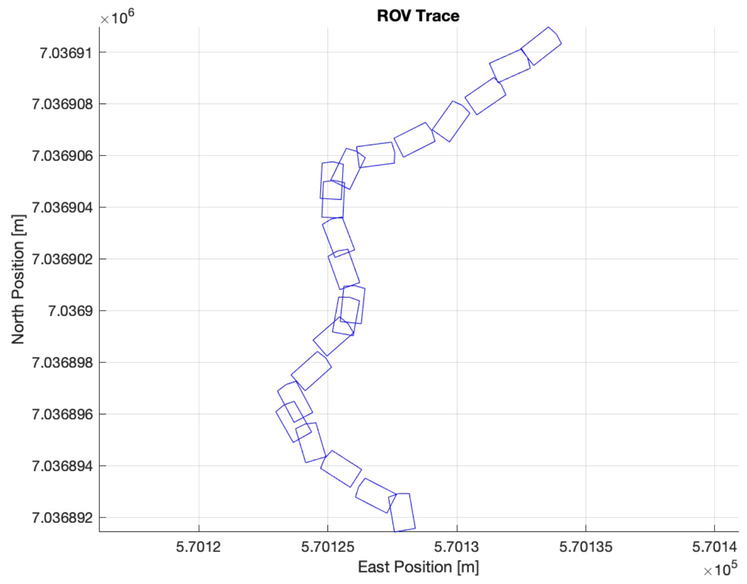


Figure 7.27: Simulation 3: Trace of the ROV

In Figure 7.27, the tracing of the ROV path is presented. The tracing corresponds with the waypoints in Table 7.3 and the path from Figure 7.26. The ROV turns hard to the left to reach the first waypoint, before turning hard right to follow the path towards the second waypoint.

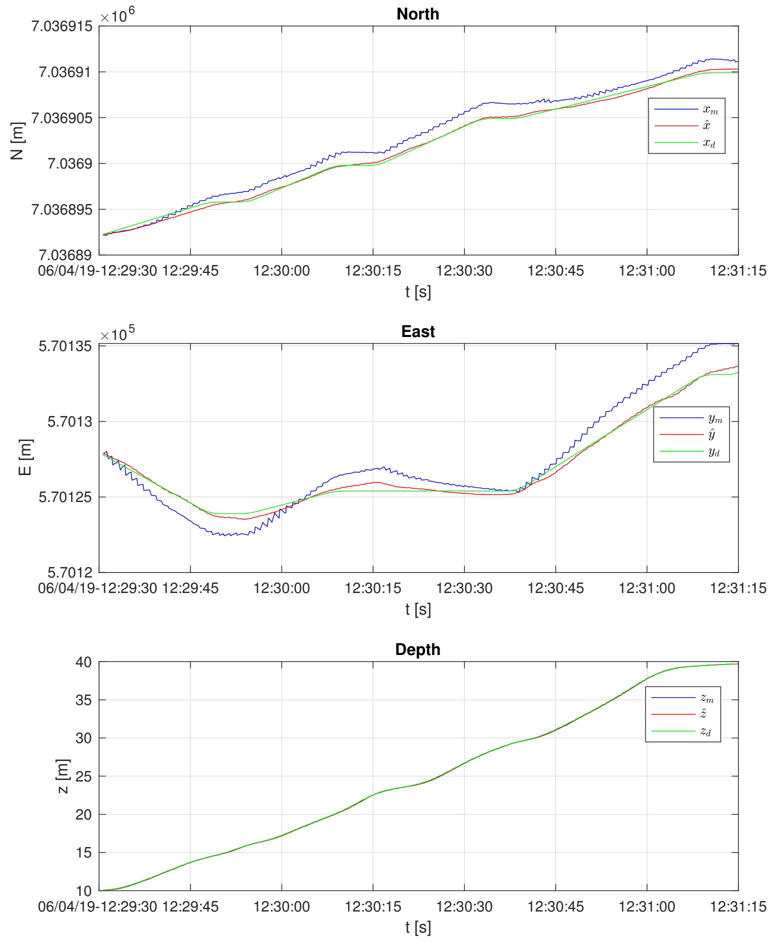


Figure 7.28: Simulation 3: Spacial states with corresponding desired states

The corresponding states for the tracing are plotted in Figure 7.28. As seen in the figure, the North desired state and the north estimated state coincide through out the simulation.

The East state estimate and the East desired state follow each other through the simulations with some deviations in the hard turns.

7.3.4 Simulation 4

The image used in simulation 4 can be seen in Figure 7.29. The features detected are presented in the same figure, indicated by the green keypoints. Comparing the two images, one can see that the structure indicated in white in the beginning are omitted from the detected features.

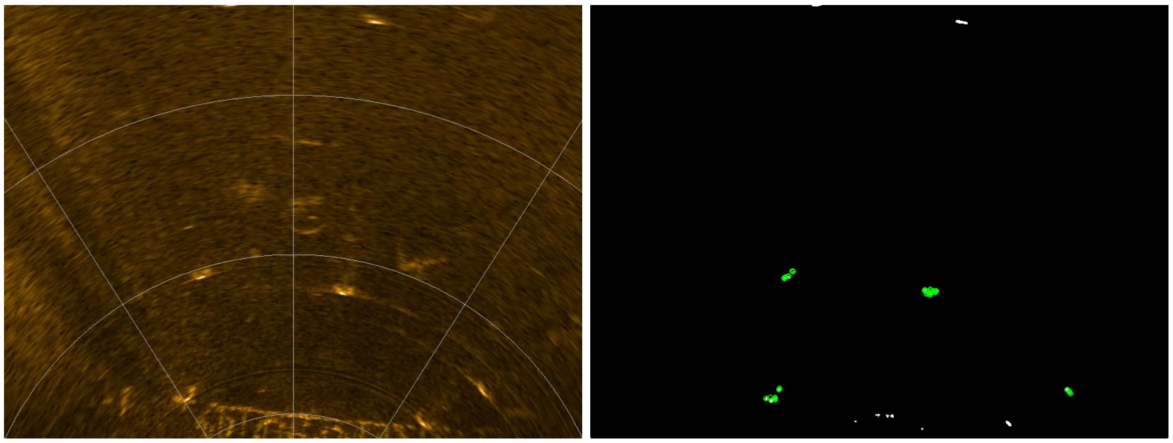


Figure 7.29: Simulation 4: Sonar frame with corresponding processed and detected features

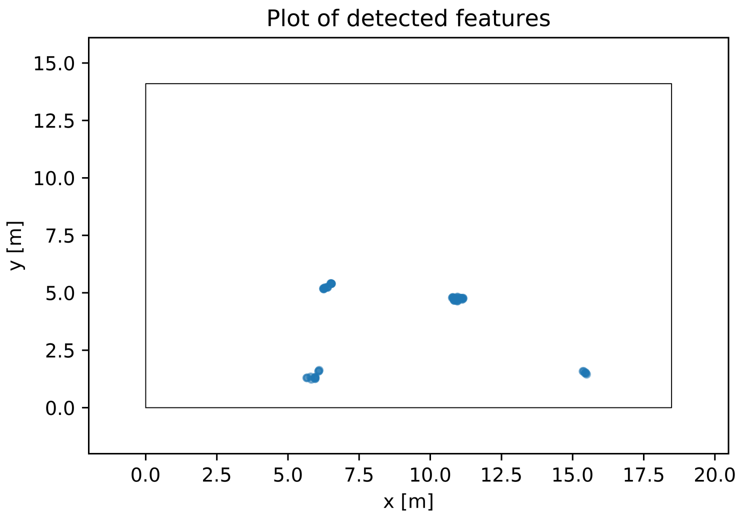


Figure 7.30: Simulation 4: Detected feature map

The keypoints are presented in the spacial map in Figure 7.30. The features are largely located to the bottom of the scanned area, which corresponds to the original image.

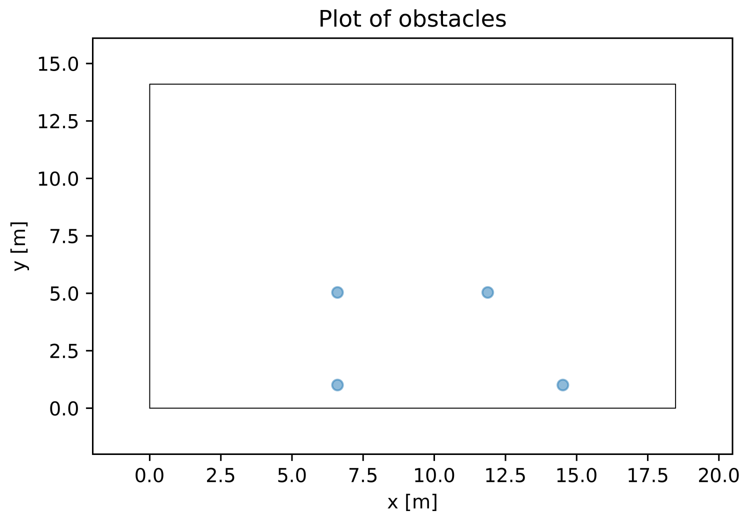


Figure 7.31: Simulation 4: Obstacle map

The keypoints are filtered and turned into single obstacles. The obstacles are presented in Figure 7.31. The corresponding Voronoi Diagram are created on the basis of the obstacles and can be seen in Figure 7.32.

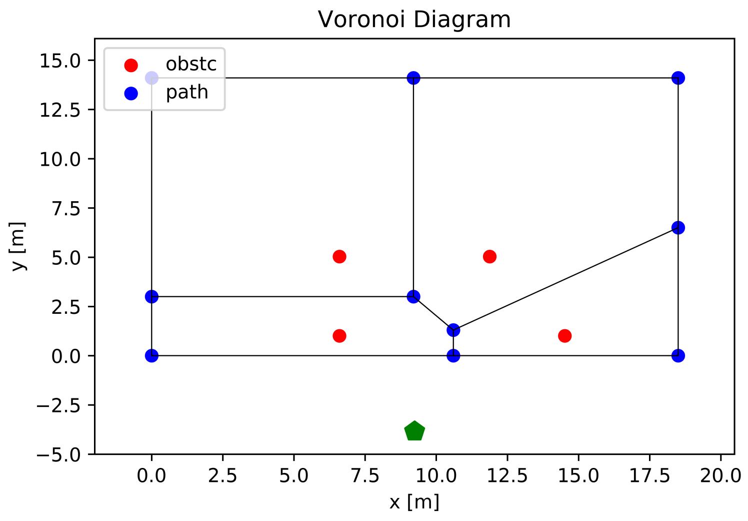


Figure 7.32: Simulation 4: Voronoi Diagram

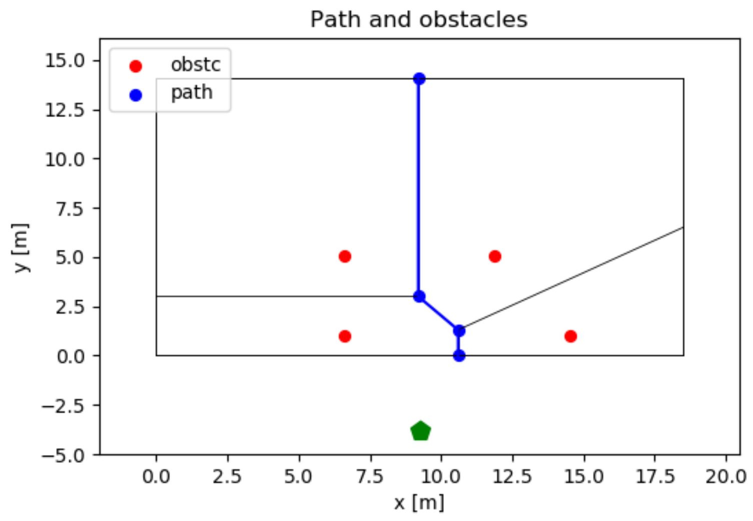


Figure 7.33: Simulation 4: Output path

The path is presented in the local spacial map in Figure 7.33, and are presented as waypoints in the NE frame in Table 7.4.

	North	East
Start	7036895	570130
Point 1	7036898.8	570134.5
Point 2	7036905.5	570127.7
Point 3	7036906.4	570127.9
Point 4	7036908.5	570130
Goal	7036912.9	570130

Table 7.4: Simulation 4: Waypoints in North-East Frame

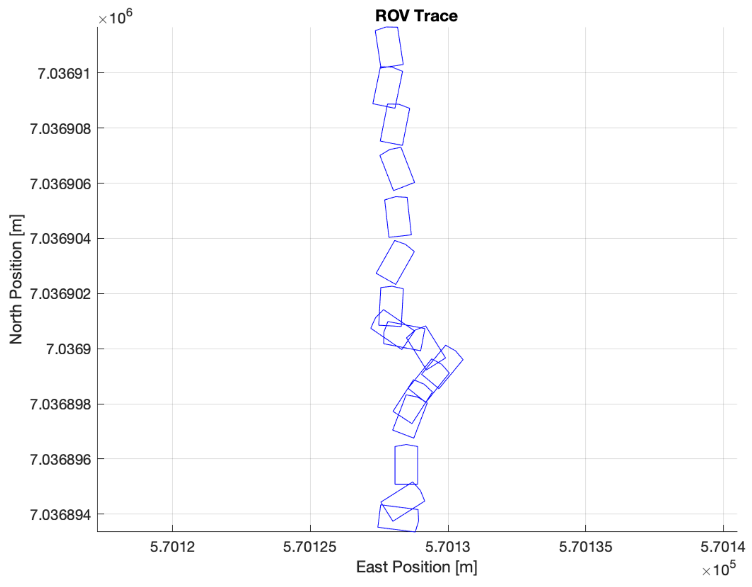


Figure 7.34: Simulation 4: Trace of the ROV

Figure 7.34 shows the tracked path of the ROV. The trace follows the path given in Table 7.4.

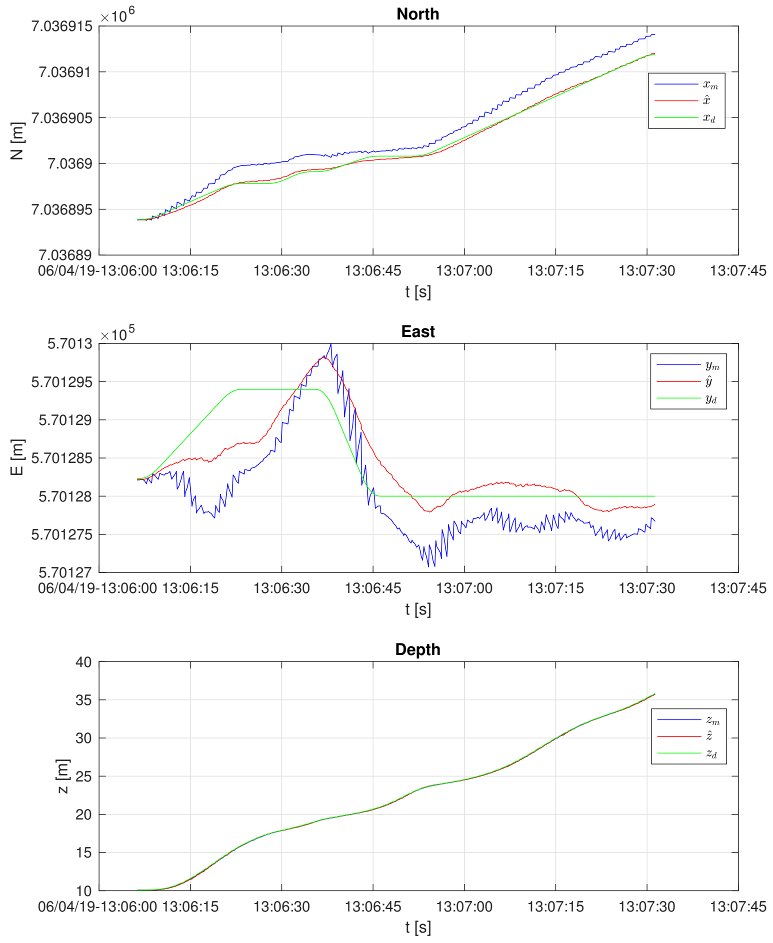


Figure 7.35: Simulation 4: Spacial states with corresponding desired states

Figure 7.35 shows the states for the simulation. The value for the state representing North direction steadily climbs through the simulation, while the East desired state shifts hard as it follows the path. The estimated east state overshoots the desired state. The ROV is not able to keep the state at the desired position, and its reaction are delayed.

Chapter 8

Discussion

8.1 Image Detection

The image processing contains filtering and thresholding. The filtering techniques tested consisted of mean, average and Gaussian filtering. The method that yielded the best results were the mean filtering method. The mean filtering filtered out the noise of the sonar image well, while still containing details. In addition, it were the only filtering method that filtered the ranging indicators of the sonar image, so that it did not appear after processing. Different kernel sizes were also tested, with less differences between them. The final kernel size were chosen as 5×5, due to its performance in reducing noise and still keeping enough details.

The different thresholding techniques applied consisted of binary, adaptive mean and Gaussian thresholding and Otus' Method for thresholding. The only thresholding technique that gave close to some feasible results were the binary thresholding technique. The Otus' thresholding were sensitive to the noise in the image, and highlighted every detail of the image, as can be seen in the figure 7.4. The Gaussian and median adaptive thresholding highlighted the ranging indicators of the image, and left the background thresheld down to the noise level of the image. The binary thresholding produced images with the high lumination features of the image highlighted and the background in black. Different threshold values were tested, and the chosen value of 127(out of 255) were chosen. This is a value that is very dependant on the input image, and is probably due to change if used with different sonar imaging data from another sonar or software. The image detection showed some promise in the results. The high lumination objects in the scanned sonar images were threshold with good accuracy. However, there were instances were the thresholding resulting in a segmentation of the image that would not only include the high luminated features on the ocean floor, but also the ocean floor itself. This was very prevalent in the cases were the sonar sensor shift fast in rotation. The rotation had a negative effect on the sonar image, and thus the detection. This could lead to a possible detection on nothing but sea floor, leaving a large number of obstacles in the obstacles map. This could cause both unnecessary turns and over complex paths, but also collisions, in the way that if the obstacle is larger than the scanned area, and the lumination value of the obstacle is high enough, it will not be marked as one obstacle, but a grid of several obstacles. In the obstacle detection phase this was mended by turning two confirmed obstacle into one if they were adjacent,as can be seen when comparing Figures 7.9 and 7.10, but if the obstacle is large enough, the problem would still arise. A possible solution to this would be to join all coinciding obstacle into one obstacle, which could work if the path generated were substantially far enough from the obstacle center.

Further, the thresholding techniques and image detection gives no confidence in the regard of accuracy. The detection algorithm ORB detects the thresheld points, but there are no further reassurance that both the created keypoints are, in fact obstacles, or that no other obstacles are present. This were partly mended by making a filter that had a minimum required set of keypoints to confirm an obstacle. The filter were constructed by checking one square of a grid, with a priory set grid size. The obstacle were confirmed and spatially placed in the middle of the grid square if there were more than 6 keypoints present in that particular square. The grid size were chosen as the height and with divided by 7, creating a 7x7 grid with width $W/7$ and height $H/7$. The number of keypoints to create from the detected features were also a variable. The number used in the simulations were that of 100 key points. This was chosen along with the number of required keypoints to find a balance between accuracy and over sensitive key point generation.

8.2 Path Planning

The results from the path planning using Voronoi Diagrams and Dijkstra algorithm were successful. However, a problem with the Voronoi Diagram is that it depends solely on the accuracy of the obstacles or points it needs to be created. By creating the Voronoi vertices and the corresponding edges, the Voronoi diagram ensures on its own that the edges are as far away from the obstacles as possible, if there needs to be a set path on the between the obstacles. The result of this is that without a guarantee of the position of all the obstacles, the Voronoi edges can not be confirmed as feasible path segments. Furthermore, in the case of Voronoi creation using few input points, the Voronoi diagram forces itself to create a vertex in the middle of the obstacle placements. This could result in that the path later selected are not in fact the safest, or feasible shortest path at all. This is illustrated in Figure 7.18. In this obstacle and path map, there are only 4 obstacles. The path selected cuts right in the middle of the obstacles, forcing its way through the obstacles, only to satisfy the basis of the Voronoi Algorithm. The premise of the path is to go from the bottom middle of the map to the middle top, and a better path here, which keeps longer distance to the obstacles and is shorter and with fewer is the simplest solution by making the path go straight up from the start position. Due to the nature of the Voronoi diagram, this solution is not proposed. This illustrates some of the problems with Voronoi based path planning, which are that feasible paths created, although technically feasible, have no guarantee that its the global best path.

A more successful example of the Voronoi Diagram can be seen in 7.11. Here the non trivial solution of choosing a path right ahead will conflict with obstacles present. The Voronoi Diagram creates a grid of feasible paths.

Another problem with the Voronoi diagram is that the are made up of linear piece wise path segments. This results in potential hard turns through the path. However, this is, as discussed later, in general not a large problem for actuated and over-actuated vehicles such as the ROV, which are capable of hard turns but can cause problems if the turns are to sever. The Dijkstra algorithm is based on finding the best path in a set of weighted paths. In this thesis, the paths are weighted by their length only. The weights however, can be changed. In the regards of combining Voronoi Diagrams and Dijkstra algorithm, a possible solution for the optimal path selection can be to include a cost function for both hard turns and closeness to the obstacles. The distance between the Voronoi edges and the obstacles are available information in the creation of Voronoi diagrams. As the information about edges are the coordinates from the two points defining the edge, there are enough information to calculate the angle between them. Including both thees constraints to the weighting of the path segments would ensure that the path is as far away from the obstacles, while still reducing the turning rate needed and the path length.

8.3 Simulation

Four simulation were performed, and the results are discussed in this section. Simulation 1 were based on the input image found in Figure 7.8. Comparing the two images in the figure one can see that the thresholding and feature detection returns keypoints in the highlighted features from the image, which are translated into the spacial map shown in figure 7.9. The filtering applied, which both include a reduction in obstacles, and the generation of one obstacle point based on the set of features detected forming a line in the feature map. Comparing the obstacle map from Figure 7.10 with the input image, there seems to be some discrepancies between the visible features and the output obstacle map. This is mostly due to the filtering applied, and it would seem that the filtering caused the bottom right feature to not be counted as a obstacle. This shows the potential fault of the method of both relying on the segmentation and filtering. This could be mended by increasing the number of features to detect, or lowering the threshold value of the filter, relying on less keypoints to satisfy the feature as an obstacle.

Assuming the obstacles to be detected satisfactory, the Voronoi diagram returns a web of feasible path segments, and the Dijkstra algorithm chooses the shortest path from the ROV start position and the goal position. The trace from the simulation is shown in Figure 7.13. Compared to the path displayed in Figure 7.12, the ROV visibly follows the path. Between waypoint 2 and 3, indicated in Table 7.1, the ROV does not reach both the waypoints, this is most likely due to the small distance between them. In Figure 7.14, one can see that there are larger difference in the desired states compared to the estimated states in the time frame were the ROV goes from waypoint 2 to 3, in the time frame 13:35 to 13:35:15. This shows that if the change in desired appears fast enough, the ROV is not able to trace the exact path, i.e keep the desired state and estimated state equal each other. On the other hand, the difference seems to be less than 1 meter at most, which based on the distance between the obstacles of several meters should be whit in a permissible range. Through the simulation the states appears to coincide with their desired states, with some overshooting when large changes appear

The basis of simulation 2 is the input image shown in figure 7.15, with the processed image and corresponding detected features in the spacial map in Figure 7.16. Comparing the features from the original image, one can see that all the features with high luminance are detected. The features are the basis of the obstacle creation shown in Figure 7.17. As one can see, the filtering reduces the number of obstacle compared to the feature map. The corresponding Voronoi diagram is created and creates vertices and edges in the general vicinity of the obstacles. As mentioned earlier, this shows some of the back sides of the Voronoi diagram. The path is created and is shown in Figure 7.19, with the corresponding trace of the map based on the waypoints in Figure 7.20. The path here is selected as the shortest feasible, from the Voronoi Diagram. As can be seen, the path cuts through the cite were the obstacles are present, as a direct result of Voronoi Diagram as feasible path generation. The tracing of the selected path shows that the path is feasible for the ROV. Some of the same difficulties present in Simulation 1 is present here as well, but since the distance between the two closest waypoints, waypoint 2 and 3, is so small, the consequences of not clearly reaching both are not so sever. When looking at the states for the ROV, one can see that it is particularly the hard turns, or changes in east state that causes problems for the tracing of the ROV. However, the places were the desired states and the estimated states deviates, there are seemingly less than one meter deviation, and only in the east direction.

For simulation 3, the features detected can be seen in Figure 7.22. Sonar image gives an overview of some rubble on the sea floor, which are detected, along with a feature in upper left corner. The features gives the basis for the obstacles, and as one can see in Figure 7.24. In the image one can see that the obstacle in the upper right corner is divided into two obstacles. This highlights the potential problem with using the point obstacles as a form of graphically express larger obstacles. The corresponding Voronoi Diagram creates an edge right through the obstacle, and thus is creating a unfeasible path segment. This again emphasises the need for a way to confirm that the obstacle points actually corresponds with the physical objects detected in the sonar image frame. If the resulting path had been chosen between the two obstacle points, the vessel could face impact and crash with the obstacle. This implies that there needs to be at least a second or third sensor system, confirming that the path ahead is actually clear. An other solution would be to represent the occupancy grid in much higher fidelity. This further would possible result in the conclusion that the Voronoi diagram is not the best suited algorithm for this type of path planning, as it needs a set of input points representing obstacles. A possible solution could also be in using a occupancy grid, using the center of area of the obstacles to calculate the input points of the Voronoi Diagram and then use the occupancy grid again to confirm that the path segments, or edges, create by the Voronoi diagram are feasible or not. The resulting path is simulated and is shown in Figure 7.27. The output simulated tracing of the ROV show that the ROV follows the path well, as there is no hard turns in the path. Reviewing the states from Figure 7.28 reveals that the tracking of the desired states are performed well through out the simulation. The North state shows almost no deviation between the desired state and the estimated state, and the east state deviates less than 0.2 meters from the desired state. The reason for this is probably the lack of hard turns during along the path.

The input image along with the detected features can be seen in Figure 7.29. As can be seen, the detection misses some of the details from the bottom of the image, as well as some in the middle. This highlights the difficulties of choosing the threshold value, as the binary thresholding removes the details, as can be seen in the image on the left. The detected features are marked in green, and consist of only the highest luminated features. The resulting path can be seen in Figure 7.33. The accurate tracking for this path were the one with the lowest performance. As can be seen when comparing the path in Figure 7.33 and Figure 7.34, the path is not full filled in the east direction. Instead of turning right straight away, the ROV slowly turns right, not keeping its route. When the transition between waypoint 1 and waypoint 2 happens, it turns sharply to the right to compensate. When reviewing the states in Figure 7.35, one can see the results of this. During the simulation, the east state does not follow its desired state until halfway through the simulation, and then also oscillating quite a bit. The reaction to the hard turning is very delayed, and the control system overshoots the desired state when first reacting.

In general, when comparing all the simulations, the ROV is able to keep itself at the desired states, but it has some deviation when dealing with hard turns.

Chapter 9

Conclusion and Further work

This thesis regards the use of sonar imaging and Voronoi Diagrams in underwater path planning. An implementation of path planning scheme based on extracting features from a under water sonar video stream has been done. Different processing techniques and different feature detectors were tested. Lastly, a Voronoi diagram based on the obstacles found in the sonar image were created, and the Dijkstra Algorithm were used for choosing the shortest path. The paths, consisting of waypoints, were used in a transit simulation of a ROV.

The input for the path planning scheme was the sonar image. To detect the features in the sonar image, a set of processing techniques and a feature detection were done. The image processing included filtering and thresholding, to extract the information. The findings showed that the best combination of filtering were mean filtering and binary thresholding. The mean filter were applied with a 5×5 kernel, and the thresholding were set to a binary thresholding with a threshold value of 127. This made it possible to detect the features from the image contain a lumination value above 127. The best detection method found were using the feature detector in the ORB detector, implemented in the computer vision library OpenCV. The feature detection created a set of keypoints in the image. The keypoints were transformed to a spacial map using the range indicator on the scanned sonar image. The path were generated by the use of Voronoi Diagram. The Voronoi Diagram created a web of feasible path segments. The path were then chosen using a Dijkstra Algorithm, were the weight of each path segment were set to their lengths.

Four simulations were conducted based on four different images from the sonar video stream. The simulations included following the generated path. The ROV, in general were able to follow the paths created well, with small deviation between the desired state and the estimated state. However, some of the paths contained changes in path direction that resulted in a deviation of up to 7 meters between the desired and estimated position. This were due to the large differences in path direction as a result of using Voronoi Diagram combined with a relative small transit length, resulting in paths that were to small in scale to follow perfectly.

The results show some promise. The computational time for the processing and detection were found to be low, thus making the application feasible for online use. Furthermore, the feature detection detected the features when the thresholding were done correctly. However, difficulties regarding choosing the threshold value for the image thresholding, resulted in features that were not detected, and features of no interest that were detected. Furthermore, there was difficulties in the implementation of the filtering that transform the detected keypoints in to input points for the Voronoi Diagram. Some of the features detected resulted in two or more input points, yielding false feasible path segments in the Voronoi Diagram.

The concept of underwater path planning using sonar imaging can have some promise. The computational time is low, giving it online capabilities. In the case of the multiple smaller obstacles in the area in front of the ROV, the use of Voronoi path planning could be feasible. Combined with a Multibeam FWL sonar with longer range, returning larger scanned zones, the methods proposed here could be applied.

9.1 Further Work

To be able to implement this on a field mission, several improvements needs to be conducted. Firstly, the path planning should be implemented with a fail safe in a form of obstacle detection during the transit. As the method proposed in this thesis only process the sonar image before the transit, an obstacle detection algorithm that ensures no collision during the path following should be included. Secondly, I would propose a way of ensuring that the path planner actually returns safe paths. The Voronoi diagram as a path planner is only as reliable as the information about the obstacles are. If the obstacles are indeed larger than the Voronoi Cells, the feasible path segments, or edges, are no longer feasible. Furthermore, to ensure the best paths a cost function could be created, both in regards of distance to obstacles and the turning rate needed for the ROV. This would ensure safer paths, and more feasible paths for the ROV to follow.

Bibliography

[1] Fredrik Dukan. *ROV Motion Control Systems.* PhD thesis, Norwegian University of Science and Technology, Department of Marine Technology, 2014.

[2] Ida Rist-Christensen. Autonomous robotic intervention using rov, 2016.

[3] Claudia S. Huebner. Evaluation of side-scan sonar performance for the detection of naval mines. `https://ets.wessexarch.co.uk/wp-content/uploads/2011/05/03-SidescanSonarSurvey.jpg`, 2018. [Online; accessed 10.05.19].

[4] Xavier Lurton. An introduction to underwater acoustics : principles and applications, 2002.

[5] Utkarsh Sinha. Fundamentals of Features and Corners. `http://aishack.in/tutorials/harris-corner-detector/`, 2018. [Online; accessed 2019-5-1].

[6] Chris Harris and Mike Stephens. A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, pages 147–151, 1988.

[7] Bin Fan. Local image descriptor: Modern approaches, 2015.

[8] Paul Herron. Furthest Point Voronoi Diagrams. `http://www.dma.fi.upm.es/personal/mabellanas/tfcs/fvd/main.html`. [Online; accessed 2019-2-10].

[9] Offshore Support Journal. Resident robots could save money and remove ships from the equation. `https://www.osjonline.com/news/view,resident-robots-could-save-money-and-remove-ships-from-the-equation_51339.htm`, 2018. [Online; accessed 10-November-2018].

[10] R.D. Christ and editors R.L. Wernli. *The ROV Manual (Second Edition).* Butterworth-Heinemann, Oxford, 2014.

[11] Committee on Autonomous Vehicles in Support of Naval Operations. *Autonomous Vehicles in Support of Naval Operations.* The National Academies Press, Washington, 2005.

[12] Asgeir J. Sørensen, Ingird B. Utne, and Ingrid Schjølberg. Autonomous vehicles in support of naval operations. Technical report, NTNU AMOS, Department of Marine Technology, 2017.

[13] Mauro Candeloro. *Tools and Methods for Autonomous Operations on Seabed and Water Column using Underwater Vehicles.* PhD thesis, Norwegian University of Science and Technology, Department of Marine Technology, 2016.

[14] Mauro Candeloro, Anastasios M. Lekkas, Asgeir J. Sørensen, and Thor I. Fossen. Continuous curvature path planning using voronoi diagrams and fermat's spirals. *IFAC Proceedings Volumes*, 46(33):132–137, 2013.

[15] M. Candeloro, A. M. Lekkas, J. Hegde, and A. J. Sørensen. A 3d dynamic voronoi diagram-based path-planning system for uuvs. *OCEANS 2016 MTS/IEEE Monterey*, pages 1–8, Sep. 2016.

[16] Mauro Candeloro, Anastasios M. Lekkas, and Asgeir J. Sørensen. A voronoi-diagram-based dynamic path-planning system for underactuated marine vessels. *Control Engineering Practice*, 61:41–54, 2017.

[17] Y Petillot, I Tena Ruiz, D.M Lane, Y Wang, E Trucco, and N Pican. Underwater vehicle path planning using a multi-beam forward looking sonar. In *IEEE Oceanic Engineering Society. OCEANS'98. Conference Proceedings (Cat. No.98CH36259)*, volume 2, pages 1194–1199 vol.2. IEEE, 1998.

[18] Enric Galceran, Vladimir Djapic, Marc Carreras, and David P Williams. A real-time underwater object detection algorithm for multi-beam forward looking sonar. *IFAC Proceedings Volumes*, 45(5):306–311, 2012.

[19] Tsz Kit Chiu. Sonar tracking and obstacle avoidance for navigation of rov. Master's thesis, Norwegian University of Science and Technology, Department of Marine Technology, 2018.

[20] G.L Foresti, V Murino, C.S Regazzoni, and A Trucco. A voting-based approach for fast object recognition in underwater acoustic images. *IEEE Journal of Oceanic Engineering*, 22(1):57–65, 1997.

[21] E. Trucco, Y.R. Petillot, I.Tena Ruiz, K. Plakas, and D.M. Lane. Feature tracking in video and sonar subsea sequences with applications. *Computer Vision and Image Understanding*, 79(1):92–122, 2000.

[22] Ørjan Grefstad. Development of an obstacle detection and avoidance system for rov, 2018.

[23] Richard P Hodges. Underwater acoustics : analysis, design, and performance of sonar, 2010.

[24] Robert J Urick. Principles of underwater sound.

[25] T. I. Fossen. *Handbook for Marine Craft Hydrodynamics and Motion Control.* John Wiley Sons, Ltd, Chichester, 2011.

[26] Daniel de Almeida Fernandes. *An output feedback motion control system for ROVs - Guidance, navigation, and control.* PhD thesis, Norwegian University of Science and Technology, Department of Marine Technology, 2015.

[27] Eirik Bjørklund Holven. Control system for rov minerva 2. Master's thesis, Norwegian University of Science and Technology, Department of Marine Technology, 2018.

[28] Morten Breivik and Thor Fossen. *Guidance Laws for Autonomous Underwater Vehicles.* 01 2009.

[29] Srdjan Stanković. Multimedia signals and systems, 2012.

[30] Jennifer Bourn. Color Meaning: Meaning of The Color Green. `https://www.bourncreative.com/meaning-of-the-color-green/`, 2011. [Online; accessed 2019-5-2].

[31] Rafael C Gonzalez. Digital image processing, 2002.

[32] OpenCV. Image Thresholding. `https://docs.opencv.org/3.4.3/d7/d4d/tutorial_py_thresholding.html`, 2018. [Online; accessed 2019-4-12].

[33] Linda G Shapiro. Computer vision, 2001.

[34] Deepanshu Tyagi. Introduction to ORB. `https://medium.com/@deepanshut041/introduction-to-orb-oriented-fast-and-rotated-brief-4220e8ec40cf/`, 2019. [Online; accessed 2019-4-25].

[35] Antonios Tsourdos, Brian A. White, and Madhavan Shanmugavel. *Cooperative Path Planning of Unmanned Aerial Vehicles*. John Wiley Sons, Ltd, Chichester, 2011.

[36] Brian Dillinger, Ronald Jenkins, and Joshua Walton. Automated waypoint generation with the growing neural gas algorithm. *Association for the Advancement of Artificial Intelligence*, 8:175–191, 1961.

[37] Sanjeev Sharma and Matthew Taylor. Autonomous waypoint generation strategy for on-line navigation in unknown environments. 10 2012.

[38] Ana Kuzle. Voronoi diagrams. `http://jwilson.coe.uga.edu/EMAT6680Fa08/Kuzle/Math.html`. [Online; accessed 2019-1-18].

[39] An implementation of fortune's algorithm in python. `https://github.com/Yatoom/voronoi`, 2018. Accessed: 18-04-19.

[40] Python implementation of dijkstra's shortest path algorithm. `https://github.com/joyrexus/dijkstra`, 2014. Accessed: 18-04-25.

Tore Øystese

Master Thesis at NTNU