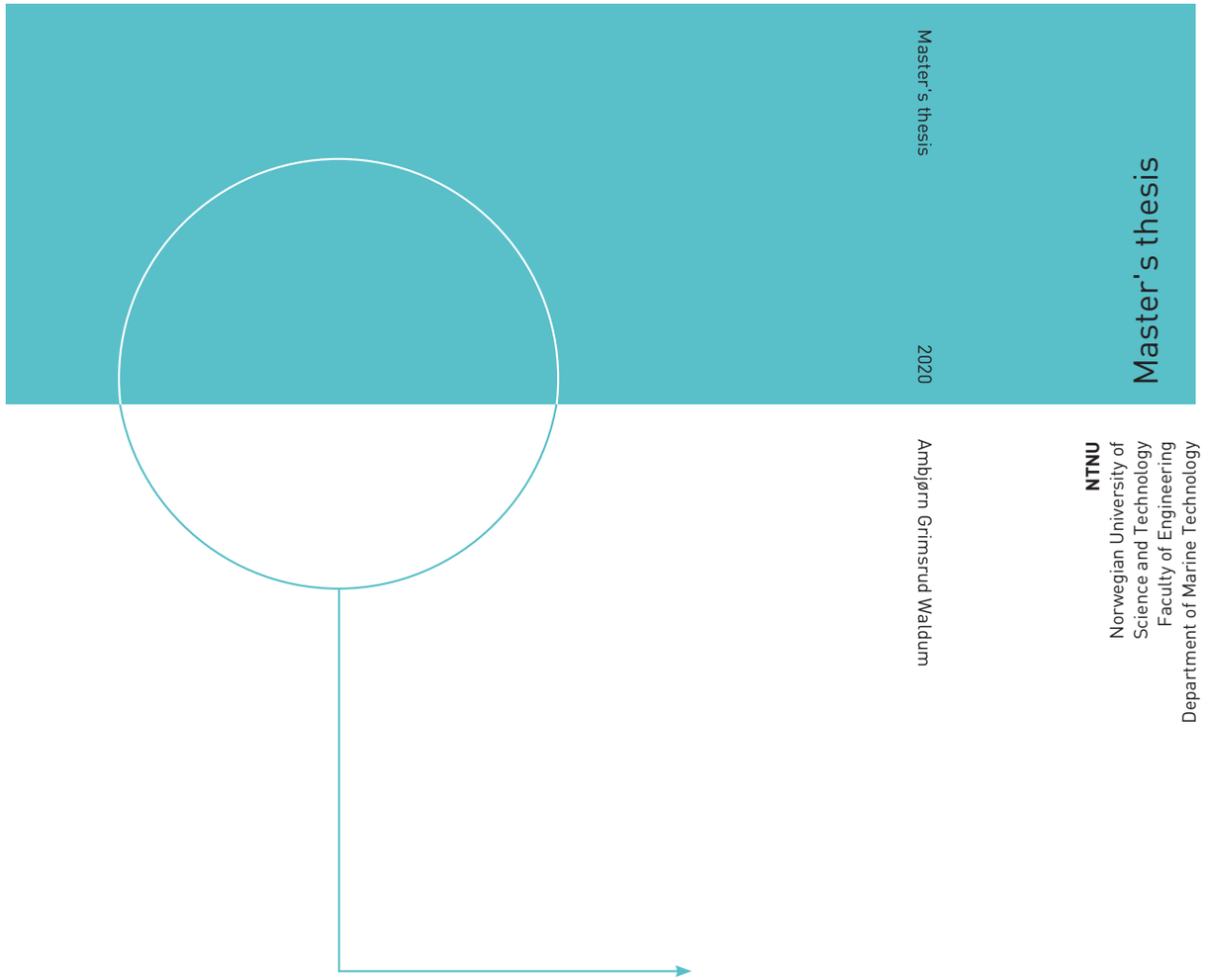


Master's thesis

2020

Ambjørn Grimsrud Waldum

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

Ambjørn Grimsrud Waldum

Sonar EKF-SLAM and mapping in an structured underwater environment

June 2020

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

NTNU

Norwegian University of
Science and Technology

Sonar EKF-SLAM and mapping in an structured underwater environment

Ambjørn Grimsrud Waldum

Marine Technology
Submission date: June 2020
Supervisor: Martin Ludvigsen

Norwegian University of Science and Technology
Department of Marine Technology

Spring 2020

Sonar EKF-SLAM and mapping in an structured underwater environment

Ambjørn Grimsrud Waldum

NTNU
Faculty of Engineering
Department of Marine Technology

Main supervisor: Martin Ludvigsen
Department of Marine Technology, NTNU
June 23, 2020

Master contract

Underwater SLAM has shown potential, but only relying on a camera feed might not be reliable as for example occlusion, lighting conditions and particles in the water might hinder performance. In addition the problem with finding a good camera model to correctly model radial and tangential distortions makes 3D-reconstruction difficult. Yet the high resolution data a camera provides is very desirable, and can also be used for object detection in addition to navigation.

Another much used underwater sensor is the sonar. If the necessary data from the water column to calculate the speed of sound is collected, a sonar can be used to give depth to objects it the scene more accurately than a camera. There are however harder to match features in a forward pointing sonar than in a camera, especially with regards to loop closure. A combination between camera and sonar might provide the best of both worlds, where a camera can be used for loop closure, and more detailed matching, while the sonar can be utilized to improve the 3D-reconstruction by offering more accurate ranges to objects than a camera would be able to in an underwater environment.

In this thesis I want to explore the possibilities of sonar slam, and eventually fusing a sonar with a camera. Firstly in a simulated world in Gazebo, and thereafter on a AUV with a forward mounted monocular camera and a forward pointing sonar. I plan to approach this project in three stages, and not go onto the next stage before the implementation of the previous stage acceptable.

Stage 1: The first goal is to use the sonar and the camera to create a 2D reconstruction of the environment that should be displayed in a grid. The grid can be used as the base for mission and path planning. The challenges here is to match features from the sonar to the camera. In addition to match features I would also like to use YOLOv3 for object detection, recognize certain objects in the environment and place them in the 2D-grid with range given by the sonar measurements.

Stage 2: The second goal is to implement a SLAM algorithm in 2D that utilizes the map from stage 1. In addition I want to implement some sort of mechanism to handle the problem of lost tracking when there are not enough features to match, as often happens in the plain versions of popular methods like ORBSLAM.

Stage 3: The third goal is to go from a 2D-grid to a 3D-world, It will greatly complicate the problem, and it is highly unlikely that it will be time to actually implement a working program. However 3D mapping will be more appropriate for an AUV that is on a mapping mission, and is therefore of more interest than a 2D grid. Therefore I would like to do some research on how to go from the 2D to 3D problem, and suggest future work that can be done to achieve this goal.

Abstract

One of the major problem of autonomous underwater vehicles(AUV) today are navigation for substantial periods of time without resurfacing to correct positional estimates with the Global Navigation Satellite System(GNSS). One technique that may be able to solve this problem is simultaneously localization and mapping(SLAM). There have been multiple underwater SLAM methods proposed with varying results. One of the challenges in the underwater environment is the necessity to use sonars instead of cameras and lidars due to their limitations in an underwater environment. The challenge with sonar is to find a robust feature extraction as objects imprint on the scan changes a lot based on point of view. In this report an EKF-SLAM algorithm utilizing a forward pointing imaging sonar is implemented and compared to a state-of-the-art EKF. The algorithm takes advantage of an modified hough-transform to extract line features from the sonar scans which are used as features. The dataset the algorithms are tested on was collected during three different experiments in an indoor pool examining how the two navigation methods compared in linear-motion, heavy rotations and over a 20 minute period. The results showed that the EKF-SLAM had performance on par with the EKF during linear-motion and heavy rotations. However, on the 20 minute bag the EKF-SLAM algorithm was able to almost completely eliminate the drift that occurs over time in the EKF. In that experiment the EKF ended up having an absolute positional error of 2.5 meters after 20 minutes, while the absolute error of the EKF-SLAM was below 0.5 meters. In addition to the state-estimation done by EKF-SLAM it also locates landmarks that can be placed into a map. In this report an experimental mapping between camera pixels and sonars by fitting a dataset to a polynomial are proposed. The purpose of this mapping is to insert objects found in a camera frame into the map created by the SLAM algorithm. A prototype of the mapping was tested in a simulated world. The test gave good results, and future development could be considered.

Sammendrag

En av de store utfordringene for autonome undervans farkoster idag er navigering i lengre perioder uten og gå opp til overflaten for og korrige posisjons estimatione ved hjelp av GNSS. En ny teknikk som kan være i stand til og løse dette problemet er direkte oversatt til norsk simultan lokalisering og kartlegging(SLAM). Det har vært flere undervanns SLAM metoder foreslått med varierende resultat. En av utfordringene i et undervanns miljø er nødvendigheten av og benytte sonar istedenfor kamera eller lidar som fungerer dårlig i vann. Selve utfordringen kommer fra at det er vanskelig og finne robuste landemerker med sonar da utsende på et objekt vil variere mye basert på synsvinkelen. I denne rapporten er en EKF-SLAM algoritme som benytter en fremover pekende bilde sonar implementert og sammenlignet med en state-of-the-art EKF. Algoritmen benytter en modifisert Hugh Transform for og hente ut linjer fra sonar skannene som blir brukt som landemerker. Datasettene algortimene blir testet på er samlet under tre forkjellige eksperiment i et innendørs basseng, og tar for seg hvordan de to metodene håndterer lineær bevegelse, mye rotasjon og estimering over en 20 minutters periode. Resultaet viste at EKF-SLAMen og EKF fungerte like bra som hverandre under linær bevegelse og under mye rotasjon. Det var derimot merkbar forkjell under 20 minutters testen, der klarte EKF-SLAM og eliminere det meste av driften EKF algoritmen bygde opp. I dette eksperimentet endte EKF en opp med en absolutt error på 2.5 meter etter 20 minutter, mens EKF-SLAM sin absolutt error var på under 0.5 meter. I tillegg til state estimeringen EKF-SLAM gjenomfører finner den også landemerker som kan settes inn i et kart. I denne rapporten blir det forslått et eksperiment for og direkte konvertere camera-pixler til sonar vinkler ved og samle et dataset og så tilpasse et tredje grads polynom til datasettet. Målet med denne mappingen er og sette inn objekter som blir funnet i kamera bilde inn i kartet som blir skapt av SLAM algoritmen. En prototype av denne mappingen ble implmentert og testet i en simulert verden. Denne testen ga positive resultater, og vidre utvikling kan vurders.

Preface

Vortex NTNU is a student organization which provides students with the opportunity to partake in the creation and development of an AUV. I have been part of this organization for two years, and the motivation for developing the system presented in this report comes from the experience gathered during the summer of 2019 where Manta, our AUV, competed in Robosub. Here the shortcomings of some of our solutions became clear and the system developed in this report tries to face some of these issues.

The aim of this report has been to implement navigational and spatial awareness solutions for Manta. By utlizing an EKF-SLAM system instead of a regular EKF the hope is to be able to have a navigational system that do not drift. In addition adding spatial awareness by building a map would provide a platform on which the mission- and path planning algorithms can use as basis for their decisions.

I would like to give a special thanks to Professor Martin Ludvigsen for guidance, support and putting the trust in me to be able to implement this system. Also for for being able to secure us a time-slot in the marine cybernetics laboratory during a time when NTNU was in lock-down. I would also like to thank my two fellow graduate students Øyvind Denvik and Erlend Vollan which I conducted experiments with at Tyholt. My gratitude also goes to everyone at Vortex NTNU which made it possible to write this thesis. A special thanks goes to Børge Pahlm, Nabil Ha and Alexander Thoresen for their help getting Manta operational in time for the experiments during lock-down, and Wai-Yen Chan and Silje Susort which ran the training of the neural network used in this report.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

Autonomous underwater vehicles(AUVs) are unmanned submersible vehicles, which can operate independent of an operator. It is an emerging technology which is in growing demand in the off-shore industry. For the most part AUV technology is still in an early stage, and much of the experiments done with AUVs today are for research and not commercial purposes. However, the potential for financial gain by replacing tasks today performed by remotely operated vehicles(ROVs) with AUVs is huge. Not only financially for the companies, but the reduction in emissions would be good for the sea and environment as a whole. In order for this transition to take place there are multiple complex problems that must be addressed. In this report the problem of the AUV accurately having to estimates its own position is contemplated.

Unlike vehicles operating on the land or in air which can rely on the Global Navigation Satellite Systems(GNSS) for positional estimations, the properties of water makes an underwater vehicle unable to utlize this powerful tool. While there are acoustic positioning systems like ultra-short baseline(USBL) and long baseline(LBL) that allows for accurate position estimation they require heavy infrastructure, USBL requires a ship to be present with specialised equipment, and LBL requires acoustic beacons to be fixed on the sea floor in the area of operation, and do only work locally in a small area. The positional estimation is therefore mostly done by the introspective sensors present on the AUV, which as a result of bias and errors in the sensor readings unavoidably will lead to the estimation drifting from the true position.

One promising alternative to positional estimation purely based on introspective sensors for land vehicles that do not utilizing GNSS is Simultaneously localisation and mapping(SLAM). In addition to the introspective sensors SLAM utilizes extrospective sensors to update the position of the vehicle based on change in its position with regards to the surroundings. Today SLAM is a hot research topic in robotics, and it is implemented into commercial robotic products from Roombas to driverless cars.

A functioning underwater SLAM system could be a solution for more robust AUV navigation. Sonars which are a sensor carried by most AUV, and are already in many cases used for mapping the seabed, could be used as the extrospective sensor in a SLAM system. One of the challenges with utilizing a sonar for SLAM is the difficulty of finding good features as they lack the amount of information found in an image taken by a camera, or the point-cloud extracted from a lidar. The shape of an object viewed through a sonar scan will vary greatly based on the point of view of the sonar, which makes it hard to recognize patterns. Despite this there have been research into sonar SLAM, and in this report one suggested sonar SLAM approach will be implemented and tested in an indoor pool.

1.2 Contribution of this thesis

The overall goal for this thesis was to implement a 2D-navigation system for AUVs utilizing a sonar, that simultaneously estimates the states of an AUV through an EKF-SLAM, while publishing a map that can be used for path- and mission planning. In addition to obstacles the map should also contain the positions of objects of interest.

The EKF-SLAM system implemented utilizes the line feature extraction proposed in [27] and the overall system structure is heavily inspired on the EKF-SLAM equations presented in chapter 11 of [4]. The

resulting state-estimation under different scenarios are compared to the popular EKF implementation found in [21].

The map to be published will be based in the landmarks found by the SLAM algorithm. To insert objects of interest into the map an experimental approach mapping camera pixels directly to sonar angles is prototyped. The idea is to find objects of interest in a camera frame with a CNN, and then transform its position into a sonar angle. By finding the bins in the sonar scan at the given angle that most likely correlates with an object its position can be inserted into the map.

The two main contributions of this thesis can therefore be summed up as the comparison between an underwater EKF-SLAM algorithm an a standard EKF, and the prototyping of an mapping approach between camera pixels and sonar angles.

1.3 Motivation

As AUVs today mostly do observation mission navigational accuracy is critical. Data collected of the seabed by a sonar or camera is useless if one is not able to tell where the data is collected from. In theory SLAM should be an excellent alternative to resurfacing that would save time. Its especially useful that sensors that could perform SLAM is already present on most AUVs. In order to get underwater SLAM to the point it needs to be to replace GNSS implementation and real world testing is important. Seeing how the different feature extractors and methods actually perform in the real world, and discover new scenarios that would not arise in a simulated case. This is very important to gain experience that would allow us to take the technology future. This is the motivation for the first part of this thesis.

If AUV is to fully replace ROVs better pose estimation is not enough, in addition spatial awareness will be critical when manoeuvring in structured environments. This is the motivation for the second part of this thesis, which tries to create a map of the environment, and place objects of interest within this map.

The thought behind choosing EKF-SLAM over Graph-SLAM. There are two popular SLAM algorithms, the classical filter SLAMS and the more modern Graph-SLAM approaches. The concern with filter SLAM is the increasing size of the dense covariance matrix, which eventually will lead to a lot of computations when a many landmarks are added to the state vector. Therefore Graph SLAM is at the moment the most popular approach for visual and lidar SLAM. However in this implementation only lines with a substantial amount of matching bins are extracted from the sonar measurements, and few if any lines will be detected in each scan. Hence the amount of features added is much smaller than the amount that would have been extracted from a lidar or camera. If an operation last long enough for the amount of landmarks to become a hindrance a solution is to utilize multiple local maps as proposed in [27]. Therefore the normal downsides with an EKF-SLAM approach should not be a issue here.

1.4 Outline

In chapter 2 of this report all theory used in the implementation of the SLAM system are presented. First the basic theory of acoustics, underwater vehicles and sensors are quickly reviewed before a more thorough explanation of the state-estimation techniques relevant for the report is presented. At the end of the chapter there is a review of Yolov3. Chapter 3 starts by introducing the reader to the hardware and software architecture of the AUV before the software implemented in this report is reviewed. Chapter 4 presents the experiments that were conducted. Chapter 5 presents the results of the experiments. Chapter 6 contains the discussion of the results, and finally in Chapter 7 a conclusion is drawn based on the results, and future work is suggested.

Chapter 2

Theory

In this chapter the theory of physical phenomena, sensors and estimation techniques that applies to this thesis will be introduced.

2.1 Acoustics in water

Underwater acoustics is the study of how sound propagate in water. From the first world war when the first sonars were developed by the British to the present day underwater acoustics have played a central role in underwater communication, localisation and observation. Radio- and light-waves that are used to solve similar problems in air have a very limited range in water due scattering, absorption and reflection, which makes acoustics preferable.

Different frequencies goes through different materials, and sound with frequencies between around 10Hz to just above 1MHz are typical for underwater acoustics. Higher frequencies would lead to to much abortions, like radio- and light-waves, while a lower frequency might lead to undesired penetration deep into the seabed. Higher frequencies yields higher resolutions, while lower frequencies gives longer range.

2.1.1 Sound Velocity

In order to get an accurate estimations from an acoustic sensor, it is essential to to have a good estimation of the speed of sound. The speed of sound is influenced by temperature, pressure and salinity. The general equation is given by 2.1.

$$c = \sqrt{\frac{E}{\rho}}, \tag{2.1}$$

Here E is the volume stiffness and ρ is the density of water. These two factors are dependent of the ambient conditions of the sea. Temperature have the biggest impact with a change of $10°C$ approximately corresponding to a 4 m/s change in the speed of sound. If the sound wave is expected to go through layers of water with different conditions, for example an echo-sounder mounted to a surface vessel recording a scan of the sea bed, temperature, salinity and pressure data of the entire water column should be collected and used when analyzing the scans. If the speed of sound is not estimated correctly the resulting estimate of the acoustic sensors will be distorted.

2.1.2 The Doppler effect

A very useful phenomena is the Doppler effect. The Doppler effect is the change in frequency of incoming waves to an observer who is moving relative to the wave source. The usual example is the pitch of an ambulance driving past an observer. Waves arriving with a higher frequency when the ambulance are moving towards the observer, and less frequent when moving away from the observer. Figure2.1 visualises the effect.

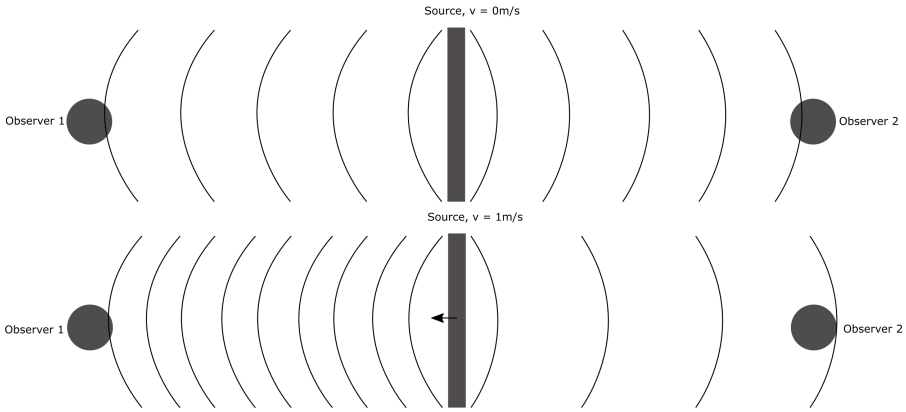


Figure 2.1: Figure shoing frequency of sound waves arriving at two observer when the source is staying still, and moving towards the left

The Doppler effect is much used in underwater robotics, both as a tool for navigation, using a Doppler velocity logger(DVL) for state estimation, and target tracking using the Doppler shift to see if targets are approaching or withdrawing from the observer. Assuming that the source is directly approaching or withdrawing from the observer the frequency recorded by the observer can be calculated as seen in equation 2.2.

$$\frac{f}{v_{wo}} = \frac{f_0}{v_{ws}} = \frac{1}{\lambda} \tag{2.2}$$

where f is the observed frequency, f_0 the frequency emitted by the source, v_{wr} is wave velocity relative to the observer, v_{ws} is the wave velocity relative to the source and λ is the wave length.

2.1.3 Reflections and Shadow Zones

When a sound wave hits a solid obstacle it will be reflected in various directions, because of this an observer may receive multiple readings of the same ping. In figure 2.3 spread A is propagated directly from the source to the receiver, while spread B is reflecting of the sea bottom. Here the sound wave propagating directly from the source to the receiver will arrive first, while the reflection will arrive later. This effect must be taken into consideration when working with acoustic data. The reflection of the bottom and the longer travel distance will usually drain more energy from the beams arriving at a later time as can be seen from the plot in the figure.

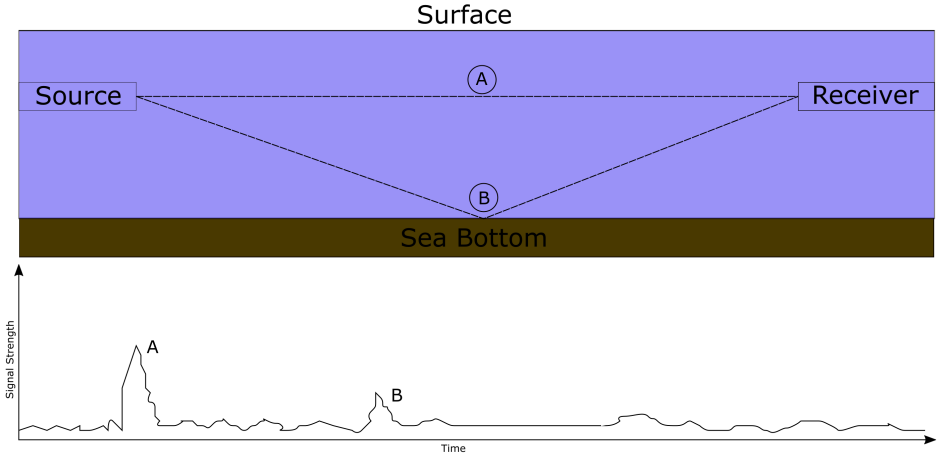


Figure 2.2: Shows two different paths from source to receiver

If there are large obstacles in the environment where the acoustic images are recorded there might be zones behind the obstacles the sound waves wont reach. These zones will appear as low intensity in the acoustic image, and can not be used to extract useful information about the area. They are often referred to as shadow zones. In environments where the waves are free to travel over long distances shadow zones may also occur as a result of ray bending. Ray bending occurs because rays are bent

towards the lowest velocity. Most often the speed of sound decreases with depth making the sound waves bending downwards. Both phenomena can be seen in figure 2.3.

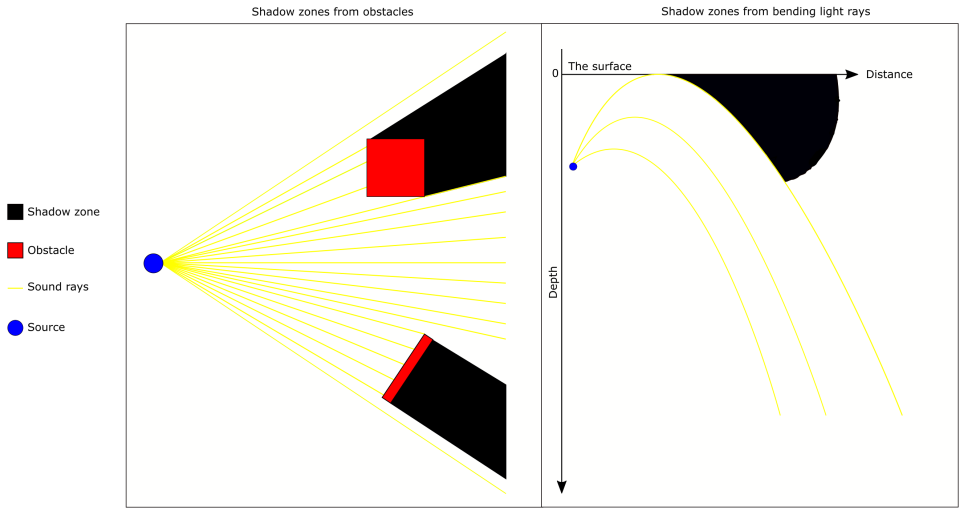


Figure 2.3: Shows shadow zones created by obstacles and bending light rays

2.1.4 Acoustic Disturbances

There are at all times sound in the sea which can disturb the acoustic systems. They can be categorised as following:

1. *Ambient noise* - This is the noises from the surroundings, like wind, waves, rain and animal noises mixed together with man-made noises like shipping, industrial activity and so forth.
2. *Self-noise* - The noise the underwater acoustic system makes itself, like from radiated noise, flow noise, electrical interference or thermal noise from the system's own electronics.
3. *Reverberation* - This sort of noise only affects active sonar systems and is caused by echoes generated by the sonar's own signals.
4. *Acoustic interference* - The noise is generated by other acoustic systems nearby, likely from the ship or underwater platforms, and sometimes from sources farther away.
5. *Acoustic mirroring* - If acoustic waves are hitting a plane surface, like a wall, it might create a mirror effect making objects appear multiple times in the acoustic image.

These noises can have a major impact on the retrieved acoustic images and will have to be taken into consideration when it is processed.

2.2 Sensors

2.2.1 IMU

An inertial measurement unit(IMU) is a combination of an accelerometer and a gyro. It measures the acceleration in three perpendicular axis and the angular velocity of the three axes. It is usually a critical part of a navigation system, especially in systems that perform dead-reckoning or for other reasons operates without the Global Navigation Satellite Systems(GNSS) over substantial periods.

Conceptually an accelerometer can be thought of as a damped-spring-mass system, where the mass will be in equilibrium when it feels zero forces. When an acceleration happens the mass will be displaced and the acceleration can be measured from the displacement in the spring. However modern IMUs are usually very small systems, called micro electro-mechanical systems(MEMS). They usually consist of a proof mass and cantilever beams, cantilever beams are beams anchored in only one end. By comparing the conductance in beams connected to the proof mass with free beams the displacement can be measured, and hence the acceleration. When using a accelerometer on earth is is important to take into consideration that gravity always will pull on the mass, and therefore the acceleration of $9.81 m/s^2$ always till be present divided among the three axis based on how the current orientation of the IMU is.

In AUVs measurements from an IMU is often combined with measurements from a DVL in a filter used for internal state estimations.

2.2.2 DVL

A DVL is an acoustic sensor that measures relative velocities between the water and/or seabed and the DVL. It works by emitting multiple sound beams, and recording the echo like a sonar using the time the signal travelled to measure distance, and the doppler shift in the signal to measure velocity.

For navigation of AUVs the most useful property of the DVL is bottom tracking. After the DVL have identified where the bottom are by looking at the returned signal, it can concentrate only on measurements from this range and ignore the rest of the water column. This allows the AUV to get measurements of its velocity relative to the seabed in three dimensions. If the seabed is assumed fixed in the navigation frame of reference the relative velocity would be the AUVs velocity directly. It is important to note that the DVL must send at least three beams is in order to measure velocity in three dimensions.

In addition to finding velocity the DVL is able to measure the elevation of the AUV over the seabed by measuring the time between the ping is emitted and the echo from the seabed is received. This allows an AUV with a DVL to operate at a fixed elevation above the seabed.

2.2.3 Sonar

Sonar(Sound Navigation and Ranging) is an acoustic sensor that uses sound propagation to detect the environment. A sonar typically consists of four main parts: A transmitter, a transducer, a receiver and a control unit. The transmitter produces an oscillating electric field which is converted by the transducer to create a vibration. To do this, a piezoelectric material is often used. A piezoelectric material can turn oscillating electric fields into mechanical vibration by deformation when affected by energy.

This also works the other way around whereas acoustic waves deform the piezoelectric element and generate an electric signal to be analyzed. The receiver detects and amplifies the relatively weak electric signal from the transducer before it is sent to the control unit. The control unit records the data, calculates ranges based on the speed of sound and displays a picture of the underwater surroundings, see 2.4

There are many kinds of sonars, but most can be divided into either a passive or an active sonar.

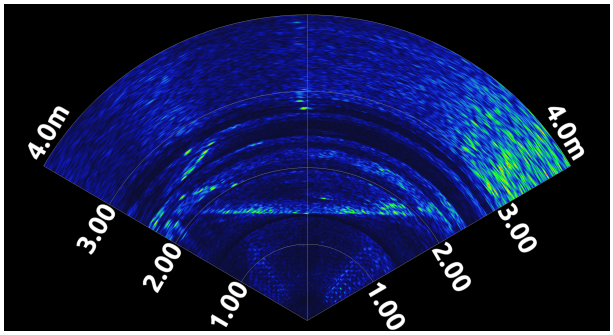


Figure 2.4: Sonarscan taken by Tritech 720i-imageing sonar of a wall about 2m in front of the sonar head

2.2.3.1 Active and passive sonars

A passive sonar is an instrument that listens for sound waves in the water. They were invented during World War 1 to listen for nearby submarines. While passive sonars can be used to find sound sources, and if the sound waves are not to bent in the water what sector the sound most likely emerged for it has no information about the time of travel, and therefore can not accurately locate the source. While passive sonar has it uses in military and research application, active sonars often play a more interesting part in navigation and localisation.

Active sonars takes a more "active" part, emitting their own sounds waves and listening for the echo. By emitting the wave in multiple directions at once one get a scan of the environment. Objects that intersect the sound wave will reflect large parts back to the sonar. By looking at the angle of the reflected signal, and its intensity it is possible to calculate where objects most likely are with regards to the sonar position.

In active sonars the data is usually divided into beams, were each beam consists of multiple bins. A beam corresponds to a set angle interval in the sonar sector, and its bins are the intensity of sound recorded at that angle during sampling, see figure 2.5. The further away from the sonar a bin is recorded, the larger area it covers as the sound propagates through the water. In order to get an accurate visualisation of a sonar scan as can be seen in figure 2.4 the varying size of each bin with distance must be taken into consideration.

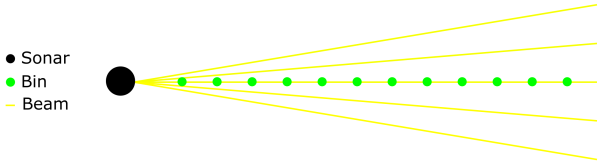


Figure 2.5: A sonar with 5 beams, the bins of the middle beam are visualised.

As sound waves are spread and absorbed while travelling through water they become weaker while propagating away from the source. In an active sonar this could become a problem as a wall close to the source would yield higher intensity than an exact replica of the wall placed further away. To combat sonars usually utilizes an adaptive gain. Such a gain will grow during a scan, and therefore amplify the received signal from far away objects more than those close to the source.

Different objects consisting of different materials will create different echos, there have been research done into identifying what objects correlates to different objects utilizing a convolutional neural network(CNN)[6]. Even though this is as of yet still inn a early research phase, one key take away is that man-made objects will have different response than natural objects. This is mostly because man-made objects often consists of clear geometric shapes, while natural objects have random shapes. This characteristic opens up the possibility of creating underwater SLAM systems that could utilise the clear features created by man-made objects to more accurately localize AUVs operating in structured environments.

2.2.4 Camera

A camera is an optical sensor used to record images. It usually consists of a lens, a sensor and the mechanical housing. In simple terms the lens is used to concentrate the light rays that are reflected of objects in the scene onto the camera sensor. The sensor is light sensitive and will record the intensities of the light when hit. By taking a recording of the state of the sensor at an instance the frame is recorded. In

older mechanical cameras that used a camera film as recording material the length between lens and film had to be manually adjusted to get objects into focus. However with the introduction of digital cameras the sensor is electrical and the correct focus can be calculated digitally instead of manually adjusting the camera. The result is an information rich image of the scene the camera is pointed towards.

Since cameras provide high resolution data of the environment they are very popular in robotics that have to interact with a complex environment, as the data is excellent as feedback for the system. This is especially true considering the great improvements in computer vision thanks to leap in computational power over the last decade.

2.2.4.1 The perspective camera model

A perspective camera model model describes the relationship between points in a 3D world and pixels on an image. The hardware of the camera, like the lens, shutters and distances in the camera will alter thees parameters to be unique for every camera. A perfect camera model will be a model that perfectly maps pixels to their points in 3D space. For the pinhole-model which is one of the established perspective camera models this relationship can be written in matrix form as

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & s_w & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} R_{3x3} & t \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \boldsymbol{\Omega T X} \tag{2.3}$$

Where $\boldsymbol{\Omega}$ is called the intrinsic camera matrix. Here f_x and f_y is the focal length in x and y-direction respectively. c_x and c_y is the centre of projection in x and y-direction respectively. Those are usually close to the centre of the image. s_w is the skew parameter, it explains shear distortion of the axis, but are for most models set to 0. In this rapport $s_w = 0$ is assumed. $\boldsymbol{T}$ is the eccentric parameters, it consists of a rotation matrix R_{3x3} and a translation parameter t. s is scale, u and v are pixel coordinates in x and y-direction respectively and $\boldsymbol{X}$ is the corresponding coordinates in a 3D world. This relationship allows for structure of the scene to be conserved, which can be utilised as a powerful tool in navigational problems. However reality are never as simple as it appears on paper, and a pure pin-hole model is a naive approach when the goal is to do 3D reconstruction in air, and especially if you want to use your system in water.

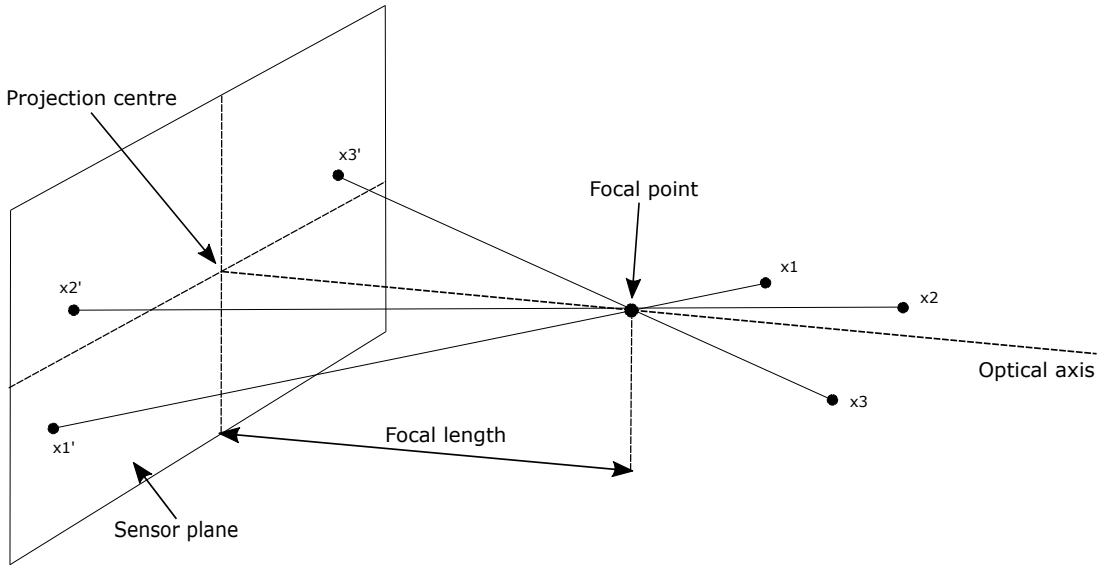


Figure 2.6: Parameters of a perspective model

Figure 2.6 visualises f_x, f_y, c_x and c_y from the intrinsic matrix Ω. From the figure one can see that a small focal length yields a lager field of view, but less detail and lager focal length yields a small field of view, but more details.

2.2.4.2 Distortions

Due to imperfection in lenses and light-reflection distortions will occur. The two major distortions in cameras is radial- and tangential distortions. The radial distortions make straight lines in reality occur like curves in the image frame. The farther away from centre of projection the bigger the effect. There are many models trying to correct for radial distortions, and they are usually approximated as a polynomial[18]. There are no consensus about how many terms to include in the polynomial.

$$\delta r = k_1 r^3 + k_2 r^5 + k_3 r^7 ... \tag{2.4}$$

Tangential distortions occur when the image frame and camera lens is not parallel. It will distort the image point in a circle around the projections centre point, and can be approximated for the x- and y-direction as follows:

$$2p_1 uv + p_2(r^2 + 2u^2) \tag{2.5}$$

$$p_1(r^2 + 2v^2) + 2p_2 uv \tag{2.6}$$

Since it is not possible for a lens to be perfectly parallel to the image frame in reality, some degree of tangential distortion will always be present.

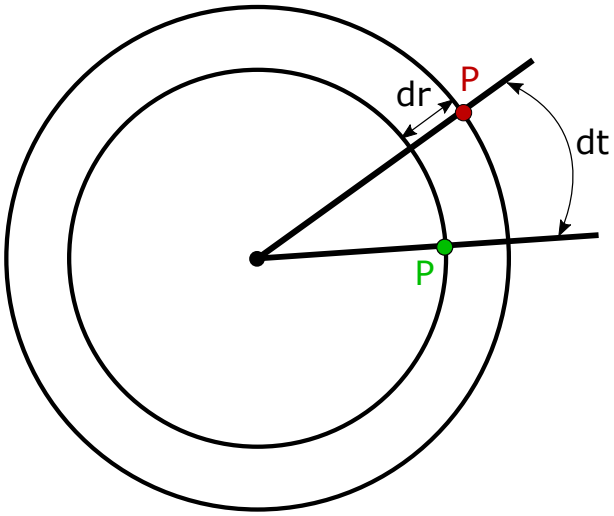


Figure 2.7: Visualisation of radial and tangential distortions

Figure 2.7 is a visualisation of how a point is distorted. Here the green P is the 3D point of interest should project to, and the red P is the point it actually projected to due to distortions. dr and dt represent radial and tangential distortions respectively. As can be seen from the figure the result is that straight lines in reality will be projected onto the image frame as curves.

2.2.4.3 Underwater challenges

There are challenges with camera calibration in water, that do not apply for calibration on land. The transition between water to glass and glass to air is one issue that is not accounted for in most camera models that will lead to refraction errors. Refraction errors happens when a light ray transitions between two mediums with different optical density. In transition the ray will change its angle of attack, and is bent according to Snell's law.

$$\frac{sin(\theta_2)}{sin(\theta_1)} = \frac{n_1}{n_2} \tag{2.7}$$

where θ_i is the angel measured from the normal of the boundary between the mediums and n_i is the refraction index of the medium.

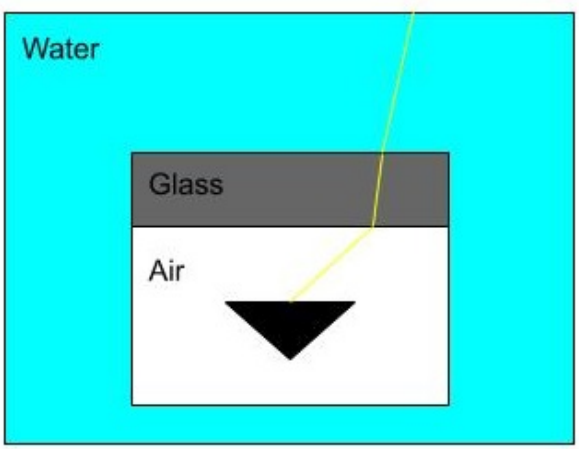


Figure 2.8: How light transitions between different medium in an underwater camera housing with a flat glass

The pin-hole model will here be fundamentally wrong, but it is demonstrated in [30] that it still works reasonably well if the objects of interest stay in a range from the camera, which it is calibrated for.

There have been proposals for solutions which takes refraction into consideration, both in terms of camera models and physical solutions. Three significant physical approaches are, 1) To minimise the distance the rays will travel through other mediums than water, by applying a very thin glass and put the camera as close to the glass as possible to reduce distance in air and glass, 2) Use a correction lens, or 3) use a dome. With the correct curvature a dome can extend the light rays with the same angle, but it requires that the lens of the camera is placed perfectly inside the domes projection centre. A dome is used in this report, however the camera is not placed in the theoretically perfect spot, and refraction errors will therefore still be present, but hopefully to a lesser extent than with a flat glass.

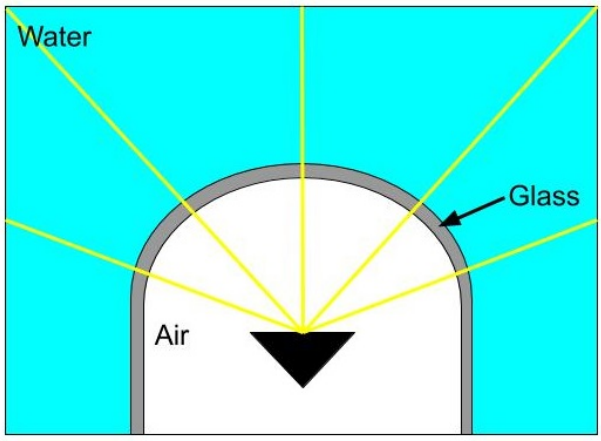


Figure 2.9: How light transitions between different medium in an underwater camera housing with a Dome

Another challenge is the light absorption and light scattering of water. Most colours are absorbed or scattered, and one is usually left with a monotone green-blue tint on the image. This might be a major concern for the indirect SLAM methods, as they might have a problem detecting features from monotone frames. In addition the absorption of visible light will reduce the distance the camera is able to view.

One more concern are the amount of free floating particles in water, especially marine snow might lead to a lot of noise in the extracted features, see figure 2.10. It might be necessary to develop a SLAM algorithm that filters out fast moving particle's in order to get a robust motion estimation with underwater visual odometry.

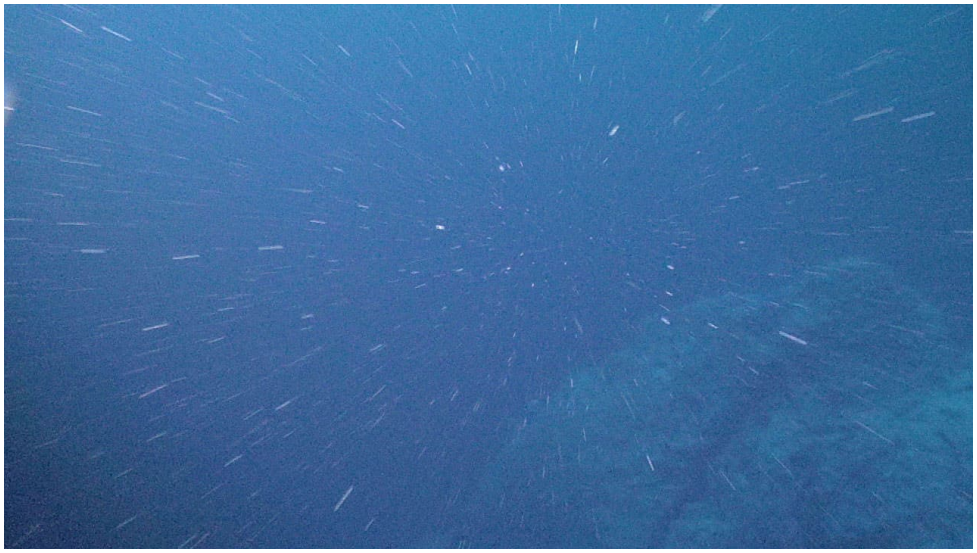


Figure 2.10: Example of marinesnow, image from wikipeida[1]

2.2.5 Pressure sensor

A pressure sensor measures the force per unit area in a gas or liquid. There are multiple approaches to measure the pressure, and different sensors measures pressure with regards to different references. Some measures against perfect vacuum, other against the atmospheric pressure, or any other arbitrary scale.

Pressure sensors in an AUV is used to measure the depth of a dive. There might be some dynamic pressures created by waves very close to the surface, but when moving down the water column the pressure is mainly hydro-static. The hydro-static pressure can be assumed to increase linear with depth. This property can be abused to get a good depth estimation at all times only utilizing a pressure sensor, the relationship between depth and pressure can be seen in equaton 2.8.

$$z = \frac{P - P_0}{g\rho} \tag{2.8}$$

where ρ is the density of water, P is the measured pressure, P_0 is the scale pressure(1bar for the atmospheric pressure), g is the gravitational constant and z is the depth.

2.3 Underwater vehicles

The main existing underwater vehicles are remotely operated vehicles (ROV) and autonomous underwater vehicles (AUV). In the following sections the two different vehicles will be explained in detail, and the basic principles of application, operation, components, buoyancy and stability will be further explained.

2.3.1 ROV

A ROV is, as the name indicates, a remotely controlled vehicle. These have been the most common unmanned vehicles, as they require less sensors and automated software than AUVs. ROVs are connected with a physical link, often referred to as a umbilical chord, to a command centre, as of today the command centres are usually located on a vessel on the surface. Through the link the ROV receives control signals and power, and transmits useful live sensor data back a pilot. From the command centre a pilot will operate the ROV by having direct control over the actuators. The pilot will usually utilize the available sensors(Depending on ROV class) like cameras and sonars to get a sense of the closest surroundings, a pressure sensor to see depth, a DVL to measure velocities, and an acoustic system to estimate the position of the ROV with regards to the command centre. ROVs can typically be divided into three different classes[17]:

- Class I - Eyeball ROV
- Class II - Observation ROV
- Class III - Work class vehicles

The Eyeball ROVs are often small in size and are used to survey areas of interest in shallow waters, they usually only carries camera. An observation with payload ROV will be larger and can carry additional payloads like sonars and CTDs for recording conductivity, temperature and pressure. They can be equipped with manipulator arms and work in the ocean. The work class ROVs can be used for inspection and exploration of subsea infrastructure. They are large and can have a mass of several tons. Unlike the observation class the work class is equipped with hydraulic manipulators to intervene with the environment performing tasks like turning valves or replacing bolts.

Work class ROV operations are not cheap, they requires a specialised crew and a surface vessel with a command centre, capability to launch a ROV into the sea and a dynamical positioning(DP) system. The time an operation will take is often dependent on the The proficiency of the pilot. The weather is also an important factor as a favourable weather window is necessary to perform operations[29]. It has therefore been a push in both research and industry to try to automate ROV-operation to reduce cost of operations.

Figure 2.11: NTNU's ROV Minerva 100K secured on the R/V Gunnerus

2.3.2 AUV

Different from ROVs, an AUV is not connected to a control centre by a physical link and therefore act independent of a pilot. It must carry its own power source, and independently calculate its own control inputs while navigating. Since the AUV have to carry its own power source the operation times are limited as they will eventually have to charge or change batteries.

AUVs must have their mission programmed ahead of deployment. A mission might be as simple strictly following waypoints to survey the seabed, but it may also have adaptive behaviour where the programmed algorithm generates new waypoints based on sensor readings. An example of more complex mission behaviour can be found in [9]. Here the authors presents a method where their AUV used a model of the ocean and sampled data during operation to alter its survey area. The advantage with this approach is that the AUV adapt to the new information it gathers during operation, which was not available to the programmer beforehand. This would be equivalent ROV pilot adjusting course based on the visual feedback from the camera stream to reach an area of interest.

For a mission to be carried out successfully the accuracy of the navigational system is critical. As GNNS is not available for a submerged AUV it must rely entirely on its sensors for navigation. Since there will always be noise present in measurements, and a bias in each sensor there will be some error in all measurements which will accumulate over time. To counter the drift caused by this accumulation the AUV is forced to surface periodically to correct its position by connecting to the GNNS. Alternatively if the AUV is operating in an area with underwater navigational equipment it might communicate with those.

As of today most AUVs are considered small, do not have hovering or manipulating abilities, and are mostly used for mapping and surveillance. They are preferred over ROVs for mapping and surveillance tasks as their lack of a physical links gives them a higher range than ROVs, which have limited range due to the forces working on their umbilical chord[17]. Even though AUVs with manipulating capabilities are still in an early phase, AUVs may still be useful for locating the area of interest to perform an inspection. There have been research done on missions combining an AUV and a ROV, the AUV to do mapping and surveillance of an area to locate the area of interest, and then deploy a ROV to inspect the area[19].

2.4 Frames

When working with navigation and localisation one of the most important questions is position in regards to what. The information that an agent are at a given x- and y-coordinate is useless unless there is a frame of reference. There are multiple ways of defining frames. In some scenarios it may be most natural for ships and AUVs operating in the ocean to use longitudinal and lateral coordinates, in another cases it might be more natural to base the positioning on the coordinates in a local map, or maybe define the initial location as the origin of the positioning frame.

In more complex systems that carries multiple sensors, like AUVs, one must also consider the frames of the different sensors. It is common to use a base frame, and then specify rotations and translations between the different sensor frames and the base frame. This is critical as the same type of sensor in a single system can yield vastly different values based on where it is located. As an example imagine the sensor readings of an IMU placed on the tip of the wing on an airbus compared to the exact same IMU placed in the cockpit during a turn. The IMU placed on the wing will experience more acceleration, and this must be taken into consideration when calculating the overall change of the vehicles states.

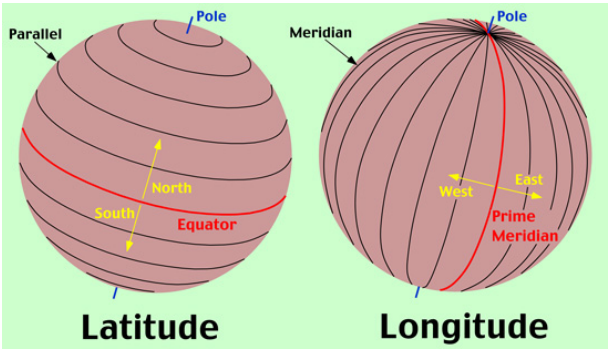


Figure 2.12: Lateral and longitudinal coordinates of the earth

2.4.1 States of a vehicle

Before diving into the rotation between frames, it is important to understand states and degrees-of-freedom(DOF) of a vehicle. The amount of DOFs of a robotics system is how many independent movements it can do without breaking any constraints. For a rigid vehicle like most AUVs it is common to operate with 6 DOFs, surge, sway, heave, roll, pitch and yaw. Where surge, sway and heave are pure translations, while roll, pitch and yaw are pure rotations. While an AUV do not have to move or rotate strictly in any of the pure translations or rotations, all translations and rotations in a 3D space can be expressed as a combination of the 6 degrees of freedom for a rigid vehicle.

There is also more complex systems, like robotics arms and snake robots which consists of a lot of modules put together by joints. Based on the amounts of modules they will have an increasing amount of DOFs in their dynamic which must be considered and handled accordingly when performing tasks. In figure 2.13 one can see an AUV with 6DOFs degrees of freedom.

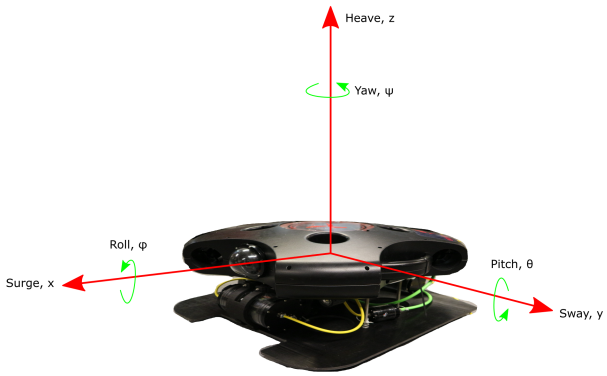


Figure 2.13: NTNU Vortex AUV Mantas 6DOFs visualised

In this report the AUV of interest is rigid, and we will be working with 6 DOFs. In different systems different states are of interest, but for a 3D positioning system the pose ρ is the most important state.

The pose is a combination between the translations and rotations of the current system, mathematically expressed as a vector of length equal to the DOFs of the system, see equation 2.9

$$\boldsymbol{\rho} = \begin{pmatrix} x \\ y \\ z \\ \phi \\ \theta \\ \psi \end{pmatrix} \tag{2.9}$$

The state of the two time derivatives $\dot{\boldsymbol{\rho}}$ and $\ddot{\boldsymbol{\rho}}$ is also of interest when predicting $\boldsymbol{\rho}$. The estimation of the states will be be looked at in Section 2.5 about state estimaton and filtering.

2.4.2 Rotation matrices

One of the most central aspects when working with different frames is to define the rotation between them. If all sensors on a system is assumed fixed in place the rotation between sensor frames can be assumed static. The rotation between different frames can be expressed mathematically as a rotation matrix.

Rotation matrices allows for easy conversion of points between frames. Given a point in frame A and a rotation matrix from frame A to frame B, $\boldsymbol{R}_{BA}$, the same point in frame B can simply be calculated as:

$$\begin{pmatrix} x_B \\ y_B \\ z_B \end{pmatrix} = \boldsymbol{R}_{AB} \cdot \begin{pmatrix} x_A \\ y_A \\ z_A \end{pmatrix} \tag{2.10}$$

given that the origins of the two frames coincide. Here x_B, y_B and z_B are the Cartesian coordinates in frame B and x_A, y_A and z_A the coordinates in frame A. Often the origin of the two frames will not coincide, if that is the case a translation term τ which describes the translation between frames must be added to the equation.

As mentioned earlier all rotations in a 3D space is a combination of the three pure rotations roll, yaw and pitch. Hence the combined 3D rotation matrix is a combination of three pure rotation matrices. The pure rotation matrices are as follows:

$$\boldsymbol{R}_{\phi} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\phi) & -sin(\phi) \\ 0 & sin(\phi) & cos(\phi) \end{pmatrix} \tag{2.11}$$

$$\boldsymbol{R}_{\theta} = \begin{pmatrix} cos(\theta) & 0 & sin(\theta) \\ 1 & 0 & 0 \\ -sin(\theta) & 0 & cos(\theta) \end{pmatrix} \tag{2.12}$$

$$\boldsymbol{R}_{\psi} = \begin{pmatrix} cos(\psi) & -sin(\psi) & 0 \\ sin(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.13}$$

Which yields a complete rotation matrix as:

$$\boldsymbol{R}_{rot} = \boldsymbol{R}_{\phi}\boldsymbol{R}_{\theta}\boldsymbol{R}_{\psi} \tag{2.14}$$

Expanded into:

$$\boldsymbol{R}_{r} = \begin{pmatrix} c(\psi)c(\theta) & -s(\psi)c(\phi) + c(\psi)s(\theta)s(\phi) & s(\psi)s(\phi) + c(\psi)c(\phi)s(\theta) \\ s(\psi)c(\theta) & c(\psi)c(\phi) + s(\phi)s(\theta)s(\psi) & -c(\psi)s(\phi) + s(\theta)s(\psi)c(\phi) \\ -s(\theta) & c(\theta)s(\phi) & c(\theta)c(\phi) \end{pmatrix} \tag{2.15}$$

Where s and c corresponds to sin and cos respectively.

2.4.3 Euler angles vs quaternion

Another important question when looking at rotations is how to represent them. A human is used to look at the world in Euler angles, one distinct angle going from 0 to 2π for each of the pure rotations. While this notation is the easiest and most intuitive there are some rotations that will lead to singularity(Division by zero) or simply an undefined rotation.

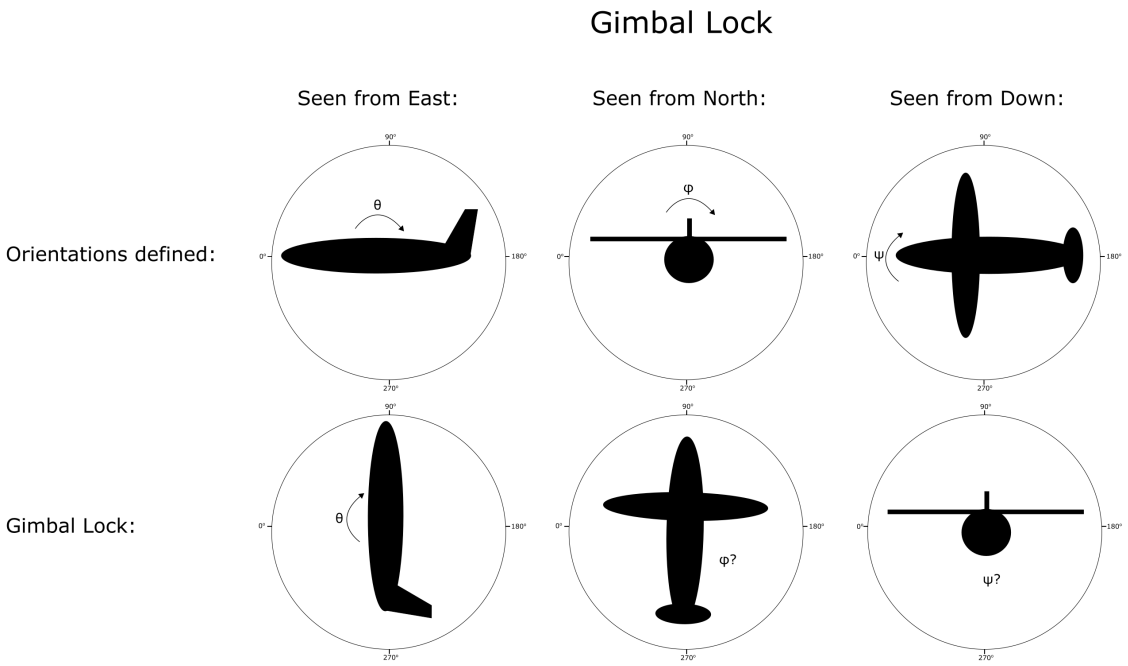


Figure 2.14: A visualisation of an aeroplane and its orientations when pitch is 0 and 90 degrees

For an aeroplane flying in the NED-frame see figure 2.14 a problem will occur once its pitch reaches 90°, this is called a Gimbal lock. At this point its yaw and roll will no longer be defined. This phenomena has created a lot of bugs in computer code.

An alternative approach have been to use quaternions, first suggested all the way back in the 1840s independently by W.R Hamilton[11] which coined the term and B.Olinde Rodrigues. While seen as necessary even back then it is first the last couple of centuries with the raise of the computer quaternions are becoming mainstream.

The connection between Euler and quaternions can be seen in equation 2.16.

$$\begin{pmatrix} q_0 \\ q_1 \\ q_2 \\ q_3 \end{pmatrix} = \begin{pmatrix} cos(\phi/2) \cdot cos(\theta/2) \cdot cos(\psi/2) + sin(\phi/2) \cdot sin(\theta/2) \cdot sin(\psi/2) \\ sin(\phi/2) \cdot cos(\theta/2) \cdot cos(\psi/2) - cos(\phi/2) \cdot sin(\theta/2) \cdot sin(\psi/2) \\ cos(\phi/2) \cdot sin(\theta/2) \cdot cos(\psi/2) + sin(\phi/2) \cdot cos(\theta/2) \cdot sin(\psi/2) \\ cos(\phi/2) \cdot cos(\theta/2) \cdot sin(\psi/2) - sin(\phi/2) \cdot sin(\theta/2) \cdot cos(\psi/2) \end{pmatrix} \tag{2.16}$$

While not intuitive for a human, quaterions allows for free rotations in 3D space without having to implement code for handling conditions where Euler angles are undefined. Therefore they are much used in fields like robotics, virtual reality and computer graphics where rotation is central. By using the conversion between Euler angles and quaterions from equation 2.16 once can derive the full 3D rotation matrix as:

$$\boldsymbol{R} = \begin{pmatrix} 1 - 2(q_2^2 + q_3^2) & 2(q_1 q_2 - q_0 q_3) & 2(q_0 q_2 + q_1 q_3) \\ 2(q_1 q_2 + q_0 q_3) & 1 - 2(q_1^2 + q_3^2) & 2(q_2 q_3 - q_0 q_1) \\ 2(q_1 q_3 - q_0 q_2) & 2(q_0 q_1 + q_2 q_3) & 1 - 2(q_1^2 + q_2^2) \end{pmatrix} \tag{2.17}$$

2.5 State estimation and filtering

2.5.1 The Bayesian filter

When dealing with a dynamic system it is often desirable to estimate its position, orientation and velocities at a given time. The estimation is done by using the measurements coming from the sensor(s) together with previous knowledge of the states. This is called state estimation, and often refereed to as filtering.

Filtering is formulated by the kinematic model $\mathbf{p}(\mathbf{x}_k|\mathbf{x}_{k-1})$ and the measurement model $\mathbf{p}(\mathbf{z}_k|\mathbf{x}_k)$. The kinematic model predicts how to the state evolve in time, and the measurement model is concerned with how the real sensor data correlates with the kinematic models prediction. For the specified models to make sense the Markov property must hold, it assumes that the next state is only dependent on the previous state as can be seen in equations 2.18 and 2.19.

$$\mathbf{p}(\mathbf{x}_k|\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{k-2}, \mathbf{x}_{k-1}, \mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_{k-2}, \mathbf{z}_{k-1}) = \mathbf{p}(\mathbf{x}_k|\mathbf{x}_{k-1}) \tag{2.18}$$

$$\mathbf{p}(\mathbf{z}_k|\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_{k-1}, \mathbf{x}_k, \mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_{k-2}, \mathbf{z}_{k-1}) = \mathbf{p}(\mathbf{z}_k|\mathbf{x}_k) \tag{2.19}$$

State estimation is done in a cyclic manner consisting of two steps, prediction and update. In the prediction step an estimate of the current state is done based on the knowledge of the previous state by the kinematic model following the total probability theorem:

$$\mathbf{p}(\mathbf{x}_k|\mathbf{z}_{1:k-1}) = \int \mathbf{p}(\mathbf{x}_k, \mathbf{x}_{k-1}|\mathbf{z}_{1:k-1})\mathbf{dx}_{k-1} = \int \mathbf{p}(\mathbf{x}_k|\mathbf{x}_{k-1})\mathbf{p}(\mathbf{x}_{k-1}|\mathbf{z}_{1:k-1})\mathbf{dx}_{k-1} \tag{2.20}$$

The update is based on the current sensor measurements and follows Bayes Rule:

$$\mathbf{p}(\mathbf{x}_k|\mathbf{z}_{1:k}) = \frac{\mathbf{p}(\mathbf{z}_k|\mathbf{x}_k)\mathbf{p}(\mathbf{x}_k|\mathbf{z}_{1:k-1})}{\mathbf{p}(\mathbf{z}_k|\mathbf{z}_{1:k-1})} \propto \mathbf{p}(\mathbf{z}_k|\mathbf{x}_k)\mathbf{p}(\mathbf{x}_k|\mathbf{z}_{1:k-1}) \tag{2.21}$$

The equations 2.20 and 2.21 is the most important in the filtering problem, but there are no guarantee that they will yield a closed form solution. Most pdfs will not lead to a closed-form expression of the prediction. Future the multiplication of pdfs done in the update step might yield an unknown pdf.

2.5.2 The Kalman Filter

By assuming both the Markov model and the measurement model to be Gaussian and linear a closed form solution is ensured. This is the basis the for The Kalman Filter(KF)[13]. In a KF the models are written on a Gaussian linear form[4]:

$$\mathbf{p}(\mathbf{x}_k|\mathbf{x}_{k-1}) = \mathcal{N}(\mathbf{x}_k; \mathbf{F}\mathbf{x}_{k-1}, \mathbf{Q}) \tag{2.22}$$

$$\mathbf{p}(\mathbf{z}_k|\mathbf{x}_k) = \mathcal{N}(\mathbf{z}_k; \mathbf{H}\mathbf{x}_k, \mathbf{R}) \tag{2.23}$$

$$\mathbf{p}(x_0) = \mathcal{N}(\mathbf{x}_0; \widehat{\mathbf{x}}_0, \mathbf{P}_0) \tag{2.24}$$

Where equation 2.24 is the initial state which also must be linear Gaussian, $\boldsymbol{F}$ is the transition matrix and $\mathbf{H}$ the measurement matrix. $\boldsymbol{Q}$ is the plant-model noise and $\boldsymbol{R}$ is the measurement noise. The models can be rewritten into a more familiar state space formulation as[4]:

$$\mathbf{x}_k = \mathbf{F}\mathbf{x}_{k-1} + \mathbf{v}_k, \ \mathbf{v}_k \sim \mathcal{N}(0, \mathbf{Q}) \tag{2.25}$$

$$\mathbf{z}_k = \mathbf{H}\mathbf{x}_k + \mathbf{w}_k, \ \mathbf{w}_k \sim \mathcal{N}(0, \mathbf{R}) \tag{2.26}$$

$$\mathbf{x}_0 \sim \mathcal{N}(\widehat{\mathbf{x}}_0, \mathbf{P}_0) \tag{2.27}$$

With these new forms for the Markov and measurement model the cyclic filtering can be formulated in the KF algorithm seen in table 2.1.

The Kalman Filter Algorithm	
$\mathbf{KF}(\widehat{\mathbf{x}}_{k-1}, \mathbf{P}_{k-1}, \mathbf{z}_k)$	Start of algorithmm
1: $\widehat{\mathbf{x}}_{k\|k-1} = \mathbf{F}\widehat{\mathbf{x}}_{k-1}$	Predicted state estimate
2: $\mathbf{P}_{k\|k-1} = \mathbf{F}\mathbf{P}_{k-1}\mathbf{F}^T + \mathbf{Q}$	Predicted covariance
3: $\widehat{\mathbf{z}}_{k\|k-1} = \mathbf{H}\widehat{\mathbf{x}}_{k\|k-1}$	Predicted measurement
4: $\boldsymbol{\nu}$=$\mathbf{z}_k - \widehat{\mathbf{z}}_{k\|k-1}$	The innovation
5: $\mathbf{S}_k = \mathbf{H}\mathbf{P}_{k\|k-1}\mathbf{H}^T + \mathbf{R}$	The innovation covariance
6: $\mathbf{W}_k = \mathbf{P}_{k\|k-1}\mathbf{H}^T\mathbf{S}_k^{-1}$	The Kalman gain
7: $\widehat{\mathbf{x}}_k = \widehat{\mathbf{x}}_{k:k-1} + \mathbf{W}_k\boldsymbol{\nu}_k$	The posterior state estimate
8: $\mathbf{P}_k = (\mathbf{I} - \mathbf{W}_k\mathbf{H})\mathbf{P}_{k\|k-1}$	The posterior covariance

Table 2.1: The Kalman Algorithm

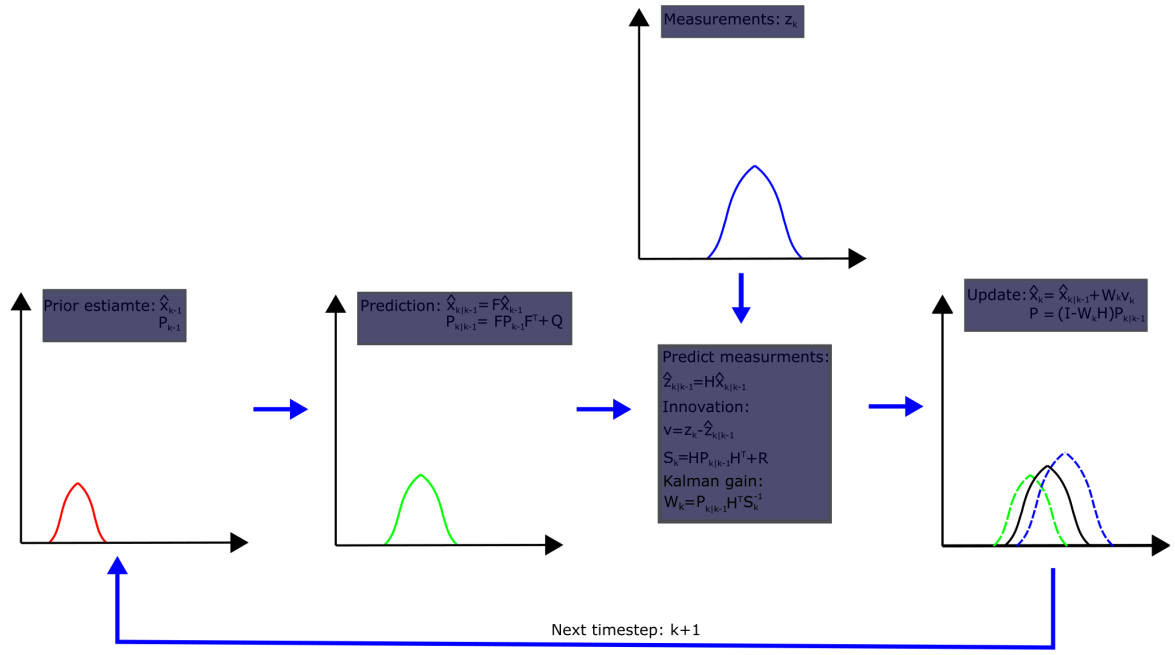


Figure 2.15: A visualisation of the KALMAN filter

A visualisation helpful to intuitively grasp the concept can be seen in figure 2.15. Left in the figure one can see the distribution of the prior estimate in red. Following the arrows the prior estimate is run through the kinematic model to predict the current estimate $\widehat{\mathbf{x}}_{k|k-1}$ marked in green. The predicted measurements are then feed through the measurement model to predict $\widehat{\mathbf{z}}_{k|k-1}$, the predicted measurements are subtracted by the actual measurements marked in blue to find the innovation $\boldsymbol{\nu}$. The covariance of the innovation $\mathbf{S}_k$ and prediction $\mathbf{P}_{k|k-1}$is then used to calculate the Kalman gain $\mathbf{W}_k$. The Kalman gain is then used to derive the most likely posterior state estimate.

Since the transition and measurement matrices usually is given, the design of a KF consists of choosing the $\mathbf{Q}$ and $\mathbf{R}$ matrices. In some sense this can be seen as weighing our trust in the model against our trust in the measurements. A high $\mathbf{R}$ and low $\mathbf{Q}$ will correspond to high uncertainty in the measurements, and more trust in the model and the other way around if the values are switched. There are many approaches to tune the matrices they will not be discussed in this report but suggested reading would be [4] pages 61 to 66.

The issue with the standard Kalman Filter is that the world does not behave in a Gaussian-linear fashion. While it for some systems might be a good enough approximation other systems deviate to far for the Gaussian-linear assumption to work. When this is the case non-linear filtering techniques must be applied. One such non-linear approach is the Extended Kalman Filter(EKF).

2.5.3 The Extended Kalman Filter

A very common approach to non-linearities is linearization around a point of interest. When applying this method linear estimation techniques can be utilized in the proximity of the chosen point. An EKF

handles non-linearities by choosing the most recent estimate as the linearization point for the prediction step, and then the most recent prediction for the update step. EKFs can be used for systems where the models can be written as[4]:

$$\mathbf{x}_k = \mathbf{f}(\mathbf{x}_{k-1}) + \mathbf{v}_k, \mathbf{v}_k \sim \mathcal{N}(0, \mathbf{Q}) \tag{2.28}$$

$$\mathbf{z}_k = \mathbf{h}(\mathbf{x}_k) + \mathbf{w}_k, \mathbf{w}_k \sim \mathcal{N}(0, \mathbf{R}) \tag{2.29}$$

Where equation 2.28 is the Markov model and equation 2.29 is the measurement model, $\mathbf{f}$ and $\mathbf{h}$ are nonlinear functions, $\mathbf{v}_k$ and $\mathbf{w}_k$ are Gaussian distributed white noise.

$\mathbf{f}$ and $\mathbf{h}$ are the nonlinear functions that the EKF linearizes. In the algorithm the linear matrices $\mathbf{F}$ and $\mathbf{H}$ are the Jacobians of $\mathbf{f}$ and $\mathbf{h}$ with respect to $\mathbf{x}_k$ and $\mathbf{x}_{k-1}$.

$$\mathbf{F}(\widehat{\mathbf{x}}_{k-1}) = \frac{\delta}{\delta \mathbf{x}_{k-1}} \mathbf{f}(\mathbf{x}_{k-1}) \tag{2.30}$$

$$\mathbf{H}(\widehat{\mathbf{x}}_{k|k-1}) = \frac{\delta}{\delta \mathbf{x}_k} \mathbf{h}(\mathbf{x}_k) \tag{2.31}$$

By inserting the linerazation step into the regular Kalmanfilter algorithm the EKF can be written as seen in table 2.2

The Extended Kalman Filter Algorithm	
$\mathbf{EKF}(\widehat{\mathbf{x}}_{k-1}, \mathbf{P}_{k-1}, \mathbf{z}_k)$	Start of algorithmm
1: $\widehat{\mathbf{x}}_{k\|k-1} = \mathbf{f}(\widehat{\mathbf{x}}_{k-1})$	Predicted state estimate
2: $\mathbf{F}=(\delta/\delta\mathbf{x}_{k-1})\mathbf{f}(\mathbf{x}_{k-1})$	Jacobian of the prediction
3: $\mathbf{P}_{k\|k-1} = \mathbf{F}\mathbf{P}_{k-1}\mathbf{F}^T + \mathbf{Q}$	Predicted covariance
4: $\widehat{\mathbf{z}}_{k\|k-1} = \mathbf{h}(\widehat{\mathbf{x}}_{k\|k-1})$	Predicted measurement
5: $\boldsymbol{\nu}=\mathbf{z}_k - \widehat{\mathbf{z}}_{k\|k-1}$	The innovation
6: $\mathbf{H}=(\delta/\delta\mathbf{x}_k)\mathbf{h}(\mathbf{x}_k)$	Jacobian of the measurement
7: $\mathbf{S}_k = \mathbf{H}\mathbf{P}_{k\|k-1}\mathbf{H}^T + \mathbf{R}$	The innovation covariance
8: $\mathbf{W}_k = \mathbf{P}_{k:k-1}\mathbf{H}^T\mathbf{S}_k^{-1}$	The Kalman gain
9: $\widehat{\mathbf{x}}_k = \widehat{\mathbf{x}}_{k:k-1} + \mathbf{W}_k\boldsymbol{\nu}_k$	The posterior state estimate
10: $\mathbf{P}_k = (\mathbf{I} - \mathbf{W}_k\mathbf{H})\mathbf{P}_{k\|k-1}$	The posterior covariance

Table 2.2: The Extended Kalman Algorithm

There are some limitations to the EKF. If the systems evolves fast compared to the sampling frequency of the sensors the error caused by the linerazation will increase. This creates problem both for the posterior estimate which will have a large error compared to the true state, but also for the posterior covariance which will not be able to capture the uncertainty. This might make the filter overconfident. It will also have problems if the models are highly non-linear which might make the linerazation even in a close proximity of a point unable to capture the behaviour of the function.

Another limitations is its dependencies on global positional updates from the GNSS or similar systems to reduce drift in its estimations that accumulates over time as discussed in Section 2.3. For a submerged AUV it is desirable to be able to operate for long periods of time without having to surface, and for this purpose an EKF is not ideal.

2.6 2D - EKF-SLAM in structured environments

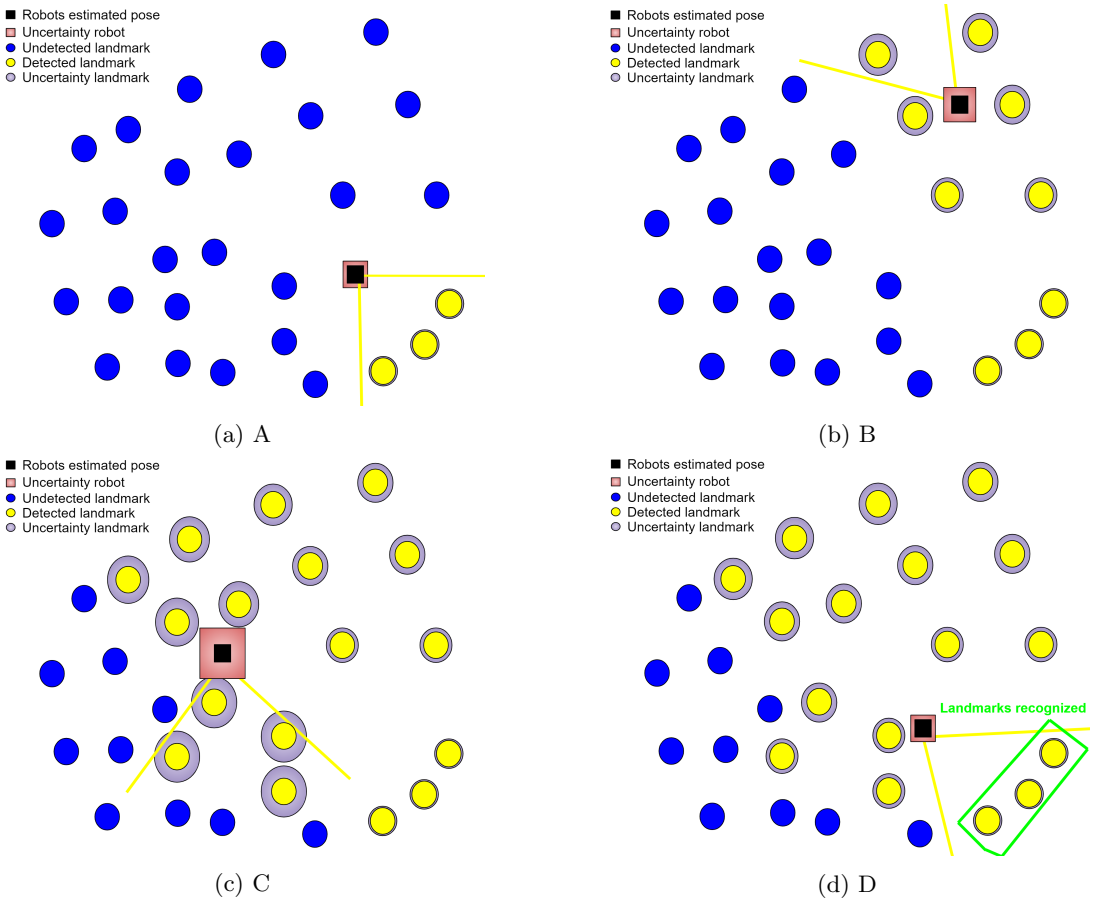
2.6.1 SLAM

As GNSS is not always available and pure inertial navigation methods will drift over time as discussed in Section 2.3, there is therefore a need for a more robust type of navigation to be utilized when GNSS is unavailable.

One popular solution to this problem has been SLAM, which tries to build a map of the surroundings and localize the robot with regards to the map. By equipping a robot with one or multiple exteroceptive sensors, a scan of the environment relative to the robot can be extracted. Assuming the surroundings are static the changes in position of landmarks between scans can be used to estimate the pose of the

robot. However only considering landmarks changes between scans will still cause drift as the external measurements are noisy.

In order to reduce drift a map is necessary that keeps track of where the landmarks are with regards to each-other in a frame of reference. By adding the scans of the environment as the robot operates a map can be built of the surroundings. The map itself will be exposed to the same error-propagation as mentioned above, however when the robot revisits an area it has been before it will be able to recognize the area. When this occurs its current uncertainty, and hence the uncertainty of its path, can be reduced to the same uncertainty as it had the last time the area was visited. This is a very powerful navigational tool.



(a) A (b) B

(c) C (d) D

Figure 2.16: Visualisation of poses and uncertainties of a robot discovering an unknown scene

From figure 2.16 one can see that in A one can see the status just after the system is deployed, here the uncertainty of the robot is the initial uncertainty of the system which is quiet small as can be seen in the size of the red square around the robot. Its exteroceptive have detected the three landmarks colored in yellow. Their uncertainty is calculated as a combination of the uncertainty of the sensor and the current belief in the robots pose. In B the robot has moved from its starting position and the error of the estimations over time have led to a bigger uncertainty in its own position than in A as is visualised by the red square. The landmarks detected will therefore have a bigger uncertainty than those seen earlier. C is a continuation of the situation in B, but the pose of the robot and position of landmarks are more uncertain than before. In D the robots sensor captures the three original landmarks which are recongnized. By looking at its own position relative to the landmarks the robot are able to locate itself with the same accuracy as the first time it was here. In addition it can backtrack and better place the landmarks previously discovered based on the information that it has completed a loop. One central part of all SLAM implementations are how to store the map. In the beginning recursive-SLAM were proposed where the map is stored in a covariance matrix. This filter approach is familar

As for sensors cameras and lidars are often used for external sensing on the surface, while sonars can be utilized in an underwater environment.

2.6.2 What is EKF-SLAM

SLAM is a fundamental problem in localisation and navigation. It is often described as a chicken and egg problem as it tries two solve two related problems at the same time, mapping the surroundings and placing itself inside the map. By adding a sensor recording the exterior, like a sonar or lidar, to a robot the measurements can be added to a EKF-core to create a SLAM system.

The hardest part of a SLAM system in to recognise features from the exterior sensor recordings and associate them to landmarks in the map. In addition in the EKF-SLAM approach the covariance matrix will grow with landmarks which makes it hard to calculate everything that is going on. In such a system the interial measurements are used to predict the state estimation, while the exterior sensor data is used in the update.

2.6.3 2D EKF-based range and bearing SLAM

This theoretical explanation of EKF-Based SLAM is based on Edmund Bakkes excellent explanation from chapter 11 in [4]. The equations and syntax in this section is based on his derivations.

One classical approach to the SLAM problem is using the familiar Markov and Measurement models found in KALMAN filters as discussed in section 2.5. In the SLAM problem the models turn out to be nonlinear, but differentiable, which makes an EKF ideal, the result is the EKF-SLAM which is explored in this chapter.

As SLAM attempts to solve both a mapping and localization problem at the same time, both the pose of the robot and the map in some frame of reference must be included in the state vector $\boldsymbol{\eta}$.

$$\boldsymbol{\eta} = \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{m} \end{bmatrix} \tag{2.32}$$

where $\boldsymbol{x}$ is the states of the vehicle, and $\boldsymbol{m}$ are the landmarks. As new landmarks are added to the map $\boldsymbol{m}$, and therefore also $\boldsymbol{\eta}$, will grow.

As discussed in Section 2.5 the prediction of the pose is given by the Markov model:

$$\boldsymbol{\eta}_k = \boldsymbol{f}_{\boldsymbol{\eta}}(\boldsymbol{x}_{k-1}, \boldsymbol{u}_k) + \boldsymbol{v}_k \tag{2.33}$$

Here $\boldsymbol{x}_{k-1}$ is the previous state and $\boldsymbol{u}_k$ is the odometry measurement between states $k-1$ and k. $\boldsymbol{m}_k$ is not estimated during prediction, as the odometry measurements carries no direct information about the landmarks.

The measurement model is given by:

$$\boldsymbol{z_k} = \boldsymbol{h}(\boldsymbol{\eta}_k) + \boldsymbol{w}_k \tag{2.34}$$

Here $\boldsymbol{z}_k$ is the true measurement, $\boldsymbol{w}_k$ is the noise of landmark measurements all assumed Gaussian. $\boldsymbol{h}_k$ is the mapping between the measurements and the reference frame of the map given by

$$\boldsymbol{h}(\boldsymbol{\rho}_k, \boldsymbol{m}^i) = \begin{bmatrix} ||\boldsymbol{m}^i - \boldsymbol{\rho}_k|| \\ \angle(-\boldsymbol{\psi})(\boldsymbol{m}^i - \boldsymbol{\rho}_k) \end{bmatrix} \tag{2.35}$$

Where $\boldsymbol{\rho}_k$ is the pose of the robot at time-step k, $\boldsymbol{m}^i$ is landmark i in the map and $\boldsymbol{\psi}$ is the yaw angle of the robot. There are two challenges with this measurement model, firstly it is nonlinear, which rules out the Kalman filter as a solution. Luckily the nonlinearities of the standard SLAM problem is differentiable, and linearisation can be utilized which leads to EKF-SLAM. In order to utilize EKF-SLAM the Jacobians of the measurement and kinematic models must be derived. For the Markov model the derivatites are straight forward from equations 2.30 and 2.31:

$$\boldsymbol{F}_{\boldsymbol{x}} = \frac{\delta \boldsymbol{f}_{\boldsymbol{x}}(\boldsymbol{x}, \boldsymbol{u})}{\delta \boldsymbol{x}} = \begin{bmatrix} 1 & 0 & -\boldsymbol{u}sin(\boldsymbol{\psi}) - \boldsymbol{v}cos(\boldsymbol{\psi}) \\ 0 & 1 & \boldsymbol{u}cos(\boldsymbol{\psi}) - \boldsymbol{v}sin(\boldsymbol{\psi}) \\ 0 & 0 & 1 \end{bmatrix} \tag{2.36}$$

$$\boldsymbol{F}_{\boldsymbol{u}} = \frac{\delta \boldsymbol{f}_{\boldsymbol{x}}(\boldsymbol{x}, \boldsymbol{u})}{\delta \boldsymbol{u}} = \begin{bmatrix} cos(\boldsymbol{\psi} & -sin(\boldsymbol{\psi}) & 0 \\ sin(\boldsymbol{\psi}) & cos(\boldsymbol{\psi}) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.37}$$

The Jacobian of the measurement models from equation 2.35 is also straight forward. Using that the derivative of the 2-norm is:

$$\frac{\delta||\boldsymbol{x}||}{\delta \boldsymbol{x}} = \frac{\boldsymbol{x}}{||\boldsymbol{x}||} \tag{2.38}$$

the Jacobians become:

$$\boldsymbol{H_x}^i = \frac{\delta \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{m}^i)}{\delta \boldsymbol{x}} = -\begin{bmatrix} \frac{1}{||\boldsymbol{m}^i - \boldsymbol{p}||}(\boldsymbol{m}^i - \boldsymbol{\rho}^T) & 0 \\ \frac{1}{||\boldsymbol{m}^i - \boldsymbol{\rho}||^2}(\boldsymbol{m}^i - \boldsymbol{\rho})^T \boldsymbol{R}(\boldsymbol{\pi/2}) & 1 \end{bmatrix} \tag{2.39}$$

$$\boldsymbol{H_m}^i = \frac{\delta \boldsymbol{h}(\boldsymbol{x}, \boldsymbol{m}^i)}{\delta \boldsymbol{m}^i} = \frac{1}{||\boldsymbol{m}^i - \boldsymbol{\rho}||^2}\begin{bmatrix} ||\boldsymbol{m}^i - \boldsymbol{\rho}||(\boldsymbol{m}^i - \boldsymbol{\rho})^T \\ (\boldsymbol{m}^i - \boldsymbol{\rho})^T \boldsymbol{R}(\pi/2) \end{bmatrix} \tag{2.40}$$

Using these notations for the Jacobian the EKF equations can be modified and entire EKF-SLAM algorithm can then be contracted to:

$$\begin{aligned}
&\hat{\boldsymbol{\eta}}_{\boldsymbol{k|k-1}} = \boldsymbol{H}\boldsymbol{P}_{k|k-1}\boldsymbol{H}^T + \boldsymbol{R} \\
&\boldsymbol{P}_{k|k-1} = \boldsymbol{F}\boldsymbol{P}_{k-1}\boldsymbol{F}^T + \boldsymbol{G}\boldsymbol{Q}\boldsymbol{G}^T \\
&\boldsymbol{S}_k = \boldsymbol{H}\boldsymbol{P}_{k|k-1}\boldsymbol{H}^T + \boldsymbol{R} \\
&\boldsymbol{W} = \boldsymbol{P}_{k|k-1}\boldsymbol{H}^T\boldsymbol{S}_k^{-1} \\
&\boldsymbol{\nu}_k = \boldsymbol{z}_k - \boldsymbol{h}(\boldsymbol{\eta}_k) \\
&\hat{\boldsymbol{\eta}}_{k|k-1} + \boldsymbol{W}\boldsymbol{\nu}_k \\
&\boldsymbol{P}_k = (\boldsymbol{I} - \boldsymbol{W}\boldsymbol{H})\boldsymbol{P}_{k|k-1}
\end{aligned}$$

As mentioned earlier there are two main challenges with EKF SLAM, one was nonlinearities, the other challenge is the growth of the covariance matrix. As landmarks are added to $\boldsymbol{m}$ the state vector $\boldsymbol{\eta}$ increase in size, while each landmark can be parameterized by a few parameters and $\boldsymbol{\eta}$ therefore remain sparse the covariance matrix $\boldsymbol{P}$ become dense.

This happens when a measurement $\boldsymbol{z}$ arrives that has captured multiple landmarks. When this happens all detected landmarks will contain some information about the pose of the robot, which again will contain some information about the other detected landmarks. While a dense large covariance matrix $\boldsymbol{P}$ is not necessarily a problem theoretically, in reality the extra computational power needed is a bottleneck for the system. In addition faulty detection of a landmark will affect multiple landmarks which can be a big issue.

2.6.3.1 Data association

Data association is an important part of SLAM that the EKF formulas do not address. In visual SLAM landmarks are tracked between frames, or rediscovered by using feature extractor to find shapes, colors or other semantic information. However landmarks detected by lidar or sonar will usually only give a range and a bearing. In order to associate detection with the map the fit between the measurement $\boldsymbol{z}_k$ and the landmark estimations $\hat{\boldsymbol{m}}^{\boldsymbol{i}}_{\boldsymbol{k|k-1}}$ is evaluated.

The simple and standard way to perform the evaluation is through voting. The two most popular voting schemes are Joint Compatibility Branch and Bound(JCBB)[22] and variations of Random Sample And Consensus(RANSAC) [8]. JCBB tries to find the pose that fits the largest number of landmarks to the measurements following three criterion. 1. A single measurement can at most be compatible with one landmark. 2. A set of n measurements is only compatible with n landmarks if the measurements fit sufficiently well with the joint distribution of those landmarks. 3. For a measurement to be compatible with a landmark it must lie within the landmarks validation gate.

A validation gate is a zone around the landmarks determined by the measurement covariance matrix $\boldsymbol{S}_k$ and a significance level α. In order to check if a fit is able to pass the gating test an gating test performed by a chi-square distribution which can be seen in equation .

$$\boldsymbol{\nu}^T(a)\boldsymbol{S}_k^{-1}(a)\boldsymbol{\nu}(a) < chi2inv(1-\alpha, d) \tag{2.41}$$

Here d is the dimension of $\boldsymbol{\nu}$ and a is the hypothesis. The hypothesis can take the form of

$$a = \begin{bmatrix} i_1 & i_2 & \dots & i_n \\ j_1 & j_2 & \dots & j_n \end{bmatrix} \tag{2.42}$$

where i_i is the landmark of the correspondence i while j_i is the measurement with the same correspondence. When a hypothesis is chosen $\boldsymbol{S}_k(a)$ contains only the relevant landmarks from the a, and the innovation $\boldsymbol{\nu}$ contains only estimates for the relevant measurements.

$$\boldsymbol{S}_k(a) = \begin{bmatrix} s_{i_1,i_1} & s_{i_1,i_2} & \dots & s_{i_1,i_n} \\ s_{i_2,i_1} & s_{i_2,i_2} & \dots & s_{2_1,i_n} \\ \vdots & \vdots & \vdots & \vdots \\ s_{i_n,i_1} & s_{i_n,i_2} & \dots & s_{i_n,i_n} \end{bmatrix} \tag{2.43}$$

$$\boldsymbol{\nu}(a) = \begin{bmatrix} \boldsymbol{z}_k^{j_1} - h(\hat{\boldsymbol{x}}_{\boldsymbol{k|k-1}}, \hat{\boldsymbol{m}}_{k|k-1}^{i_1}) \\ \boldsymbol{z}_k^{j_2} - h(\hat{\boldsymbol{x}}_{\boldsymbol{k|k-1}}, \hat{\boldsymbol{m}}_{k|k-1}^{i_2}) \\ \vdots \\ \boldsymbol{z}_k^{j_n} - h(\hat{\boldsymbol{x}}_{\boldsymbol{k|k-1}}, \hat{\boldsymbol{m}}_{k|k-1}^{i_n}) \end{bmatrix} \tag{2.44}$$

The JCBB tries to find the hypothesis a that passes the gating test in equation 2.41 with the most amounts of correlations. If there is a tie the winner is often chosen by the smallest Mahalanobis distance.

In order to find the different hypothesis a depth-first search algorithm is run on the data. It explores possible landmarks for the first measurment, and if it finds a compatible landmark it searches for landmarks for the subsequent measurement and so on. When the algorithm finds a landmark that is individually compatible with a measurement the methods test for join compatibility between its current lists of measurements and landmarks. If a joint probability fails the test in equation 2.41 the algorithm backtracks and tries another association. When it is clear that there are no longer a possibility to find another hyphothesis better than the current best hypothesis it also backtracks. This algorithm is best explained by a figure.

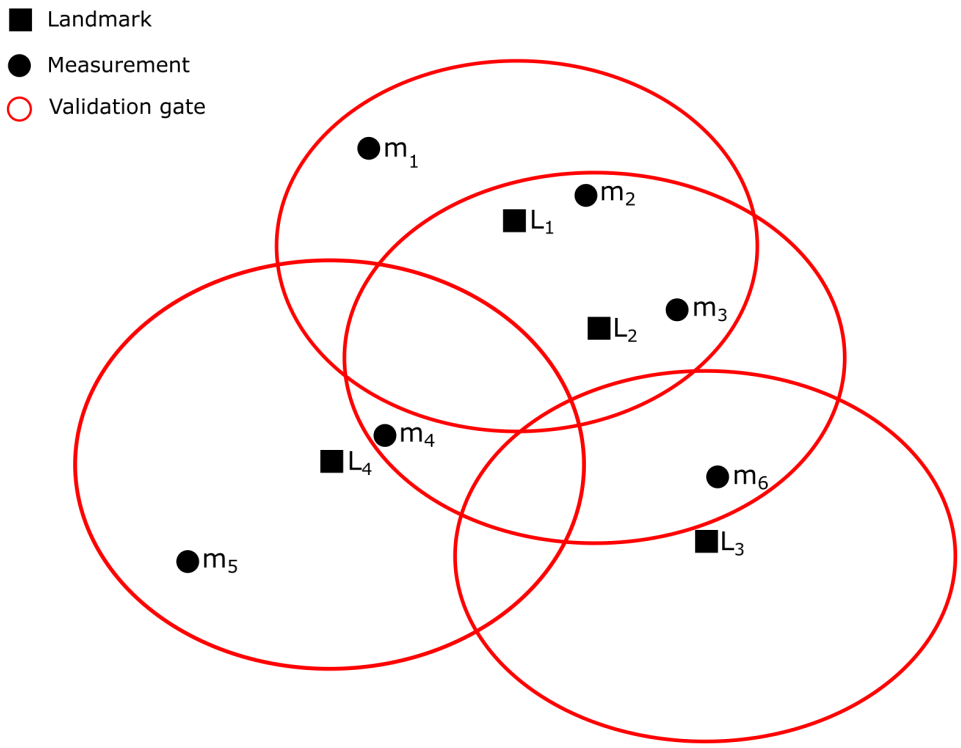


Figure 2.17: Shows landmarks and measurements to be evaluated by the depth first algorithm

In figure 2.17 four landmarks and their corresponding validation gates can be seen together with six measurements from a scan of the environment. A matrix can be created of the individual compatibility between the landmarks and measurements.

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{bmatrix}$$

Here the rows represents the measurements and the columns landmarks. The algorithm would first check the first measurement, which is only compatible with landmark 1 and run the combination through the gating test. If this combination is able to pass the gating test the algorithm will save the combination as the best hypothesis and try to add other compatible pairs and see if they pass the gating test, the hypothesis with the most amount of hypothesis will be chosen as the best. When all options are terminated the algorithm will go to measurement 2 and repeat the process until a best hypothesis for the measurements are found.

2.6.4 Sonar for SLAM

Laser scans and cameras have been the preferred sensors for SLAM as they provide data with high spatial resolution. However, in an underwater environment the range limitations of cameras and lidars makes in-practical for an underwater SLAM system. This leaves acoustic systems, and more precisely active sonars as a more viable sensor to perform underwater SLAM.

Sonar is the sensor of choice for long distance sensing in underwater environments. Yet working with acoustic landmarks is not easy as they will vary substantially based on which vantage point they are observed from. For natural scenarios were the sonar is used on the sea bottom landmarks will have to be extracted from the data and stored as points. Recognizing previously visited landmarks will therefore be a challenging problem as they will vary heavily based on your current position.

The problem substantially changes when one assumed to operate in a man-made structured environment. As shown in [link to study of acoustic response] man-made structures usually yields higher intensity responses than natural structures as they most of the time are in geometrical shapes with flat surfaces. Therefore they usually end up as lines in an acoustic scan. Keeping lines as landmarks as opposed to points is helpful as they will keep their shape when observed from different vantage points. Another change is that flat surfaces, like walls, do not let the wave pass through it, removing any shadows that are common in a scan of a natural environment. In tight confined spaces flat walls might however cause a problem as they could cause reflections by bouncing the sound waves back and forth.

There have been research done into underwater sonar-slam, in [28] a Rao-Blackwellized Particle Filter were used with forward looking sonar to create a 2D-slam approach that enabled a SLAM that could be used in environments without artificial landmarks. In [23] a combination of IMU, DVL, stereo-camera and sonar where used to improve the 3D-mapping of underwater caves and ship wrecks. The sonars range data were used to extend their pose-graph leading to more accurate scaling than just utilizing a stereo-camera. However, the method were heavily dependent on the camera, and it became clear that a minimum amount of visibility in the water was necessary for their method to perform.

In [27] an EKF-SLAM algorithm were implemented for structured environments that gave promising results in an real outdoor experiment conducted in a harbour. The authors used lines as features extracted by a modified Hough Line Transform. In the following sections the authors sonar feature extraction approach will be explored.

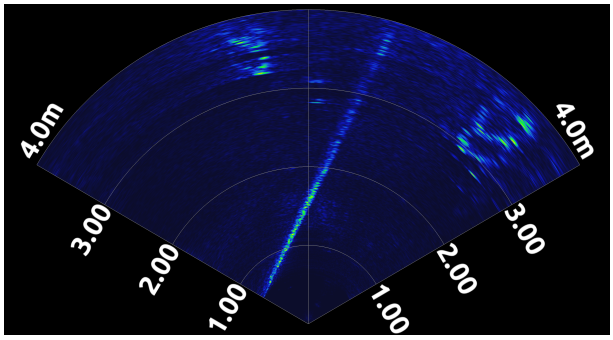
2.6.4.1 Line-based feature extraction

Line extractors from an image have for a long time been a central part of computer vision and SLAM. The Hugh Line Transform[7] is a transform used to detect straight lines. The Hugh Transform works by transforming potential lines correlating with edges into a parameter space. The parameter space will then work as a voting scheme, and the local peaks over a threshold in the parameter space will correlate to lines in the image.

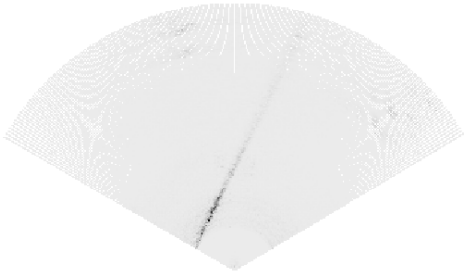
In [27] a modified version of the Hugh Transform is proposed, using a small mechanically scanned imaging sonar(MSIS) in a SLAM system working in a structured environment. He divides the extracting process into four parts, beam segmentation, data buffer, defining the Voting Space and the Voting the theory of his implementation will be taken a closer look at in the following sections.

2.6.4.1.1 Beam segmentation: When looking at a sonar scan, objects are present in neighbourhoods of high intensity bins, indicated by the red and yellow areas in figur 2.18a. When processing the data it is cheaper to only work in one channel, in other words grayscale. The same data processed in grayscale can be seen in figure 2.18b. The high intensity bins is thus the only part of the scan that is of interest when extracting line features. By using a threshold only the bins with intensities above the threshold will be stored. An image of the scan after thresholding can be seen in figure **??**, this imprint is not used directly to find line features, but are used later in order to calculate the uncertainty of a lines location.

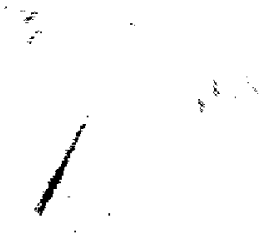
The bins used for extracting the line are the local maximums of the stored bins, as they are the ones that most likely correlates with the correct object position. Figure **??** shows the local maximas, notice that they are within the imprint of the bins after using a threhold as seen in figure **??**. More than one bin can be extracted for each beam, as more than one object might intersect a single beam. Lastly if two local maximums are too close to each other the maxima with the lowest intensity are removed.



(a) Colored raw data



(b) Raw data



(c) Threshold data

(d) Local maxima

Figure 2.18: The line segmentation process

2.6.4.1.2 Data buffer: As a MSIS is used a continuous stream of acoustics beams will arrive at the sonar head, a data buffer is necessary to store the information of the most recent beams. As no line can be present in more than 180° scan, the data buffer stores beams present in the most recent 180° and discards older beams.

The following information is stored in the buffer:

- The range and bearing of the selected bins (polar coordinates in the sensor reference frame). This are the bins segmented from the beams as discussed in the previous section. They are used in the voting process for finding the best line candidates.
- The segmented beam with all bins with an intensity above a threshold. This beam is used to measure uncertainty.
- The position of the vehicle when the beam was acquired.

2.6.4.1.3 Voting space: A modification of the classic Hugh Transform are used for finding the best line features from the acoustic image. The lines are parameterized by the in polar coordinates with $\rho^{\boldsymbol{B}}$ being the shortest distance from the base-frame, $\boldsymbol{B}$, to the line and $\theta^{\boldsymbol{B}}$ being the difference in orientation between the line and the base-frame. $\boldsymbol{B}$ can be set arbitrarily, but is here chosen as the current position of the sonar frame.

Having the parameters still leaves the problem of quantization of the voting space. With an ideal sonar and infinite of computation power one would choose the highest resolution possible However, due to the

limited resolution on the sensor itself, the quantization should be set no lower than the linear and angular resolution of the sonar. When a quantization has been set the voting space will be a grid with the set resolution as can be seen in figure 2.21.

2.6.4.1.4 Sonar model: Since each beam sent by the sonar has a beam width, each bin stored in the sonar can not simply be considered as a point, but should be considered as an arc spanning the beam width, α. By using this approach a simple model would be to restrict lines correlating to a given bin as tangents on this arc, see figure 2.19.

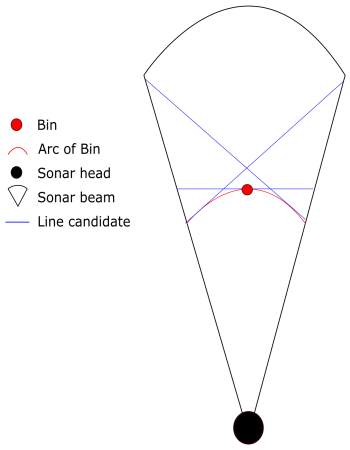


Figure 2.19: How a bin should be interpreted when processed and three line candidates

Mathematically the bearing of all those lines would simply be contained within

$$\theta_{\boldsymbol{B}} - \frac{\alpha}{2} \leq \theta_{Li} \leq \theta_{\boldsymbol{B}} + \frac{\alpha}{2} \tag{2.45}$$

Where $\theta_{\boldsymbol{B}}$ is the bearing from the sonar head to the bin, α is the beam width and $\theta_{L_{i,k}}$ is the bearing from the sonar head to candidate line. The amount of bearings, i, extracted will vary depending on the chosen resolution.

In [27] the author argues that even though this simple model would work well in air it does not explain the acoustic image from a MSIS well enough. By analysing the images he found that the object detection capability is not limited to the arc-tangent surfaces, but that those beams intersecting the surface within the limits defined by a certain maximum incidence angle, β, also produce a discernible return. Therefore an extended model is proposed where not only lines that are tangent of the arc is considered, but also beams that intersect the arc with an incidence angle smaller than $\pm\beta$.

Mathematically this can be represented using the i bearings found from equation 2.45. For every one of those bearings k values are taken in a set resolution within $\pm\beta$. Use figure 2.20 for reference.

$$\theta_{L_i} - \beta \leq \theta_{L_{i,k}} \leq \theta_{L_i} + \beta \tag{2.46}$$

$\theta_{L_{i,k}}$ will be all the bearings of lines compatible with the bin and will have the size $i \times k$. In addition to the bearings the range parameter $\rho_{L_{i,k}}$ corresponding to each of the bearings must be obtained. This is a simple trigonometric problem

$$\rho_{L_{i,k}} = \rho_{\boldsymbol{B}} cos(\theta_{\boldsymbol{B}} - \theta_{i,k}) \tag{2.47}$$

This process is done for every bin in the data buffer.

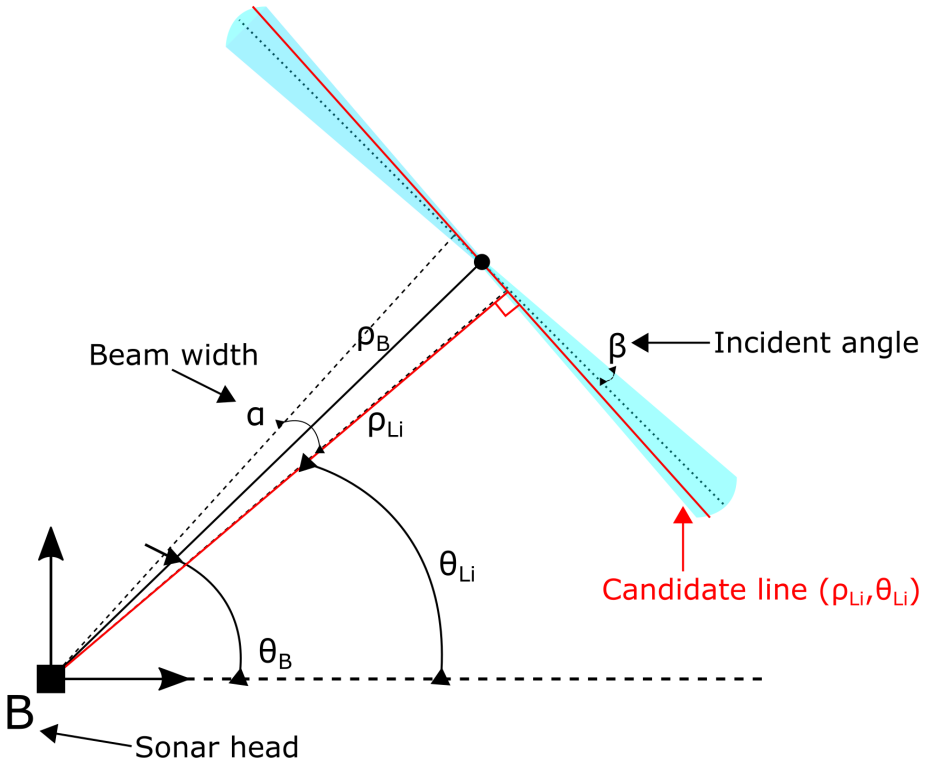


Figure 2.20: Shows a candidate line from the sonar model

2.6.4.1.5 Voting: Each bin in the data buffer will give one vote to each line it correlates with from the sonar model explained in the previous section. The votes are counted in the voting space grid discussed above. Each bin can only give one vote to a certain parameterization. This ensures that cells with multiple votes are a result of lines compatible with multiple bins. In figure 2.21 one can see an visualisation of the voting space where the x-axis is different ρ values and the y-axis different θ values. The whitest areas have received the most amount of votes, and correaltes to the best lines in the image.

Lines with votes over a certain threshold are considered winning lines and will be imported as features in the SLAM algorithm. Even though this voting scheme might seem computationally expensive, the fact that only the local maximum bins are considered reduces the work load by a lot.

Figure 2.21: The result of voting scheme of the scan shown in figure 2.18

2.6.4.2 Uncertainty Model

The candidates proposed in the line extraction are deterministic which does not integrate into the stochastic world of the EKF. Both an estimation for the location of the line and an uncertainty model of the line must be implemented.

There is a high probability there will be objects in areas where there are multiple high intensity bins, and a low probability that there will be any in the areas with low intensity bins. Therefore, the process off applying a threshold as described in the beam segmentation section can be seen as applying a confidence interval to the acoustic image. In other words a line feature will fall within the segmented image with a given confidence. The problem can be approximated to a bivariate Gaussian distribution on its ρ and θ parameters. Therefore, the problem consist of finding the bivariate Gaussian that best fits the segmented data.

Since the chosen candidate was the candidate with the most amount of votes it must lie within the segmented area. By taking a confidence interval of the segmented data we get a zone which the line will be within with a percent certainty. Then this section is searched for lines to fill it, which is stored as points in polar coordinates. The mean and covariance of a beam will be the mean and variance of all beams found in the segmentation the winning line lies within. Figure 2.22 shows all the lines found as possibilities for the most voted lines from the data shown in figure 2.18.

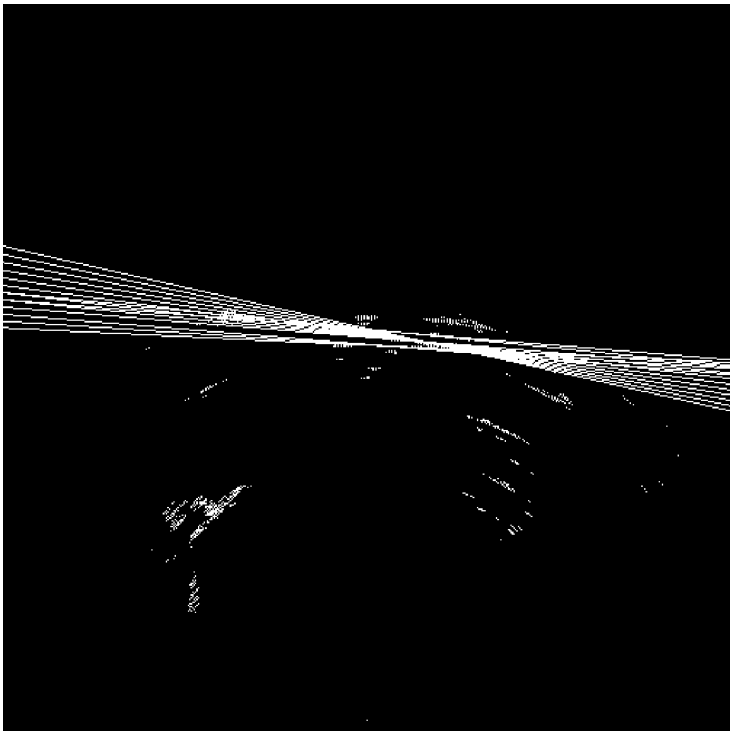


Figure 2.22: Resulting lines found for a potential line segment in the scan shown in figure 2.18

2.6.4.3 Obtaining Segments

In a map it is desirable to know the size of the features, a line parameterized in polar coordinates is unbound. However, given the acoustic threshold imprint it is fairly straight forward to find the most likely acoustic imprint by finding the overlap between the mean parameters of the bivariate Gaussian and the segmented image.

The endpoints of the lines are stored as polar coordinates pointing two the ends of the line segment. Even though they are used to visualise a map they are not actually integrated into the state vector. However for path-planning an accurate map with feature sizes is necessary, in this report the segments will be stores in an occupancy grid for later use. The lines inserted in

2.7 Combining sonar and camera

In this report the goal is not only to create a map of the environment and use the map for localisation, but also to put objects of interest into the map. As it is hard to use the sonar data by itself to recognise objects of interests the goal is to use a CNN to detect objects in the image frame, and then project the pixel values onto the sonar scan to calculate the objects position with regards to the sonar frame.

Theoretically it would be possible to do this using the sensor models of the camera and sonar, given the rotation and translation between the two frames. However due too the imperfections often experienced with camera models in an underwater environment, and the complexity of this approach an experimental approach is used in this report. However the theoretical approach is explored here.

2.7.1 Theoretical approach

For this approach it is assumed that the sonar and camera are placed close enough to each other that the scene seen by the camera is also recorded by the sonar. The orientation of the camera and sonar is the same, and the sensors shares the same centre line, in other words the central beam of the sonar corresponds to the central beam of the sonar.

As discussed in the Section 2.2.4 a camera model explains the projection between points in the scene and the camera sensor see equation 2.3. By detecting objects at certain pixels a camera model can be used to project lines back toward the point in 3D. A monocular camera will not be able to provide depth, but it can indicate at which angle from the camera sensor an object is detected. By using the information given by the camera with the depth data collected by a sonar, a fairly accurate estimate of an objects location in a 2D-plane should be possible to extract.

Even though a monocular camera can not give the depth directly, it can yield the angle from centre of the line, see figure 2.23. Here the angle $\boldsymbol{\theta}$ is the angle the projected trajectory intersects the xz-plane. This is the angle of interest as it is the same plane that a scan from an imaging sonar with the same orientation will cover.

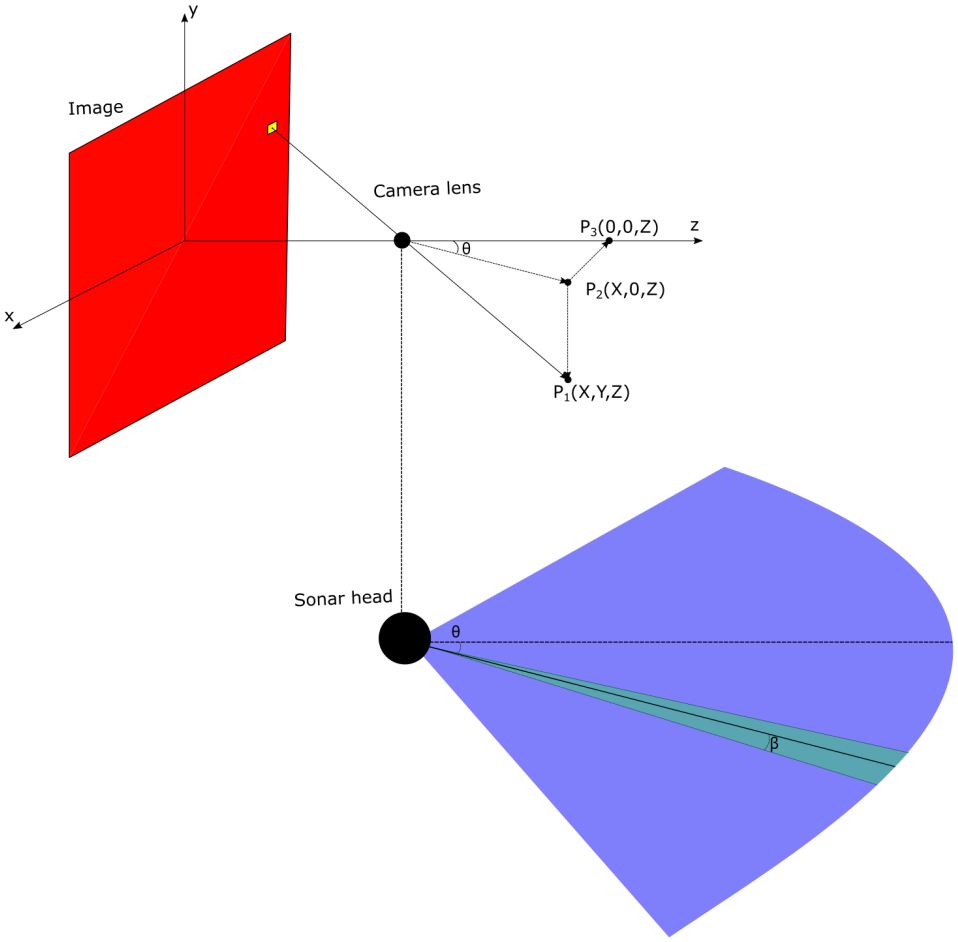


Figure 2.23: Projection of a camera pixel marked in yellow through a pinhole, $\boldsymbol{\theta}$ is the angle between the trajectory of projection and the xz-plane

Equation 2.3 will have to be rearranged in order to find θ:

$$\mathbf{X} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \boldsymbol{\Omega}^{-1}\boldsymbol{T}^{-1}s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \tag{2.48}$$

here $\mathbf{X}$ is the 3D position of the object, $\boldsymbol{\Omega}$ is the intrinsic matrix explained in equation 2.3, $\mathbf{T}$ is the eccentric matrix, u is the x-pixel coordinate, v is the y-pixel coordinate and s is the scale.

After having found $\mathbf{X}$ figure 2.23 makes it clear that $\boldsymbol{\theta}$ can be found simply by an inverse cosine:

$$\boldsymbol{\theta} = cos^{-1}(\frac{Z}{\sqrt{X^2 + Z^2}}) \tag{2.49}$$

Since the camera and sonar is assumed to have the same centre line the object should be located within a band β around the angle $\boldsymbol{\theta}$, see figure 2.23. The depth can then be estimated by looking at the high

intensity bins of the beams within this band around $\boldsymbol{\theta}$. In the case of multiple clusters of high intensity bins within the band logic must be written in order to extract the cluster that most likely correlates with the object.

In this report the objects will be large images, and it is safe to assume that in order for the computer vision algorithms to be able to classify the objects correctly the camera must have a clear line of sight. Therefore the logic could be to pick the first local maxima over a set threshold as the range to the object. Other cases may require different analysis of the sonar data.

2.7.2 Limitations of the theoretical approach

The challenges of optics in an underwater scenario discussed in Section 2.2.4.3 limits the usefulness of a theoretical approach. An accurate camera model $\boldsymbol{\Omega}$ must be found for the working conditions, meaning an camera calibration will have to be performed for every new condition such a system is set to work under. For operations in large areas the different water conditions might reduce the accuracy even if a good model is found for the initial conditions. Distortions between the mediums and the glass must also be taken into consideration and included in 2.48.

The assumption of having the exact same orientation between the camera and sonar, and only having translation about the y-axis between the sensors might also limit its use as it would not be feasible for many application.

2.8 YOLO - You only look once

2.8.1 A short history of Convolutional neural networks and YOLO

In recent time convolutional neural networks(CNNs) have become popular in the field of computer vision, and are used for everyday tasks such as opening computers via face recognition, to more critical tasks like extracting signs, other cars, light-signals and pedestrians for the decision algorithms found in self-driving cars. Tesla which is one of the leading self-driving car developers are ONLY relying on cameras for extracting information about the surroundings of their cars, not relying on lidars as most other manufacturers.

The recent popularity are mainly due to great improvements in computers, as the idea of CNNs are based on principles discussed in theory as early as the 1980s. The first breakthrough was in 2012 when AlexNet[14] achieved a top-5 error of just 15.3% during the ILSVRC-2012 considerably better than the shallow(In terms of depth) traditional state-of-the-art computer vision algorithms of the time. It consisted of eight layers in total, where five of them were convolutional, and used dropout for regularisation. This trend caught on and in the preceding years even deeper networks won. Googlenet[32] the 2014 winner consisted of 22 layers in total, while ResNet[12] the 2015 winner consisted of 152 layers achieving a top-1 error of only 3.57% in the ILSVRC-2015.

However, even though the CNNs achieves great classification score, they were not created specifically to run on real-time applications. For the competitions and non-time critical tasks that is fine, but for application where processing of an incoming stream of images is important, like in self-driving car, these networks are to slow. A lot of research went into accelerating the prediction time of the networks, and the three popular networks created as a result of the research was SDD[16], Faster R-CNN[10] and YOLO[24]. All these networks sacrifice some accuracy from the state-of-the-art networks, but greatly reduces the processing time necessary per image.

YOLO was first proposed in 2015, when it achieved an fps of 45 with a Titan X GPU while maintaining almost twice the accuracy of other real-time networks of the time. In 2017 an improvement to named YOLOv2[25] were proposed where some of the shortcomings of YOLO was addressed. The improved version were mainly focused on recall and localisation errors as those were the shortcomings in comparison with other fast state-of-the art algorithms of the time. Batch-normalisation were added after each convolutional layer, the classifier worked with higher resolution images, 448x448 instead of 224x224, switched the last dense layers to convolutional layers, combined the map of YOLOv2 were increased by 6%. In 2018 an incremental update to YOLOv2 were introduced called YOLOv3[26]. YOLOv3 turns the image into a grid, and preforms predictions on three scales in the centred in each grid.

2.8.2 YOLOv3 - How it works

2.8.2.1 Convolutional layers

YOLOv3 uses darknet-53 as its backbone. Darknet-53 is a network consisting of 53 layers trained on Imagenet to extract feature maps from an image. Features maps are the result of filters convolutioning over an image, and are refereed to as feature maps because the different filters "extracts" features from the input. The first convolutional layer of a CNN takes the original image as input, but preceding layers uses the feature map from the previous layer. For the human eye feature maps tends to make little sense after having gone through a couple of convolutions as can be seen in figure 2.24, but the computer are able to use thees maps to find similarities in classes.

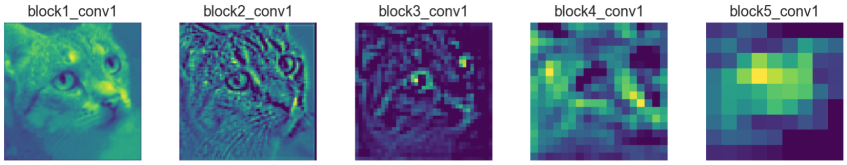


Figure 2.24: An image showing the feature map evolution going through multiple layers in a network

The convolutions are done by sliding a square matrix over the input, the square matrix is often refereed to as a filter or kernel, from here on out they will be refereed to as filters in this document. Different filters extract different features, and multiple filters are used in each convolutional layer. Filters can differ in size. The operation itself is fairly straightforward, one aligns the filter with the desired cell in the input and takes the sum of the products of the overlapping numbers. See figure 2.25. The extraction from the input must have the same size as the filter size, in the example case $3x3$ with the pixel of interest in the middle.

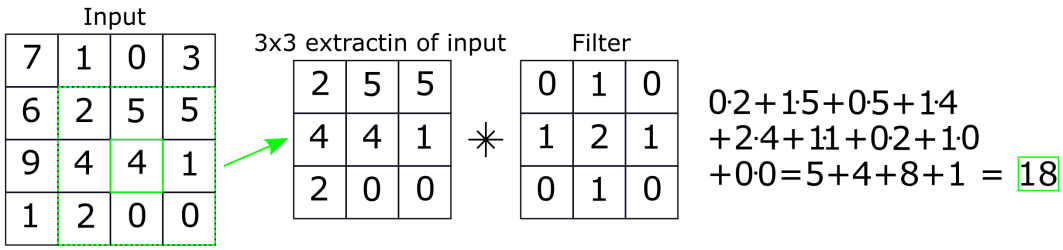


Figure 2.25: Showing the convolution operation

Stride and padding are two other parameters that also are important for the convolutional layers as they together with the filter size decides the dimensions of the output of a layer. Stride tells the filter how many cells its should shift between each convolution. If $stride = 1$ the filter will move just one cell after a convolution, if $stride = 2$ it will skip one cell between each convolution. Padding is to add a border around the input, in zero-padding for example a border of zeros with a given width is added to the input. Padding is important because it allows a layer to create an output with the same dimensions as the input which is often desirable as one wants to run a resolution through multiple feature extraction steps before down-sampling future. The formula for calculating the output size of a convolutional neural network layer is given by equation 2.50.

$$Output = \frac{W - K + 2P}{S} + 1 \tag{2.50}$$

Here W is width of input, K is the filter size, S is the stride and P is padding. If $S = 1$ the amount of padding that is needed to create an output with the same dimensions as the input can be calculated with equation 2.51.

$$Padding = \frac{K - 1}{2} \tag{2.51}$$

An example of a convolution of a $4x4$ input with a $3x3$ filter with and without padding can be seen in figure 2.26.

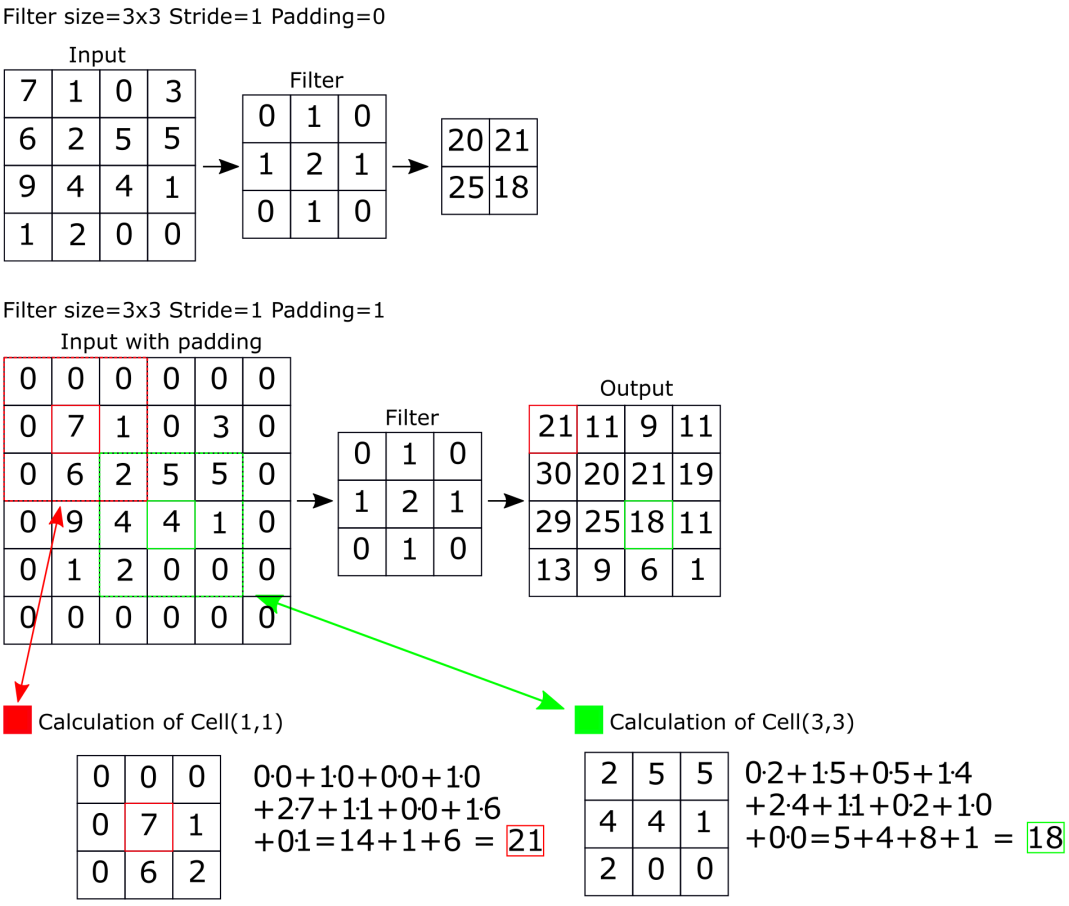


Figure 2.26: Convolution with and without padding

The end result of the convolutional layers are feature maps that the network can use to check which of the classes it is trained on(if any at all) corresponds the most with the extracted features.

2.8.2.2 Down- and Upsampling

In neural networks the majority of feature maps are usually much smaller than the original input. The main reason for this is that when looking for a feature it will not be present in all of the input. Therefore downsampling can remove redundant parts of the feature map which will lead to faster computional time.

Downsampling is usually done in one of two ways, either by using a pooling layer or setting a higher stride as explained in the previous section. The most common pooling technique is max pooling, where one simply looks at a square matrix of size NxN and only keeps the largest number. Another approach is mean pooling where one calculates the mean of all the numbers in the NxN matrix.

However reducing the feature maps down to a fraction of the original input makes it hard to detect small features. Therefore there have been a lot of suggestions upsampling the smaller feature maps and combining the output of the upsampled maps together to detect features at different scales. This is often referred to as a feature pyramid[15]. YOLOv3 takes advantage of this technique predicting objects at three different scales.

2.8.2.3 Detection layers

In the detection phase 53-new layers are added onto the existing 53-layers in darknet-53, adding up to the total of 106 layers. In the detection phase of YOLOv3 it splits the input image, at different resolutions, into a cell grid. For each cell three bounding boxes is chosen, and the objectiveness and class score for each bounding box is calculated. Figure 2.27 shows a cell marked in red which predict three bounding boxes in yellow.

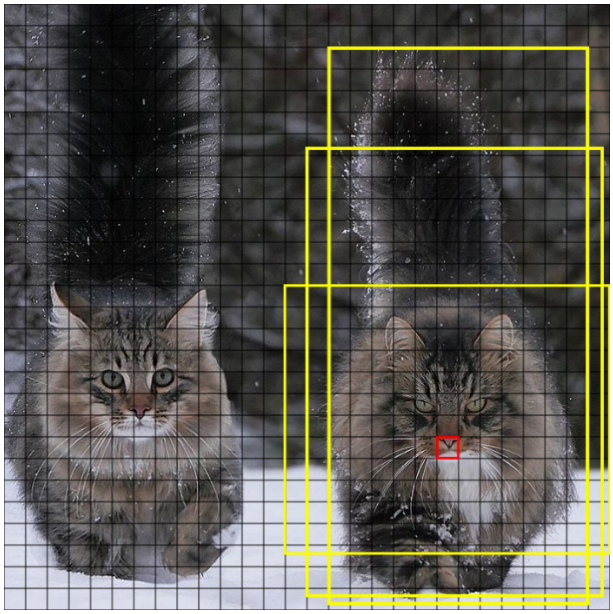


Figure 2.27: An image showing bounding boxes suggested for a cell

Unlike earlier versions of YOLO, YOLOv3 runs detection at three different scales of the image. It is done by applying a 1x1 kernel on features maps at thee of three different scales. The output of the detection is a vector of size $1X1x(3x(5+C))$. Here C is the amount of classes the network is trained on, $(5+C)$ is necessary because the output vector uses the first five terms for bounding box information. In addition it is multiplied by three because it predicts three bounding boxes foe each scale. The structure of the vector can be seen in figure 2.28.

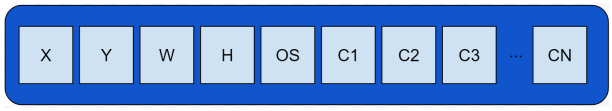


Figure 2.28: A representation of an output vector in a cell

The first two terms of the vector, X and Y, is the centre of the bounding box in pixels, the next two, W and H is the height and width of the box, and lastly OS is the objectness score of the box. The last elements Ci corresponds to the class predictions. By predicting at different scales YOLOv3 is able to detects objects of different size better than its predecessors.

2.8.2.4 The complete architecture

The complete YOLOv3 architecture can be seen in figure 2.29. It consists of 106 layers in total, with prediction done at three different scales in order to find objects at different scales. Between each layer a batch normalisation is performed, and the downsampling is done by increasing the stride of the network. Upsampled feature maps are concatenated together with the feature maps with the same resolution before downsampling in the earlier stage of the network.

All this together creates a network which even though it is not quiet as fast as YOLOv2, it provides a slightly higher map, and is much better at detecting objects at different scales.

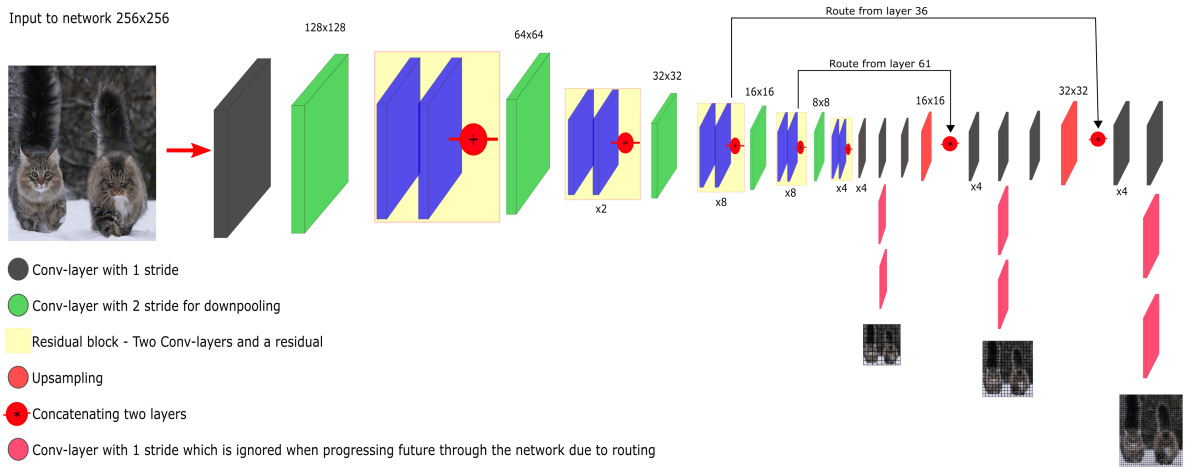


Figure 2.29: A figure of the complete YOLOv3 Architecture

Chapter 3

Method

3.1 Robosub

Robosub[35] is a yearly AUV competition held by robo-nation for students. Every year teams from all over the world builds AUVs to compete against each other. The goal of the competition is for the different teams to demonstrate the autonomy of their AUV by completing underwater tasks, with a new theme each year. The tasks varies from simple observation and recognition tasks, to interactions with the environment by pulling levers and picking up objects. The solutions in this report were implemented on Vortex - NTNUs AUV which were going to compete at Robosub2020.

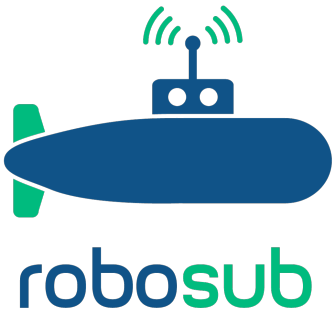


Figure 3.1: The Robosub logo

3.2 An introduction to Manta

Manta is an inspection inspection-class hovering AUV designed and developed by students of the student-organization Vortex - NTNU. It serves both as an experimental educational platform for the students of the organisation, and as a competition vehicle. It was first developed as an ROV in 2017-2018 where it participated in MATE ROV[34], but was in 2019 converted to an AUV. In the summer of 2019 Manta competed in its first Robosub held in San-Diego, USA.

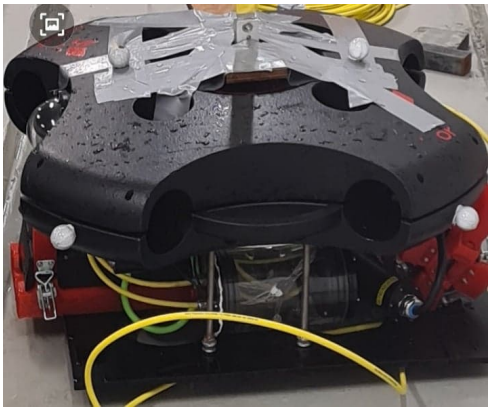


Figure 3.2: Manta

During Robosub2019 Mantas autonomous capabilities were limited to an EKF fusing IMU and DVL data, four uncoupled PD controllers for surge, yaw, sway and heave and two cameras running computer vision algorithms. The competition was therefore used more as an opportunity for gaining experience and learn the shortcoming of the implemented solutions. With the experience gathered a new software structure where proposed by Kristoffer Solberg in his Master Thesis[31]. The new structure is a more complete AUV software that would allow Manta to perform more complex mission. A figure of the proposed software architecture can be seen in figure 3.3. The implementation presented in this report presents some of the solutions implemented for the navigation and spatial-memory blocks in the figure.

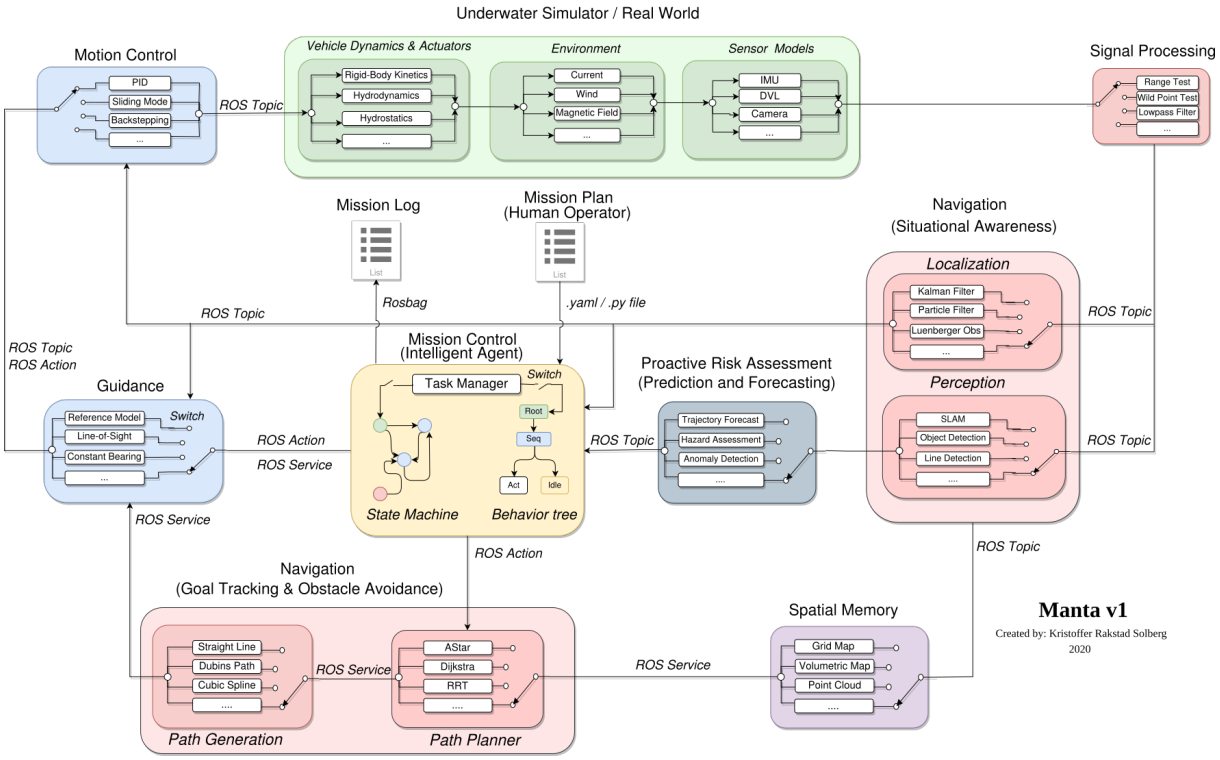


Figure 3.3: An overview of Mantas software architecture v1. Courtesy of Kristoffer Solberg[31]

3.2.1 Sensors and actuators

The two most critical components of an autonomous system are the actuators and sensors. The sensors and actuators carried by Manta can be seen in figure 3.5. Actuators are important for movement and interaction with the surroundings. Mantas only actuators are eight Blue Robotics T200 Thrusters. The amount and placement of thrusters determines the controllability of the system. The placement can be seen in figure 3.4 and allows Manta to be controlled in all six-degrees of freedom. However, the capability of the actuators to perform a manoeuvre is not enough to ensure that it is performed accurately. In order for actuators to perform accurate manoeuvres precise estimation is critical.

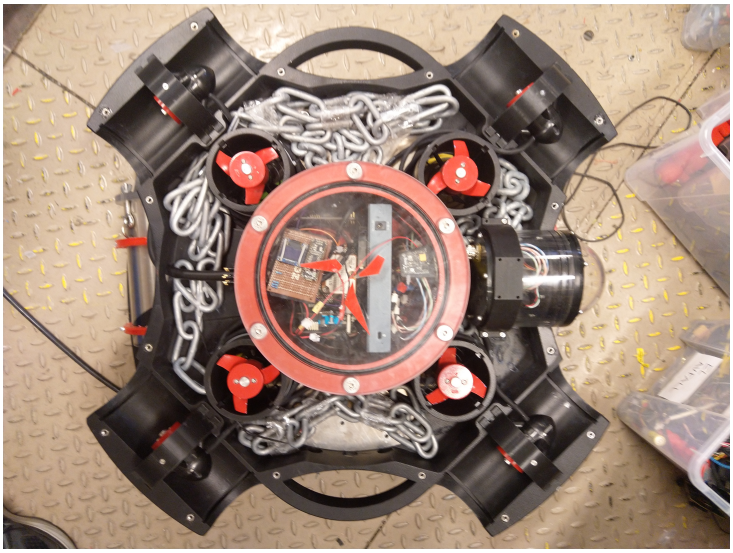


Figure 3.4: Figure showing the thruster configuration

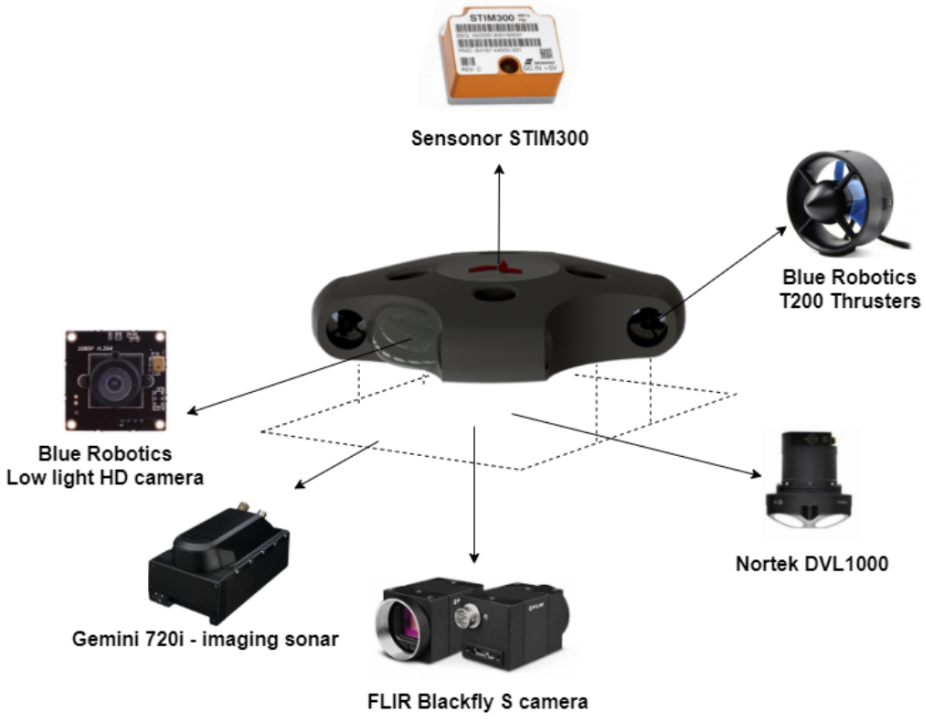


Figure 3.5: An overview of Mantas sensors and actuators. Courtesy of Øyvind Denvik[38]

3.2.1.1 Introspective sensors

Manta carries two sensors used for introspective estimation, Sensornors STIM300[20] and a Norteks DVL 1000[2].

The STIM300 contains an 3-axis accelerometer, 3-axis angular rate and 3-axis inclinometers. It provides Mantas accelerations and angular velocities. The sensor is fixed in the estimated centre of gravity of Manta, and is used as the origo of the body-frame of reference in the navigation system.

In addition to being a DBL, Norteks DVL 1000 also contains a pressure sensor. The DVL is used to measure the linear velocities of Manta relative to the bottom or water. All experiments in this report is conducted in a shallow pool and the linear velocities is therefore assumed to be relative to the bottom which is static. The pressure sensor is used to estimate the depth of Manta by taking advantage of the linear hydro-static pressure. The DVL is placed bellow the IMU, therefore a transformation from the DVL frame to the body frame is necessary when processing DVL data. All sensors are assumed to be

fixed on Manta which is assumed to be a completely rigid body. All transformations between sensors is therefore assumed static. The transformations between the sensor-frames and body-frame can be seen in table 3.1.

3.2.1.2 Exteroceptive sensors

Manta is also equipped with two exteroceptive sensors, a HD-camera[5] by blue robotics and a 720i-imaging sonar by Tritech.

The Blue Robotics Low Light HD-camera mounted in a dome pointing straight ahead. It is placed in front of the body-frame. The camera is ideal for underwater operations as it is made especially for low light situations and have on-board color-handling. The image stream is sent through ethernet to a NVIDIA-Jetson TX2 which uses them as input for computer vision algorithms.

The 720i-imaging sonar by Tritech is mounted below the HD-camera but with the same orientation. It allows for high quality scans ranging from 1 to 125 meters in front of the sonar head and a field of view of 100°. The scans are used to map the environment and state estimation.

Static Transforms						
Sensors	**X**	**Y**	**Z**	**ROLL**	**PITCH**	**YAW**
Sensornor STIM300	0	0	0	3.1415	0	0
Nortek DVL 1000	-0.035	-0.017	-0.211	3.1415	3.1415	0
Nortek Pressure Sensor	0	0	0	0	0	0
720i-Tritech Imaging Sonar	0.32	0	-0.14	0	0	0
Blue Robotics Low Light HD-Camera	0.26	0	0.05	0	0	0

Table 3.1: Table showing static transforms of sensors

3.2.2 On board computers

An Odroid XU4 controller is used as Mantas main computer and is responsible for running the GNC-system. It is flashed with minimal Ubuntu 16.04 in order to limit unnecessary background processes and maximise computational power. Odroid was the stand alone computer when Manta was used as a ROV. However, when transitioning from a ROV to an AUV additional processing power was required, especially for image processing.

NVIDIA Jetson TX2 is a small powerful computer packed with a 256-core NVIDIA pascal GPU. It was acquired in order to take care of the necessary image processing during the trainstion from ROV to AUV. It will also be the unit responsible for running the SLAM algorithm implemented in the report.

3.2.3 Software

3.2.3.1 ROS-Kinetic

The robotic operating system(ROS)[36] is a open-source meta operating system. ROS-kinetic is the version supported on Ubuntu 16.04, and is used as the framework for all the software developed for Manta.

In a ROS system different instances of code is run as *Nodes*. A ROS *Master* registers all *Nodes* that runs on the system and initiates communication between different nodes. After communication has been established the rest of the communication between the nodes happens through peer-to-peer communication. There are multiple methods for *Nodes* to communicate, the three most common are *topics*, *actions* and *services*.

ROS *topics* is asynchronous method of communication, where one or multiple *Nodes* publishes messages to a topic, and one or multiple *Nodes* subscribes on the topic. The subscribers will then receive all messages published on the topic. ROS *actions* will publish messages to a topic continuously like ROS *topics* but only during certain predefined states. When a signal indicates that the state is no longer active the *action* will stop publishing and stay idle. Lastly ROS *services* works as a service, a *Node* must manually request the service and then the service is performed. ROS *topics* are great for information that is necessary at all times like data from the IMU. ROS *actions* are great for computationally heavy tasks that have to be run for a certain amount of time but do not have to be active at all times, like a convolutional neural network. ROS *services* are great for tasks that do not have to be performed continuously.

The *Nodes* are communicating with ROS messages. There are a lot of standard ROS messages, but one can also create custom messages. The messages can carry most data types and allows for free communication between *Nodes* running code from different programming languages. ROS supports C++, Python and Lisp, but are working on implementing a lot more of the modern programming languages. In Mantas system both python and C++ *Nodes* are run.

3.2.3.2 Gazebo - Simulator

Gazebo[33] is a open-source 3D robotics simulator. It provided a robust physical engine, good graphics and the possibility to import custom robotic models and sensors. There is a ROS package called *gazebo_ros_pkgs* that wrapps around the stand-alone Gazebo simulator and provides interfacing between Gazebo and ROS. This allows for simulation of robots in Gazebo using ROS messages. Vortex NTNU is using the popular Unmanned Underwater Vehicle Simulator(UUV Simulator)[37] plugin in Gazebo to simulate Manta. The forward pointing camera and forward pointing sonar provided by the UUV simulator are used to simulate Mantas exteroceptive sensors. A snapshot of the simulator an be seen in figure 3.7

Figure 3.6: A snapshot of Vortex NTNUs simulator

The clear underwater conditions of Gazebo and geometrical shapes provides ideal conditions that can not be replicated in the real world. However, Gazebo provides a great platform for prototyping and testing of the overall system.

3.3 Software implementation

The software in this report is created with the end-goal of providing Manta with spatial awareness and a more robust state-estimation for Robosub2020. The end goal can be split into three sub-goals: 1. Implement a EKF-SLAM system that is able to reduce the drift that would build up in a 20-minute Robosub run, 2. Real time creation of a map which can be used for path- and mission planning. 3. Combine a neural network, camera and sonar data to place objects of interest detected in the camera frame into a 2D-position on the map.

3.3.1 Navigation frames

The implemented system consists of four frames, map, sonar, dvl and bodyon of the AUV, it is the pose of the body frame with regards to the map frame that are estimated. The sonar and DVL frame are sensor frames also moving with the AUV correlating to the positon and orientation of the sonar and DVL respectively. The static transformations between the DVL, sonar and body-frame is important to take into consideration in order to properly process the data they collect.

3.3.2 Mapping in an occupancy grid

An occupancy grid is a 2D grid with a set resolution which can be used as a map. Each cell of the grid has a allocated number representing the probability of occupancy. The SLAM approach implemented in this report is in 2D space which makes an occupancy grid ideal for representing the map.

In this implementation the possible probabilities in a cell will be 100% meaning the cell are occupied, 0% mean the cell are unoccupied, and lastly -1 if the cell is not yet observed by the AUV. This is done to simplify as the extra complexity needed for other probabilities are not prioritised in this report.

The sonar is the natural choice to create a 2D-map as a scan from a forward pointing imaging sonar gives a 2D representation of the area in front of the sonar head. However, all the information in a scan is not useful for mapping as for example echos and reflections could create obstacles where in reality there are none. Here the useful information from each sonar scan is considered the range to the first local maximum over a set threshold in each beam of the sonar. If no bins exceeds the set threshold in a beam no information is extracted for that particular beam. The data is processed in this fashion as the first local maximum should be the bin that most likely correlates with the first substantial object the sonar hits and is therefore relevant for the map. Other local maximums in the beam may be correlated to other objects, but might also be reflections and echos.

By combining the range to the first local maximum with the current pose estimation of the AUV a map of the surroundings can be created by converting the ranges from the sonar frame to the map frame. By setting the cells in the occupancy grid correlating with the position of the best bins to occupied and all cells between that cell and the AUV to unoccupied a map can be created in real time. This would be a naive mapping approach as it would require perfect sensors for the map to be correct which do not exist in the physical world.

However, in this report a mapping algorithm taking the first local maximum bins and placing them on an occupancy grid will first be implemented in a simulator where the sensor data can be assumed perfect. This is done as an experiment to verify that the mapping algorithm are filling occupancy gird correctly. Thereafter a mapping algorithm using the same process of filling in the occupancy grid, but rather than using the entire sonar scan, only uses the bins correlated with the landmarks of the SLAM algorithm. Unlike the deterministic map created by the perfect sensors, this would allow for correction of the map when adjustments are done to the landmarks estimated position.

3.3.3 Line extraction

The line extractor is implemented very similar to how it is described in [27], and were thoroughly reviewed in Section 2.6.4 of this report and will not be repeated in its entirety here. Instead a short review of the entire pipeline will be presented in the next paragraph before a small discussion on the minor changes done to the algorithm is discussed.

When a new scan arrives it is immediately processed by using a threshold to reduce noise and only extract the data that most likely corresponds to objects. The thresholded data is then used to find the local maximum bins in each sonar beam. These are the bins that most likely represents the area where the object is located. The processed data is then used in the modified Hough Line Transform. It is important to note that the data should not be considered simple points, but rather arches as the sound waves expand in the water. The α and β values used for equations 2.45 and 2.46 respectively can be seen in table **??**. The best lines resulting from the modified Hugh transform are used in the EKF-SLAM algorithm to update positions or add new landmarks. In addition to the best lines the thresholded scan used to extract the data most likely to corresponds to objects are also stored and saved for uncertainty calculations in the EKF-SLAM algorithm.

$\boldsymbol{\alpha}$	0.020943951
$\boldsymbol{\beta}$	0.52359877

There are some minor modifications. In [27] a 360 degree mechanical imaging sonar were used, the AUV in this report utilizes a forward looking sonar. As a result there are no need for a buffer to keep track of old scans consisting of the "last 180-degrees". In this implementation all scans are processed.

3.3.4 EKF-SLAM

In this report the implemented EKF-SLAM algorithms estimations will be compared to an EKF. The EKF of choice is the EKF present in the robot localization package for ROS[21].

The EKF-SLAM implemented is a 2D-SLAM with landmarks stored as polar coordinates. When the system launches all states are set to zero, and landmarks are stored regards to this initial position. Since this is considered a 2D problem, only the position in the 2D-plane and the rotation of the agent on this

plane is of interest. The state vector $\boldsymbol{\eta}$ can be written as:

$$\boldsymbol{\eta} = \begin{bmatrix} \boldsymbol{x} \\ \boldsymbol{m} \end{bmatrix}, \ \boldsymbol{x} = \begin{bmatrix} x \\ y \\ \psi \end{bmatrix}, \ \boldsymbol{m} = \begin{bmatrix} \rho_0 \\ \theta_0 \\ \vdots \\ \rho_n \\ \theta_n \end{bmatrix} \tag{3.1}$$

Where x, y and Ψ are the x-position, y-position and rotation in yaw of the AUV respectively, and ρ_i and θ_i are the range and angle of the landmark i with regards to the starting position.

3.3.5 Prediction

The odometry prediction is done by utilising the linear velocity measurements from a DVL and the angular velocity measurements of a gyroscope. The sensors contains no in formation about the landmarks contained in $\boldsymbol{m}$, and the prediction step will not be affected the landmarks. Taking the velocity measurements as odometry input the prediction is fairly straight forward:

$$\hat{\boldsymbol{x}} = f(x, y, \psi, u, v, \omega) = \begin{bmatrix} \hat{x} \\ \hat{y} \\ \hat{\psi} \end{bmatrix} \tag{3.2}$$

$$\begin{aligned} \hat{x} &= x + cos(\psi)u\Delta T + sin(\psi)v\Delta T \\ \hat{y} &= y + sin(\psi)u\Delta T + cos(\psi)v\Delta T \\ \hat{\psi} &= \psi + \omega\Delta T \end{aligned} \tag{3.3}$$

Where u and v are the linear velocities in the x- and y-axis, and ω is the angular velocity.

The predicted covariance $\mathbf{P}$ will similarly also just affect the non-landmark states and is calculated as

$$\mathbf{P}_{k|k-1} = \mathbf{F}\mathbf{P}_{k-1}\mathbf{F}^T + \mathbf{G}\mathbf{Q}\mathbf{G}^T \tag{3.4}$$

where $\mathbf{P}_{k-1}$ is the estimated covariance from the previous step and $\mathbf{F}$ is the derivative of the odometry prediction. $\mathbf{Q}$ are the process noise and $\mathbf{G}$ are a matrix to expand $\mathbf{Q}$ to the same size as $\mathbf{P}$. As the odometry prediction do not contain any information about the landmarks $\mathbf{G}\mathbf{Q}\mathbf{G}^T$ will be zero for all rows and columns expect the three first corresponding to the states in the system. The chosen Q-matrix is the same that was used during Robosub2019.

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & -sin(\psi)u\Delta T + cos(\psi)v\Delta T \\ 0 & 1 & cos(\psi)u\delta T - sin(\psi)v\Delta T \\ 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

$$\mathbf{Q} = \begin{bmatrix} 0.05 & 0.0 & 0.0 \\ 0.0 & 0.05 & 0.0 \\ 0.0 & 0.0 & 0.02 \end{bmatrix} \tag{3.6}$$

3.3.6 Update

The update is done by the standard EKF-equations.

$$\begin{gathered} \widehat{\mathbf{z}}_{k|k-1} = \mathbf{h}(\widehat{\mathbf{x}}_{k|k-1}) \\ \boldsymbol{\nu}{=}\mathbf{z}_k - \widehat{\mathbf{z}}_{k|k-1} \\ \mathbf{H}{=}(\delta/\delta\mathbf{x}_k)\mathbf{h}(\mathbf{x}_k) \\ \mathbf{S}_k = \mathbf{H}\mathbf{P}_{k|k-1}\mathbf{H}^T + \mathbf{R} \\ \mathbf{W}_k = \mathbf{P}_{k:k-1}\mathbf{H}^T\mathbf{S}_k^{-1} \\ \widehat{\mathbf{x}}_k = \widehat{\mathbf{x}}_{k:k-1} + \mathbf{W}_k\boldsymbol{\nu}_k \\ \mathbf{P}_k = (\mathbf{I} - \mathbf{W}_k\mathbf{H})\mathbf{P}_{k|k-1} \end{gathered}$$

However the process of predicting landmarks $\widehat{\boldsymbol{z}}_{k|k-1}$ and finding the innovation $\boldsymbol{\nu}$ are a little more complex than in a normal EKF. When predicting landmarks, the landmarks stored with regards to the map frame

will have to be converted into the sonar frame. Since the landmarks are stored as lines, the conversion is done by finding the point on a line corresponding to a landmark that is closest to the sonar-head in the map-frame, and converting it to a range ρ and angle θ in the sonar frame.

This is done by first finding the point in cartesian coordinates closest to the center of the map-frame:

$$X_{map} = \rho_{landmark} cos(\theta_{landmark}) \tag{3.7}$$

$$Y_{map} = \rho_{landmark} sin(\theta_{landmark}) \tag{3.8}$$

The line of interest is 90° on the line from the centre of the map frame through the point (X_{map}, Y_{map}) and can then be parametrized as:

$$Y_{line} = B_1 X_{line} + C_1 \tag{3.9}$$

where $B_1 = tan(\theta_{landmark} + \frac{\pi}{2})$ and $C_1 = Y_{map} - tan(\theta_{landmark} + \frac{\pi}{2}) X_{map}$.

The point of interest on the line is the point closest to (X_{sonar}, Y_{sonar}), that point will be in the intersection between the first line, and a line perpendicular to the first line going through (X_{sonar}, Y_{sonar}). The slope of this line must be the reciprocal and opposite slope of the first line:

$$B_2 = -\frac{1}{B_1} \tag{3.10}$$

and the constant is given by:

$$C_2 = Y_{sonar} - B_2 X_{sonar} \tag{3.11}$$

With the new slope and constant the perpendicular line equation can be written as :

$$Y_{line} = B_2 X_{line} + C_2 \tag{3.12}$$

This is an equation system with two equations and two variables which are solvable, the solution will be the intersection between the lines which is the point of interest. The point can then be converted into polar coordinates in the sonar frame:

$$\rho_{prediction} = \sqrt{(X_{closest} - x)^2 + (Y_{closest} - y)^2} \tag{3.13}$$

$$\theta_{prediction} = cos^{-1}(\frac{X_{closest} - x}{\rho_{prediction}}) - \psi \tag{3.14}$$

where x and y are the positions in the state vector $\boldsymbol{\eta}$, and ψ are the rotation in η. This process is performed on every landmark in the state vector. A visualisation of the process can be seen in figure **??**.

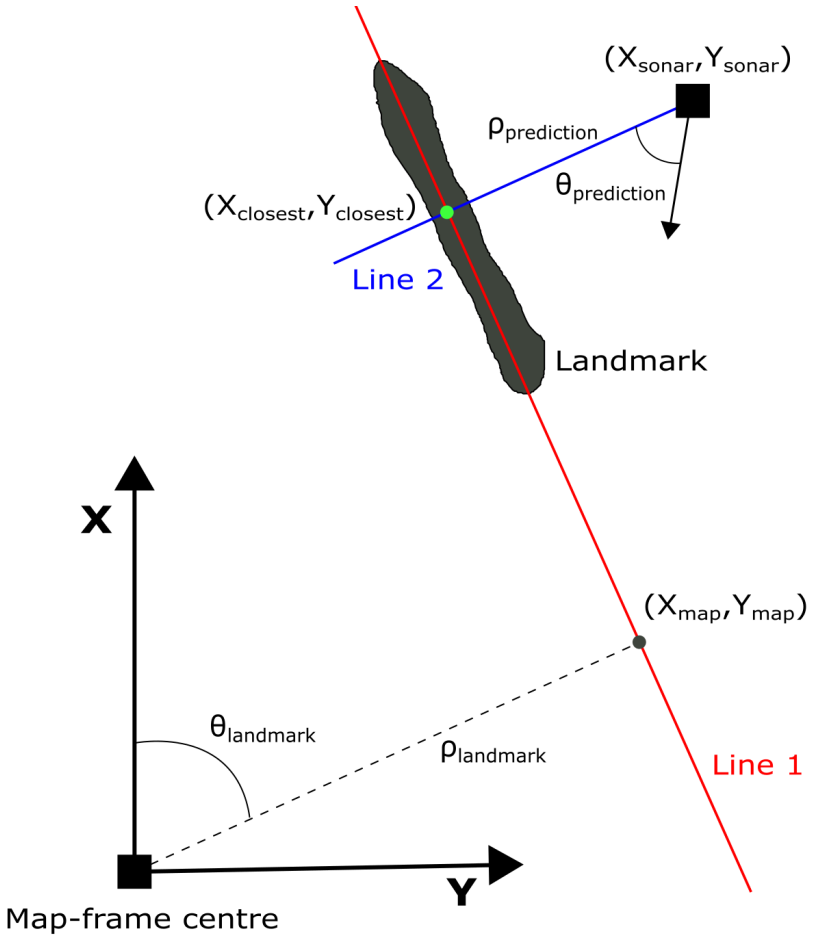


Figure 3.7: A visualisation of the predict landmark process

In the innovation the predicted position of landmarks are compared with the lines found in the current sonar scan. Most of the time only a few of the landmarks will be present in a scan, and an association have to be made between the predictions and the measurements. In this implementation a predicted line is considered possibly associated with a line in the current scan if $\rho_{predicted}$ are within ± 0.3 meters of ρ_{found} and the angle $\theta_{predicted}$ are within ± 0.3 radians of the θ_{found}. 0.3 meters and 0.3 radians were chosen when working with the experiments as they gave good results, but should ideally rather be based on the uncertainty of the predicted landmark.

After all possible associations have been found, the algorithm have to figure out which of the associations that are most likely. In this implementation the best association is considered the association that contains the most amount of landmarks, and are able to pass a gating test. If multiple associations with the same amount of landmarks are able to pass the test, the association with the best test score is chosen. The gating test used in this calculations is done by using the Manhattan distance between the predicted and found ρs and θs. The innovation is then calculated as the difference between the associated pairs of found and predicted landmarks.

$$\boldsymbol{\nu} = \boldsymbol{z}_k - \widehat{\boldsymbol{z}}_{k|k+1} \tag{3.15}$$

All landmarks in the state vector which were not found to have an associated line in the current scan are ignored in the innovation calculations.

3.3.6.1 Add landmarks

If the algorithm were not able to find a suitable association for any of the found lines, they will be added to the state-vector as new landmarks. The $\rho_{landmark}$ and $\theta_{landmark}$ is found by reversing the process of predicting measurements, instead going from the sonar-frame to the map-frame. The found parameters are then added to $\mathbf{m}$ in the state vector.

The initial uncertainty of the new landmark is found by looking at the stored thresholded sonar scan. Here all lines that can fit within the same area as the landmark in the thresholded scan is counted, and

the variance between them is calculated for both ρ and θ. The landmarks covariance with regards to the positions are given by $\mathbf{H}$. Then two new rows and columns are added to the covariance matrix $\mathbf{P}$ and are updated as can be seen in equation 3.16.

$$\mathbf{P} = \begin{bmatrix} \sigma_x & \sigma_{xy} & \sigma_{x\psi} & \cdots & cos(\theta) & 0.0 \\ \sigma_y x & \sigma_y & \sigma_{y\psi} & \cdots & sin(\theta) & 0.0 \\ \sigma_{\psi x} & \sigma_{\psi y} & \sigma_{\psi} & \cdots & 0.0 & 1 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ cos(\theta) & \sin(\theta) & 0.0 & \cdots & \sigma_{\rho_n} & 0.0 \\ 0.0 & 0.0 & 1 & \cdots & 0.0 & \sigma_{\theta_n} \end{bmatrix} \tag{3.16}$$

3.3.7 Tuning

The tuning process of a KALMAN filter consists of finding a suitable $\mathbf{Q}$ and $\mathbf{R}$ matrix. The relationship between the $\mathbf{Q}$ and $\mathbf{R}$ matrix influences how much of an impact the innovation will have on the estimations compared to the odometry prediction. This can be seen by observing that the kalman gain $\mathbf{W}_k$ depends on the relationship $\mathbf{PS}^{-1}$. As mentioned above the values for $\mathbf{Q}$ were set based on the values used at Robosub2019, therefore the tuning process during the experiments conducted in this report was to find a $\mathbf{R}$ matrix that gave good results.

The process of tuning $\mathbf{R}$ where done by starting with a diagonal matrix with 0.000001 in the diagonal terms, and then comparing the EKF-SLAM estimations to the EKF estimations and register the deviation. By increasing the diagonal term by a magnitude and redo the experiment until the diagonal term reached 1000 it became clear what the upper and lower limits for a suitable $\mathbf{R}$ was. It became clear that an diagonal term above 100 lead to very little influence by the SLAM algorithm and the result was very similar to the EKF. When the diagonal term dips below 0.001 the influence from the SLAM algorithm became huge and if a landmark was initialised badly, or the innovation between two landmarks was substantial the update would lead to the estimation of the pose doing a substantial jump. After more testing the sweet spot seemed to be between 0.1-1, and in the end the results presented in this report used diagonal terms of 0.5.

$$\mathbf{R} = \begin{bmatrix} 0.5 & 0.0 & \cdots & 0.0 & 0.0 \\ 0.0 & 0.5 & \cdots & 0.0 & 0.0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0.0 & 0.0 & \cdots & 0.5 & 0.0 \\ 0.0 & 0.0 & \cdots & 0.0 & 0.5 \end{bmatrix} \tag{3.17}$$

3.3.8 Object detection - YOLOv3-tiny

Every year a theme is chosen for ROBOSUB and images of the theme are chosen for the teams to recognize. ROBOSUB2020s theme was set to be Skidoo, the four images to be recognized can be seen in figure 3.8. They will all be one white backgrounds.

Figure 3.8: Images chosen for ROBOSUB2020

YOLOv3-tiny is used on the AUV as it is on of the most accurate networks to be able to run in real-time on a NVIDIA-TX2. On the TX2 mounted in the AUV YOLOv3-tiny is able to achieve on average 18.7 fps. YOLOv3-tiny is run in ROS through an darknet-ROS package[3].

3.3.8.1 Creating and training the dataset

When training a neural network it is important for the dataset to be representative of what is desired for the network to recognize. In this case the objects are simply 2D images, so there are not going to be much variation expect the rotation, size, and the light conditions. The dataset should therefore contain images where the objects are placed at different distances and orientations under different light conditions.

Instead of the time consuming task of physically collecting a dataset a script were written which automatically augmented one or more of the four objects and placed them into a background image. Scaling was used to simulate distance, rotation and skew for simulation of different orientations and inverse and color manipulation for different light condtions. This is a very fast method to collect a large dataset, and for the simple case of recognizing still images on a white background it tends to work well.

As the images are fixed to buoys close to the surface of the pool, it was noticed at Robosub2019 that the images tended to reflect in the surface, creating an upside-down duplicate of the original. It is undesirable for the neural network to detect the reflection and as a precaution the images in the training set were never rotated more than 90°.

3.3.8.2 Result of training

The network had no problems fitting to the dataset, and the validation accuracy quickly surpassed 90%. The loss graph from training can be seen in figure 3.9.

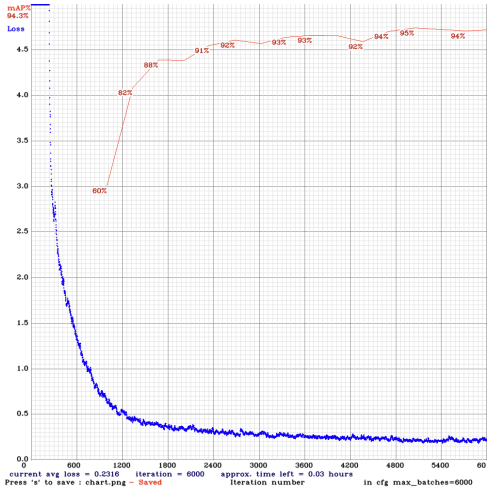


Figure 3.9: Loss graph from training the network

3.3.9 Pixel-width-to-sonar-angle mapping

3.3.9.1 Approach

While light rays travel straight in a uniform medium, they bend in transition between mediums. For an underwater camera the bending will occur in the transition between water-glass and glass-air. This interaction limits the fov of a underwater camera compared to the same camera in air. However, one can still assume that objects at i given pixel-width correlates to an object at a certain angle with regards to the focal point of the camera. As reviewed in the theory section of this report one can theoretically figure out this angle by looking at the camera model. This would require a very good camera calibration and would be quiet time consuming, especially if the camera model is to be combined with the a sonar, as a perfect transformation between the sensors and two perfect sensors models are required for an ideal result.

In this report a more direct approach is suggested. It should be possible to collect a dataset of correlating pixel-width and sonar angle pairs, and then simply fit a polynomial to the pairs. This would allow for a function taking a pixel-width as a function, returning a sonar angle that the object represented by the pixel is most likely to be around. The idea is to use this approach with object detection to first locate an object of interest in the camera frame, then use the average pixel-with on the mapping function to find the sonar angle the object is most likely located. If this works well this approach can be used to place the objects in the map, which can be very valuable for pathplanning when deciding how to approach the object, or mission planning if it is desirable to return to the object at a later stage.

3.3.9.2 Calibration

The calibration is done by collecting a dataset of pixel-width sonar angle pairs and fitting the pairs through a polynomial. In order to perform the calibration a world was created in Gazebo were a red pole were placed some meters in front of a gray wall. By converting the image to HSV-color space and extracting only pixels that are red the average pixel-width of the pole can easily be found. In addition since the pole is placed infront of the wall the sonar-beams that contains the pole will be the beams with high intensity bins closest to the sonar head. See the right part of figure 3.10 for a snapshot of the sonar scan with the first local maximums marked in black and the pole clearly standing out. The average of the sonar angles correlating to the pole will be stored together with the average pixel-width as a pair and added to the dataset. In order to get a good fit it is important to collect pairs from the entire fov of the camera, and have them as evenly spread as possible.

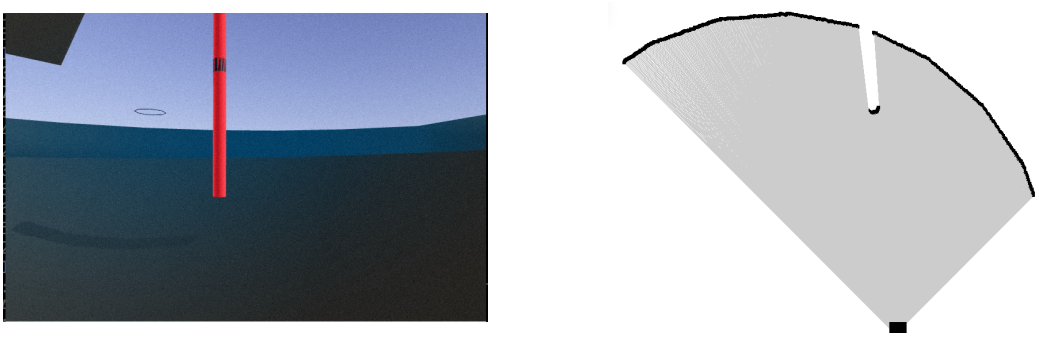


Figure 3.10: Visualisation of the pole in the image frame and sonar frame at an instance during calibration

After collecting a dataset three fitting methods were applied, least-square, 2nd degree polynomial and 3rd degree polynomial. The 3rd degree polynomial performed best and was chosen as method of choice for the experiments. One could most likely get an even better fit with higher degree polynomials, but a 3rd degree gave a good results, and higher degree polynomial might eventually lead to over-fitting on the data. The final calibration parameters used to get the results in this report were:

Variables	Values
a	9.19635425e-10
b	-1.5910143097e-6
c	-0.000685106
d	0.738420211

And the final function mapping pixel-width to beam-angle equals:

$$f(x_{pixel}) = ax_{pixel}^3 + bx_{pixel}^2 + cx_{pixel} + d \tag{3.18}$$

Because of technical difficulties with the AUV during the second pool-test no calibration set were collected for the AUV.

Chapter 4

Experiments

4.1 Simulated experiments

Two experiments were conducted in simulated worlds in Gazebo. The first experiment was created as a verification of the mapping approach used to fill the occupancy grid, and the second is a prototype of the pixel-width to sonar angle mapping.

4.1.1 Maze mapping

The forward pointing sonar and camera in the UUV package were used for this simulation. The sonars max range were set to 10 meters, and the scan angle were set to 100° as the Tritech sonar utilized in the physical experiments. In order to test the mapping algorithm and the compatibility with the path planning algorithms a simulated maze were created. The test were conducted by manoeuvring the AUV through the maze with an joystick. The goal with this experiment was to verify the mapping algorithm, making sure it is building the occupancy grid correctly. The maze can be seen in figure 4.1.

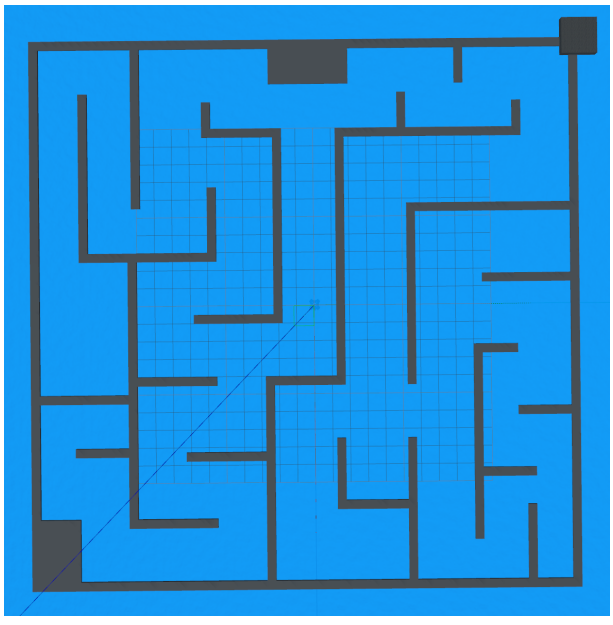


Figure 4.1: Top-down perspective of a simulated maze.

4.1.2 Pixel-beam mapping

The forward pointing sonar and camera in the UUV package were used for this simulation together with the forward pointing camera in the same package. A simulated world were created with all four Robosub2020 images. Two of the images were fastened on a gate in the foreground, while the last two were fastened to bouys next to each other. They are set this way to simulate how they would be at the competition. The goal of this experiment was to prototype the pixel-beam mapping, by seeing if the algorithm was able to place the objects accurately in the occupancy grid.

4.2 Physical experiments

4.2.1 Marine cybernetics laboratory

All the physical experiments were conducted in the Marine Cybernetics Laboratory at Tyholt in Trondheim. It is a part of the Department of Marine Technology at NTNU where it is mainly used for master students and PhD candidates. Its consists of two rooms, a control room and a 60mx6,5mx1,5m basin. The laboratory also provides accurate real-time tracking of surface and underwater vehicles through its Qualsys Motion Caputre System.

4.2.1.1 Qualsys motion camera system

The QUALSYS setup consist of three very accurate infra-red cameras being calibrated to locate small balls in their field-of-view and estimate their motion. By always having at least three balls visible for at least two of the three cameras they can triangulate almost like a GPS system to estimate the trajectory of the cluster of balls, since the distance between the cameras are known. Assuming the balls are fixed on the body with regards to each other Qualsys is able to calculate the full 6-DOF of a body in the systems field of view. During calibration the system calculates the average residuals of the estimation of each of the three cameras, the calculated residuals for the calibraton done before the experiments in this report can be seen in table 4.1

Camera	**Residual(mm)**
1	1.56162
2	1.29991
3	1.51208

Table 4.1: The average residuals of the different Qualsys cameras

As can be observed from the table the average residual are tiny. In this report the trajectory estimated with Qualsys is used as ground truth when comparing to the estimated trajectories of the onboard navigation systems.

Figure 4.2: Shows basin and Qualsys cameras

4.2.2 Artificial landmarks

In order to not only have the walls of the basins as landmarks artificial landmarks were placed in the pool during the experiments. The landmarks consisted of metal pipes, boxes and a huge calibration board. In addition four buoys with the images the CNN should recognize for Robosub2020 were used as can be seen in figure 4.3.

Figure 4.3: Two of the bouys being used as landmarks

4.2.3 Technical difficulties

During march Manta were dismantled as wiring and hardware modifications were underway to prepare it for Robosub2020. This process was interrupted when Norway went into lock-down due to Covid-19 as of 17. march, NTNU were shut-down. When an pool-experiment was set up to the end of April start of May Manta was still in a dismantled state and some quick fixes had to be made to in order to get Manta ready for the experiments.

As a result Manta was not fully operational during the experiments. The thrusters were not operational, and the implementation of the sonar was not up and working with the computers on the AUV. Because of this Manta had to be pushed during the experiments instead of working under its own power. In addition it was not possible to record the sonar data from the AUv, and an external PC had to be used for the recording. This created some problems with synchronizing of the data during pre-processing as the DVL/IMU data and sonar data no longer was recorded on the same system leading to time-stamps no longer matching. However the most unfortunate was a couple of weeks later when a calibration dataset between the camera and sonar was to be recorded. During this experiment the AUV refused to power on, and as a result no calibration set from the physical experiments were collected.

4.2.4 Straight line

For this experiment the AVU were fixed to a bridge that can be pushed on rails along the pool, see figure 4.4. It was desirable to fix it this way in order to minimize travel in all but one direction. Artificial landmarks were placed in line of sight of the sonar. During the experiments the bridge were pushed about five meters in one direction, and then reversed back to the starting point. The length of the straight line were limited by the range of the Qualsys system. An illustration of the experiment set-up can be seen in figure 4.5. The goal of this experiment was to get an indication of how well the EKF-SLAM performs in the absence of any rotations.

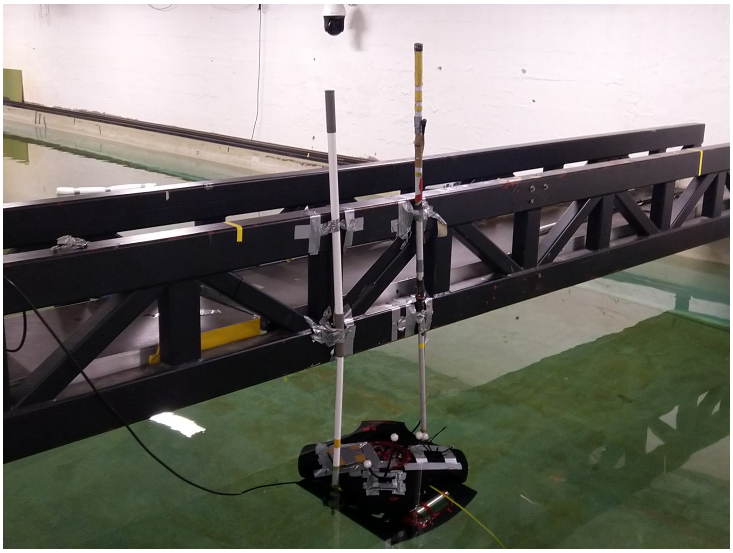


Figure 4.4: Shows the AUV fixed to the bridge

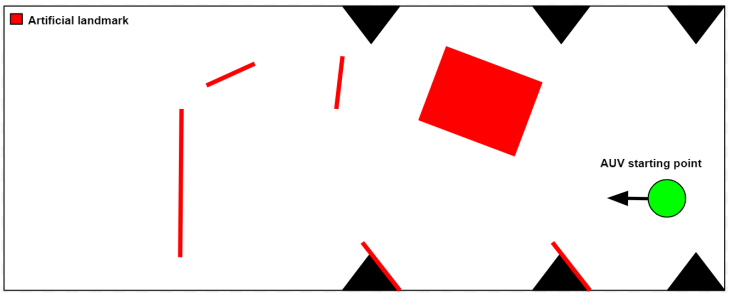


Figure 4.5: The set-up for the line

4.2.5 Random loop

Because of technical difficulties with the thrusters the AUV it had to be physically pushed during this experiment. Artificial landmarks were placed in a random manner throughout the pool and the AUV were pushed in a loop. The placement of landmarks can be seen in figure 4.6. The goal of this experiment was to see how the EKF-SLAM would perform under heavy rotations. In such conditions the shape of the landmarks detected by the sonar will vary a lot compared to simply going in a straight line.

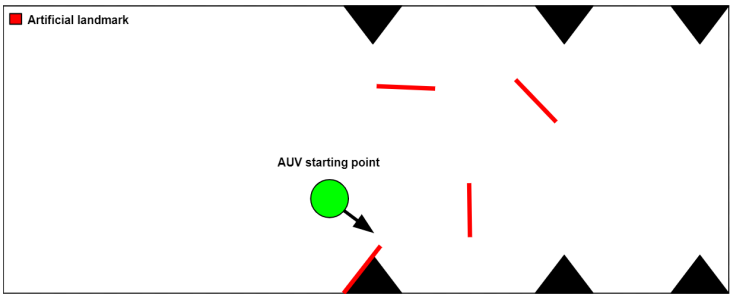


Figure 4.6: The set-up for the loop

4.2.6 Continuous square

As with the random rool the AUV had to be pushed for this experiment. The original plan was to have the AUV going in a 4x5 meter continuous square for 20 minutes, but because the AUV had to be pushed it was difficult for the person pushing the AUV to not go in front of the sonar and/or the Qualsys camera system when manoeuvring multiple squares. As a work around the AUV were pulled backwards when necessary and consecutive squares were traversed in opposite directions.

Less artificial landmarks were used in this compared to the other experiments, as the AUV would be able

to use the walls of the basin for most of the experiment. The placement of landmarks can be seen in figure 4.7. The goal of this experiment is to see if the EKF-SLAM would be able to reduce the drift that would build up in an EKF during a 20 minute experiment.

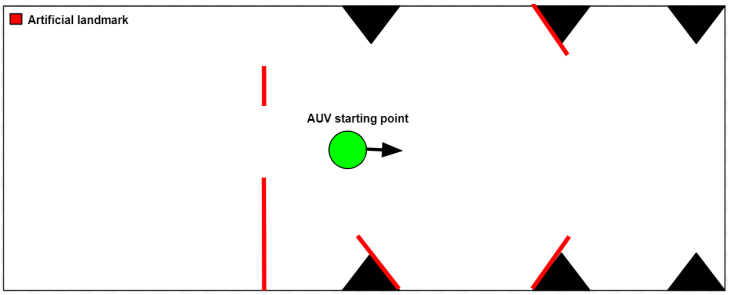


Figure 4.7: The set-up for the loop

Chapter 5

Results

5.1 Part 1 - Simulation

5.1.1 Occupancy grid

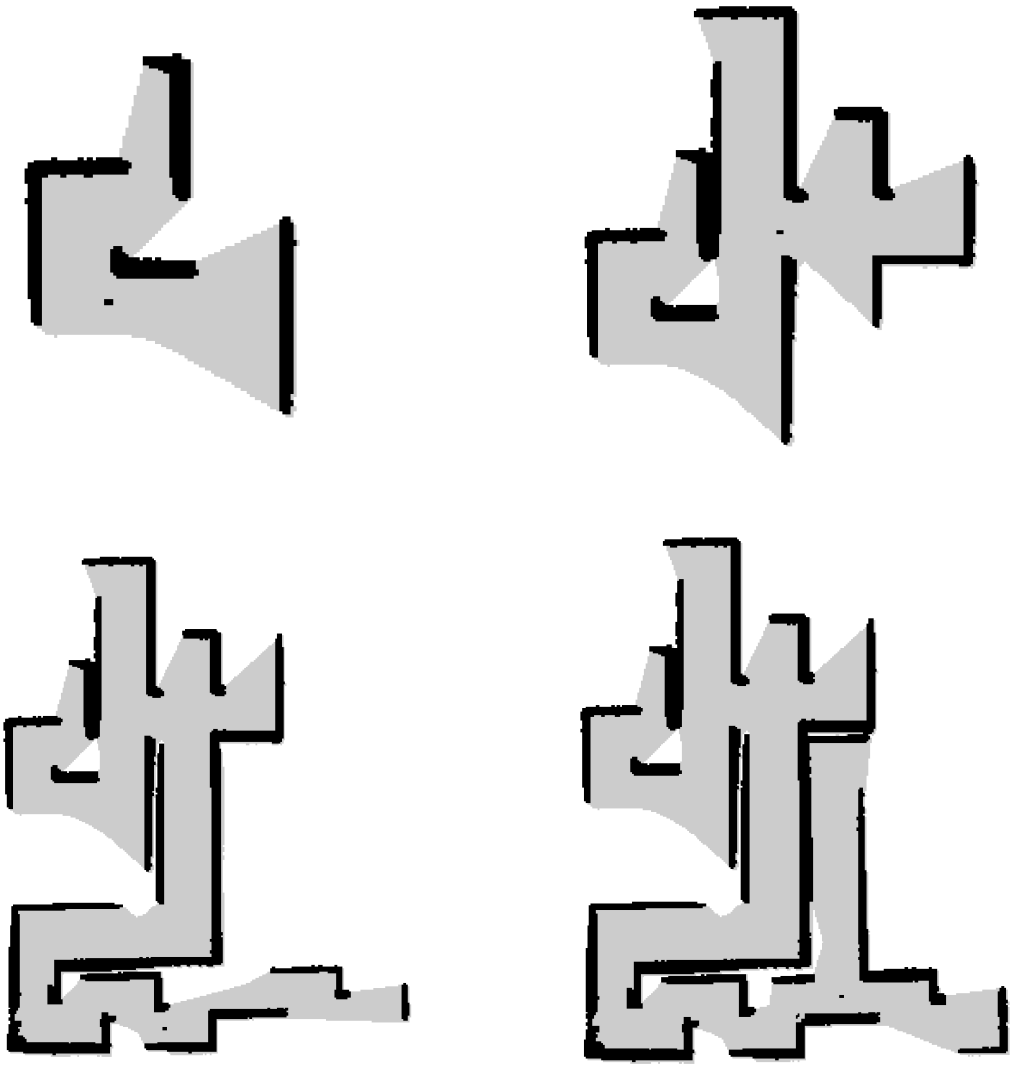


Figure 5.1: Visualisation of the occupancy grid resulting from an exploration of the maze in figure 4.1.

5.1.2 Camera and sonar

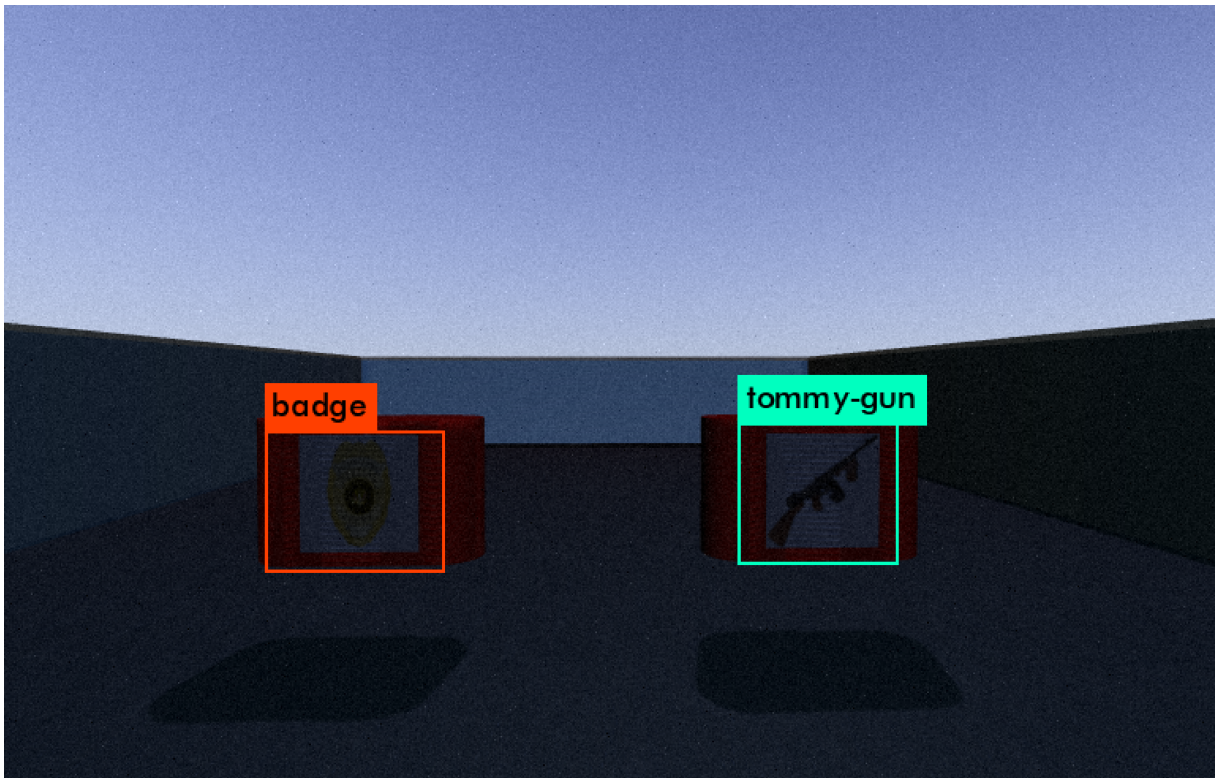


Figure 5.2: YOLOv3 working in the simulator world

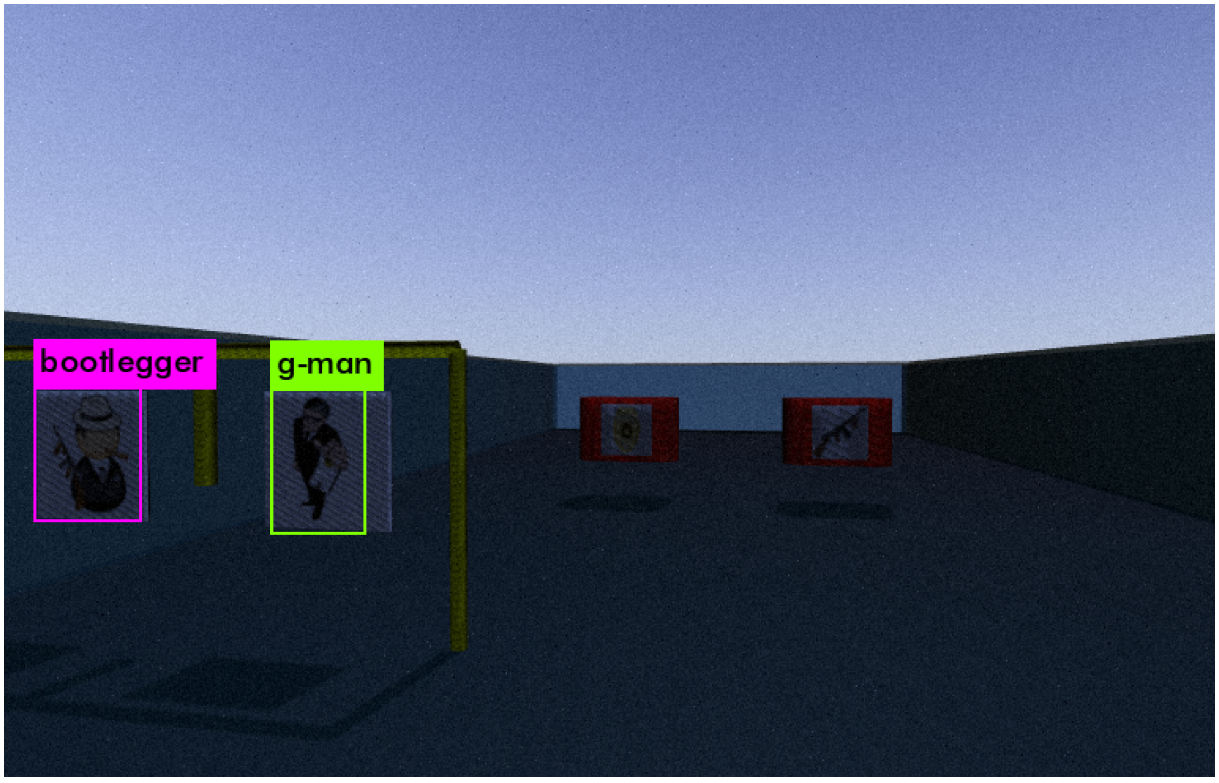


Figure 5.3: YOLOv3 working in the simulator world

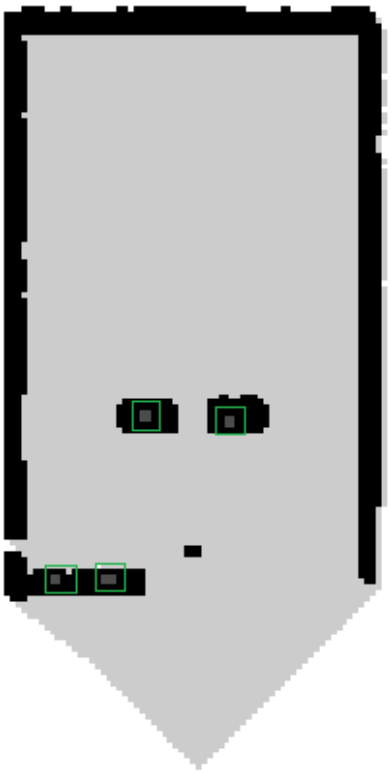


Figure 5.4: Occupancy GRID with CNN landmarks marked as lighter gray squares, here additionally highlighted by green boxes

5.2 Part 2 - Experiments

5.2.1 Straight line

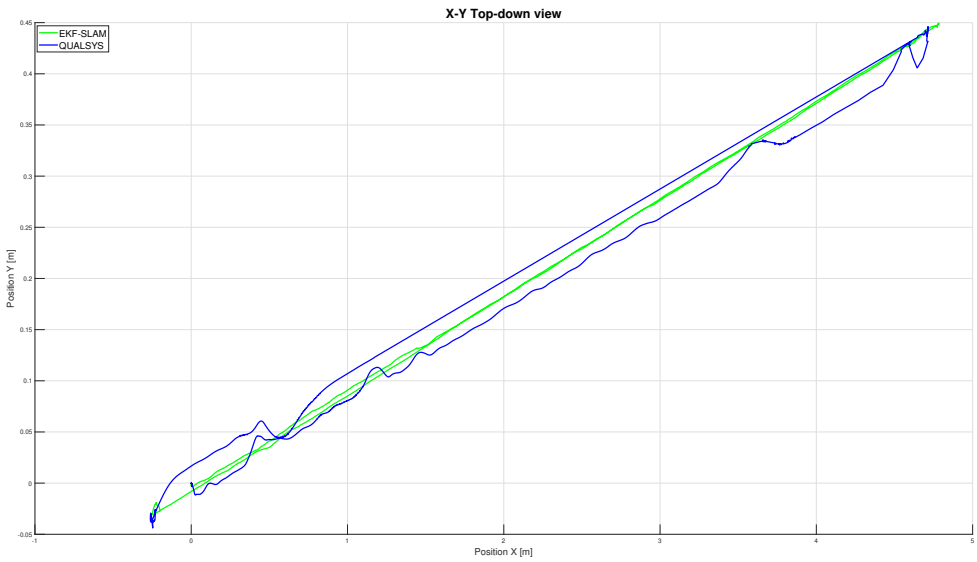


Figure 5.5: EKF-SLAM estimation compared to QUALSYS top-down view

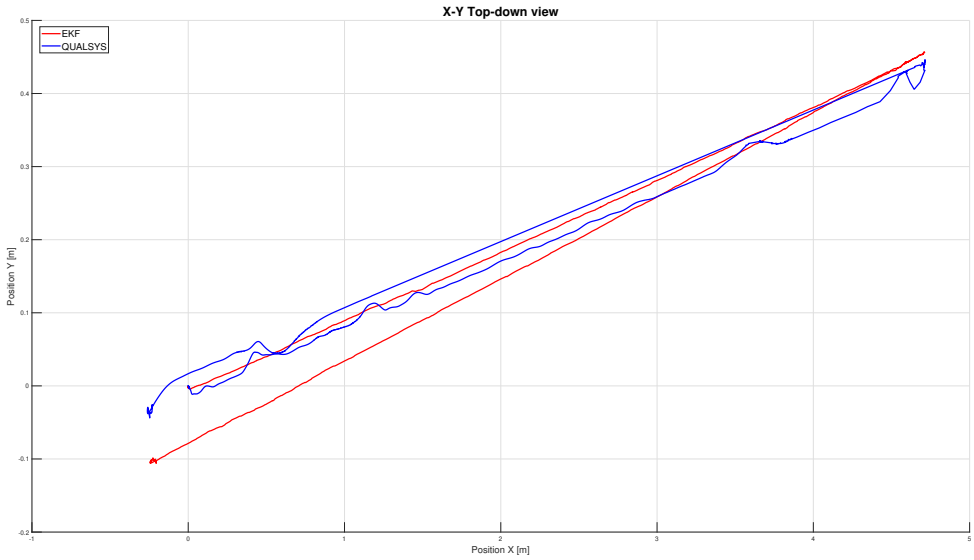


Figure 5.6: EKF estimation compared to QUALSYS top-down view

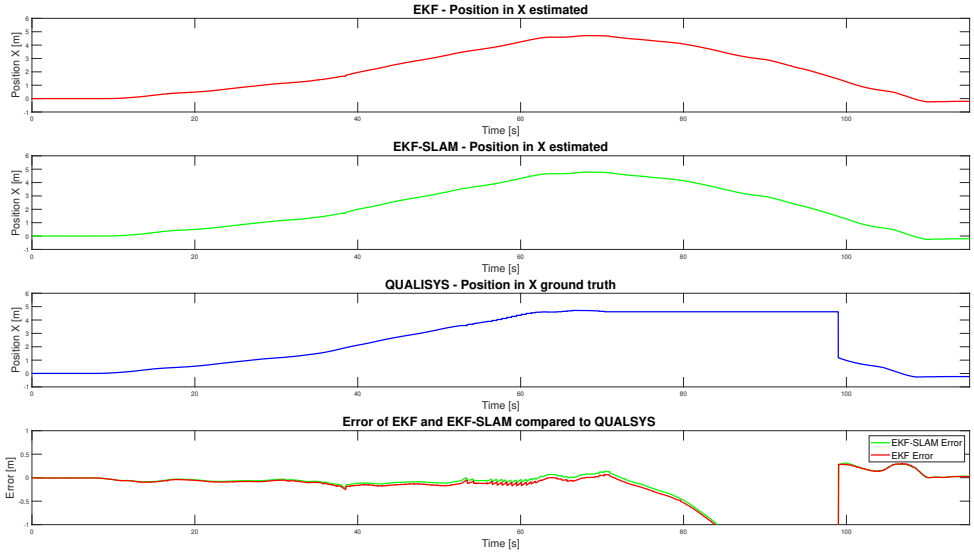


Figure 5.7: Estimated X-position over time for EKF, EKF-SLAM and QUALSYS

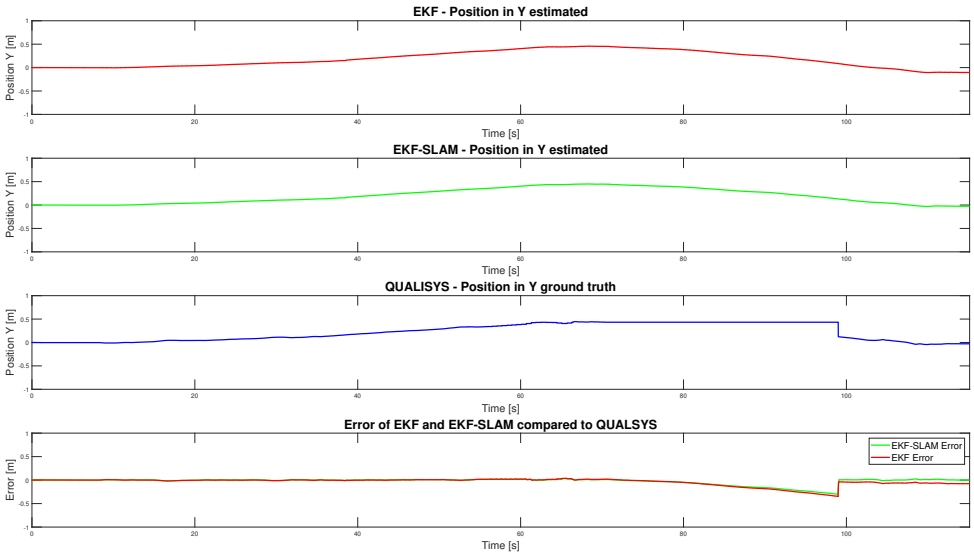


Figure 5.8: Estimated Y-position over time for EKF, EKF-SLAM and QUALSYS

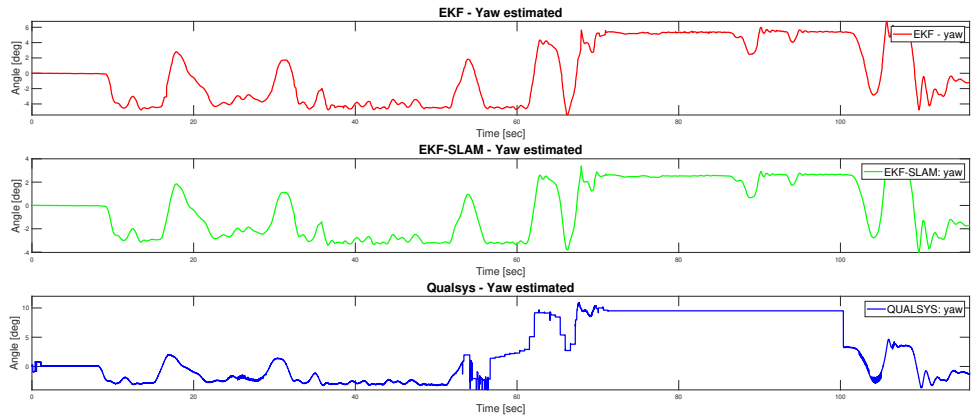


Figure 5.9: Estimated Yaw over time for EKF, EKF-SLAM and QUALSYS

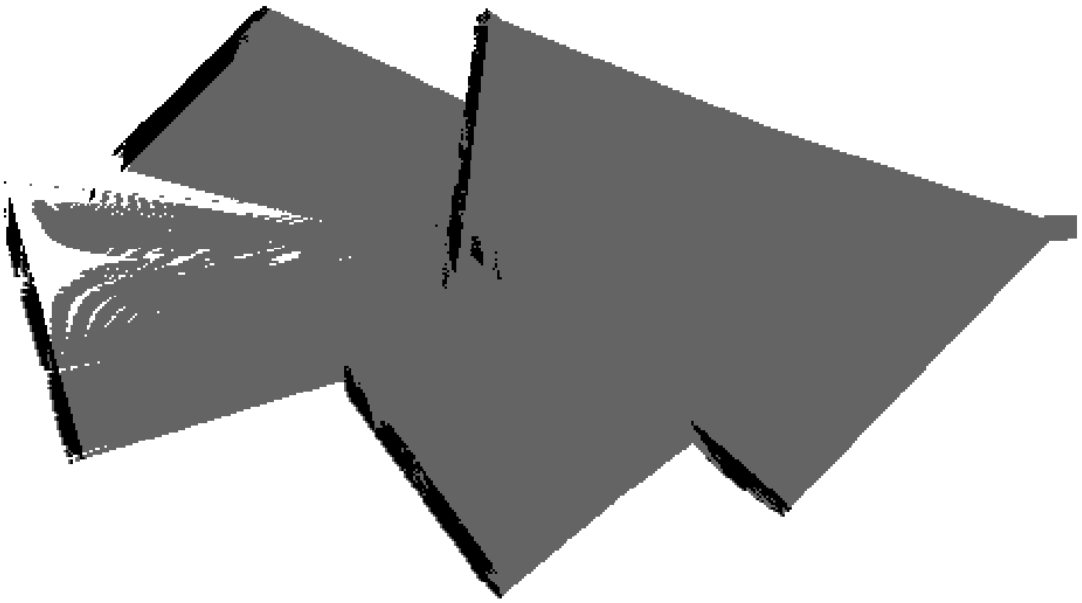


Figure 5.10: Map created of straight line

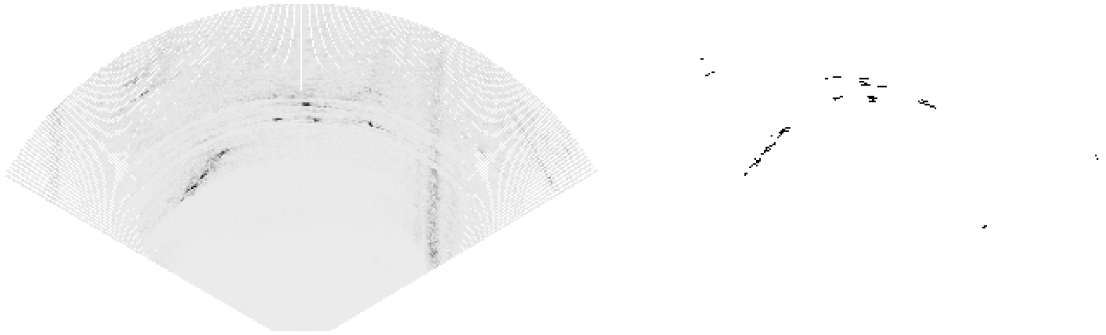


Figure 5.11: Sonar reading of big calibration board

5.2.2 Random loop

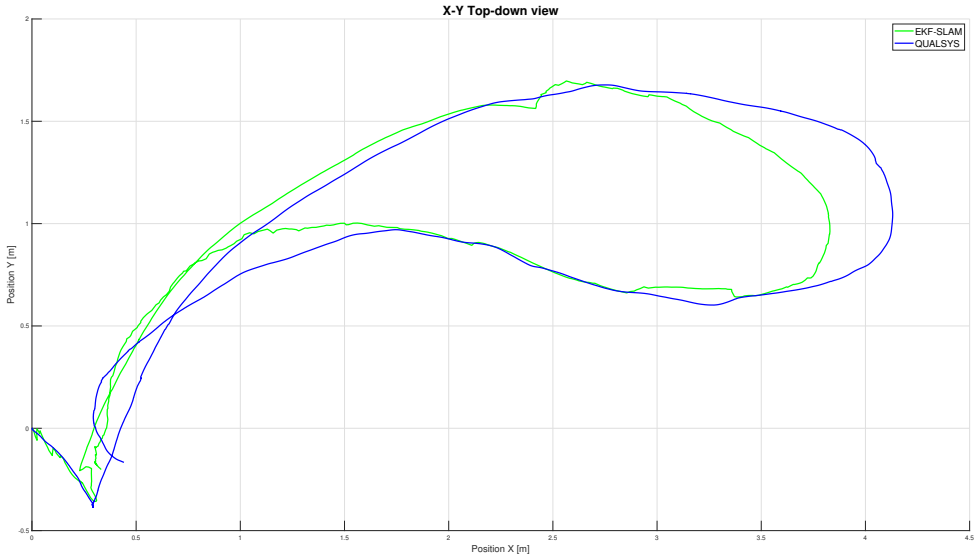


Figure 5.12: EKF-SLAM estimation compared to QUALSYS top-down view

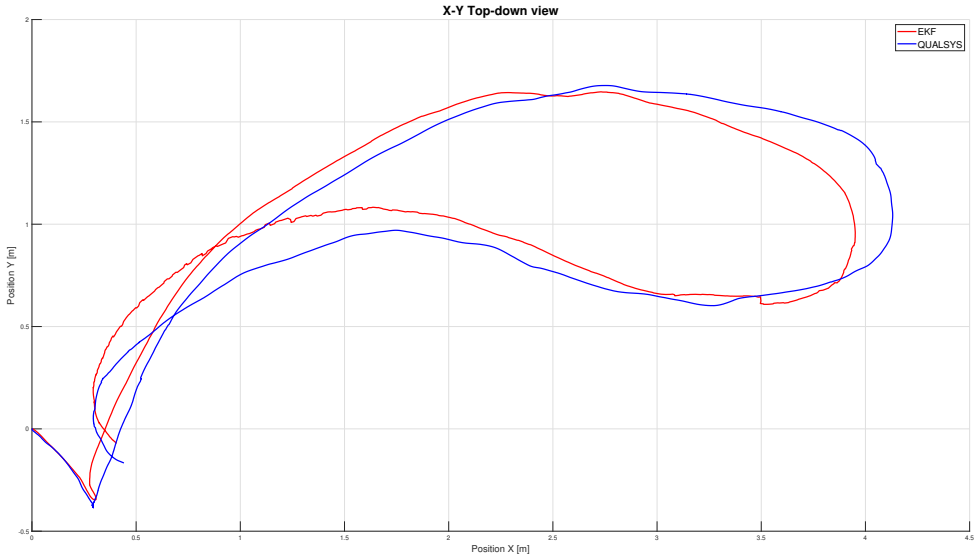


Figure 5.13: EKF estimation compared to QUALSYS top-down view

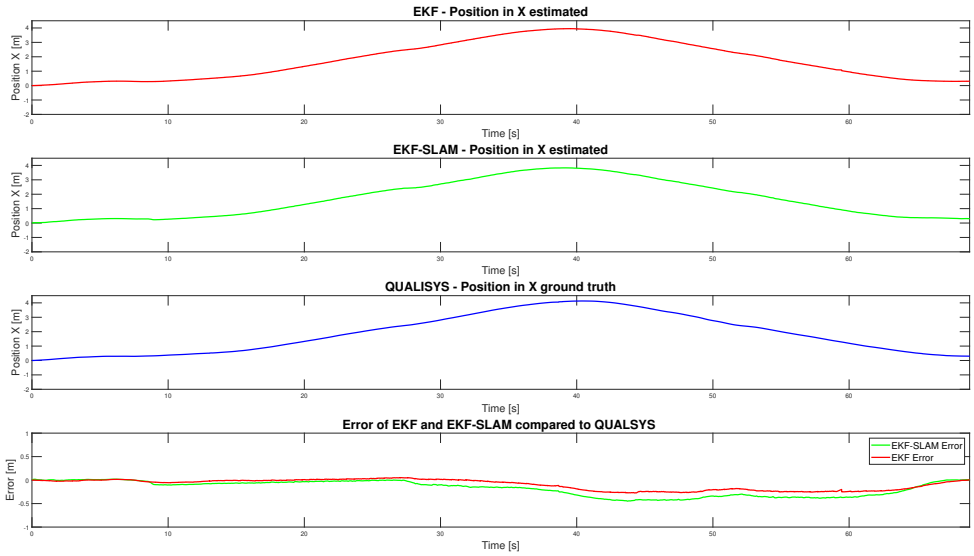


Figure 5.14: Estimated X-position over time for EKF, EKF-SLAM and QUALSYS

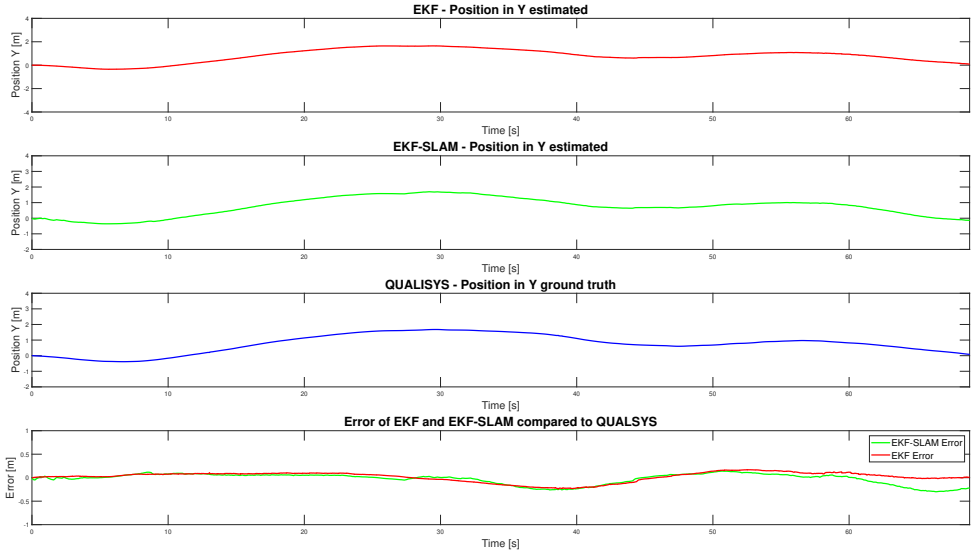


Figure 5.15: Estimated Y-position over time for EKF, EKF-SLAM and QUALSYS

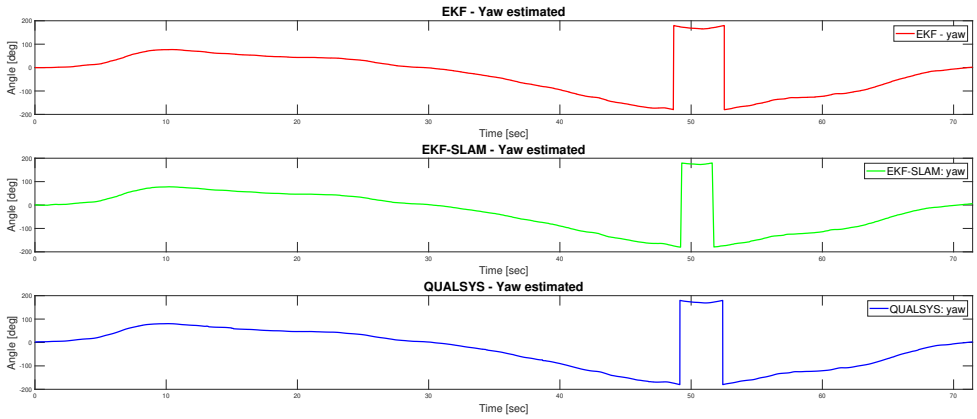


Figure 5.16: Estimated YAW over time for EKF, EKF-SLAM and QUALSYS

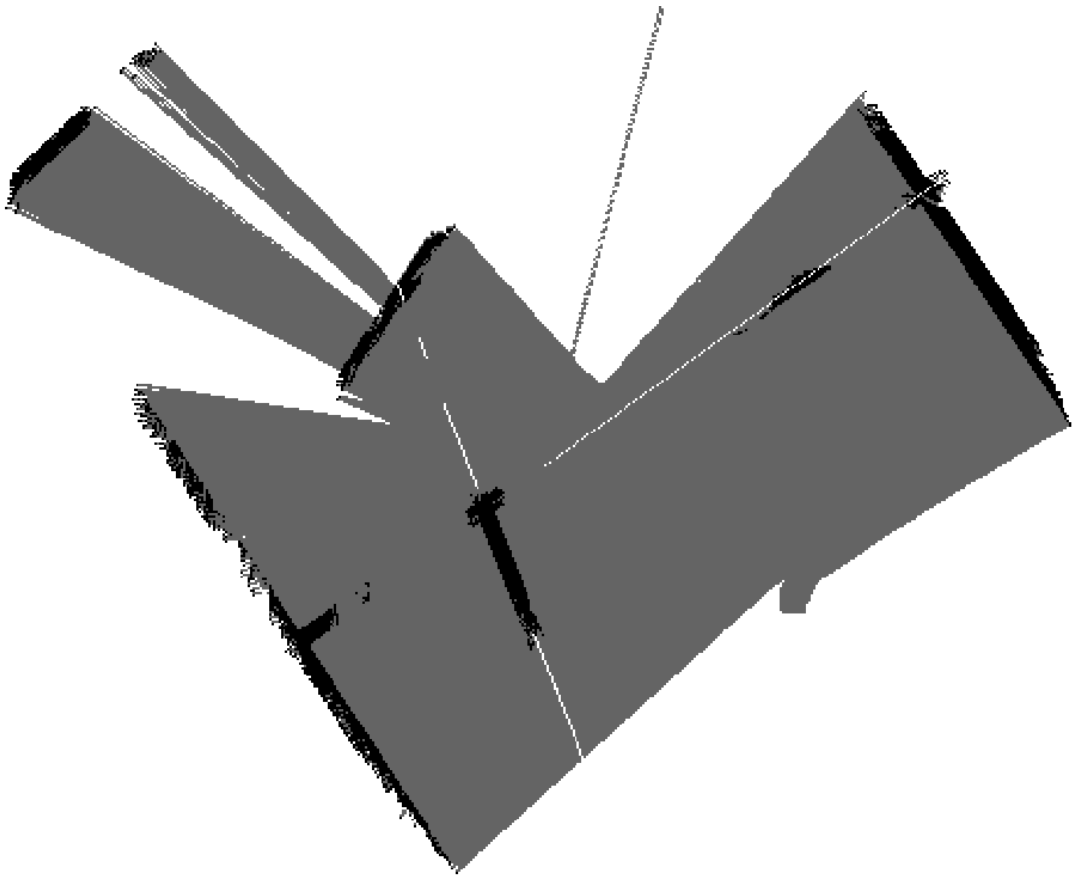


Figure 5.17: Map created of random-loop

5.2.3 20 minutes square

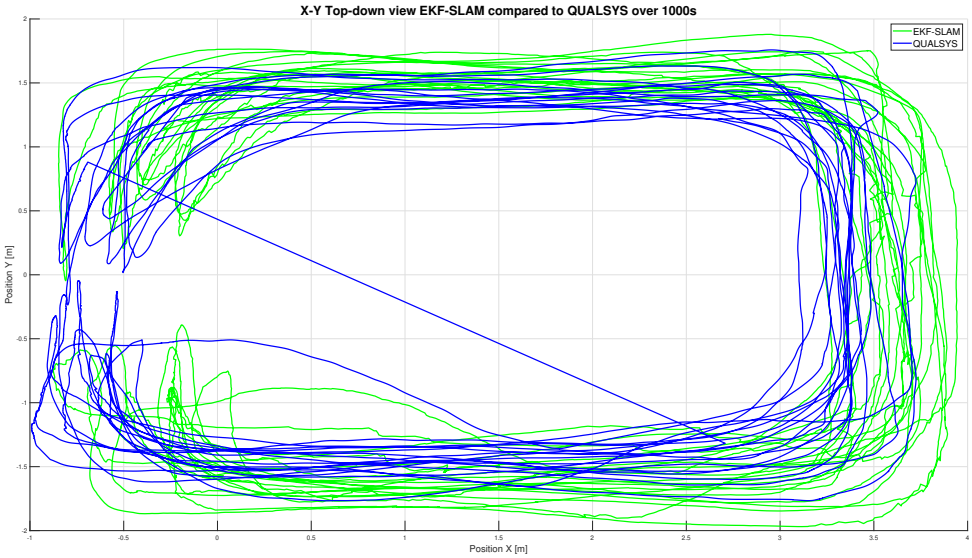


Figure 5.18: EKF-SLAM comparison to QUALSYS

Figure 5.19: EKF comparison to QUALSYS

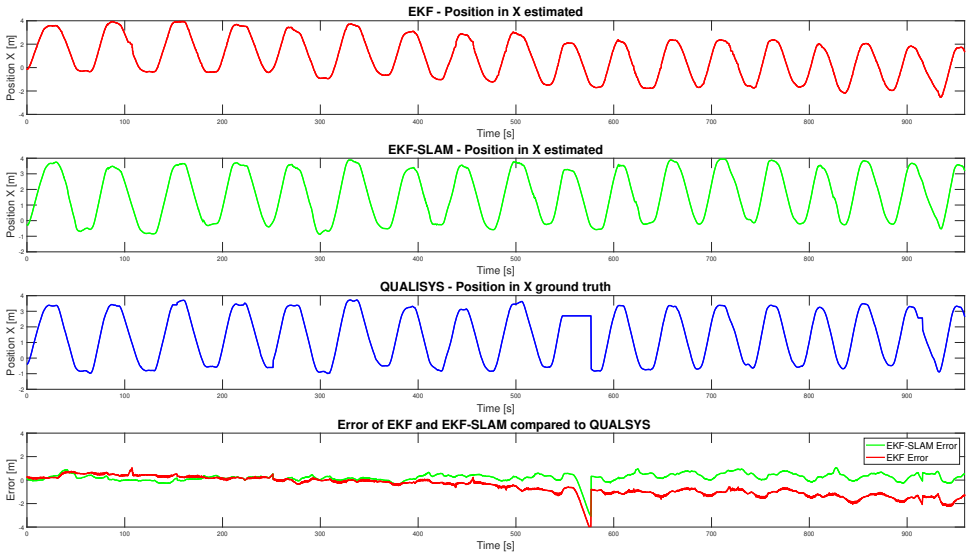


Figure 5.20: Estimated X-position over time for EKF, EKF-SLAM and QUALSYS

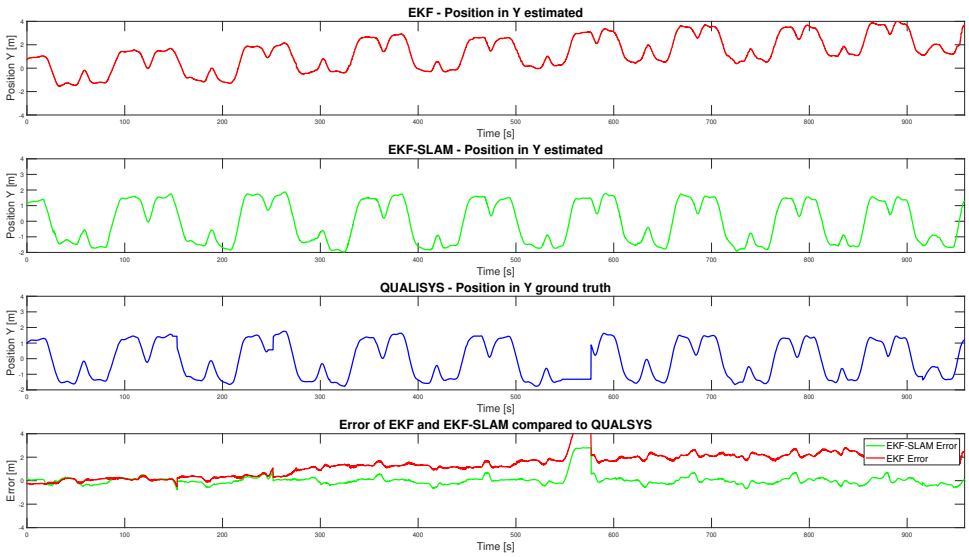


Figure 5.21: Estimated Y-position over time for EKF, EKF-SLAM and QUALSYS

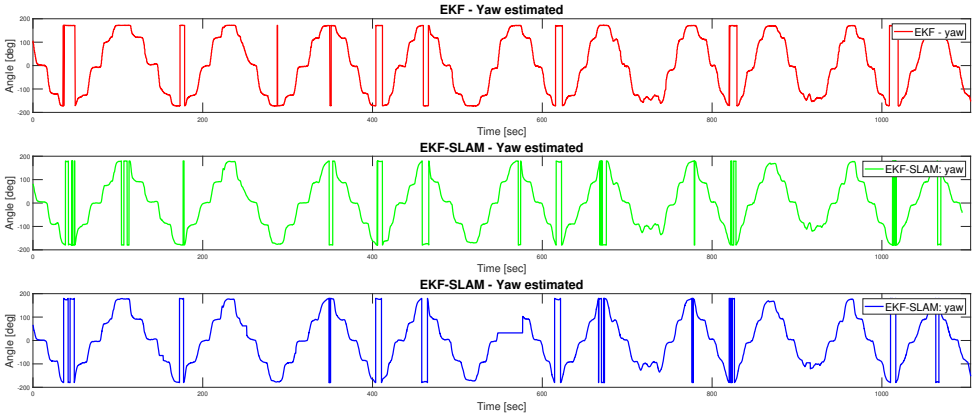


Figure 5.22: Estimated Yaw over time for EKF, EKF-SLAM and QUALSYS

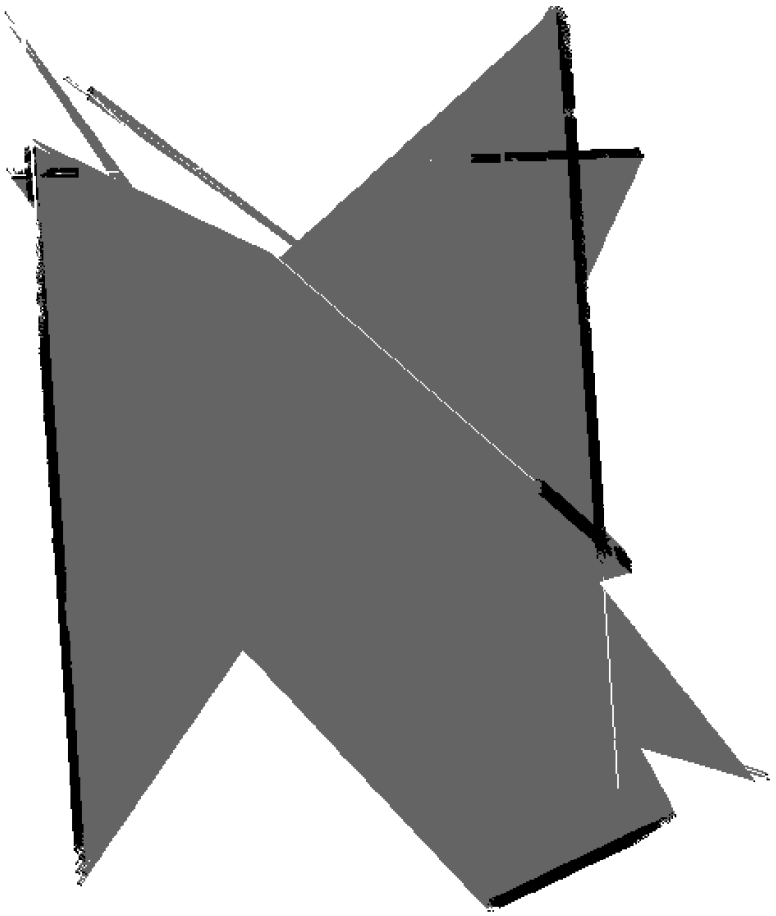


Figure 5.23: Map created of 40min bag

5.2.4 YOLOv3 underwater

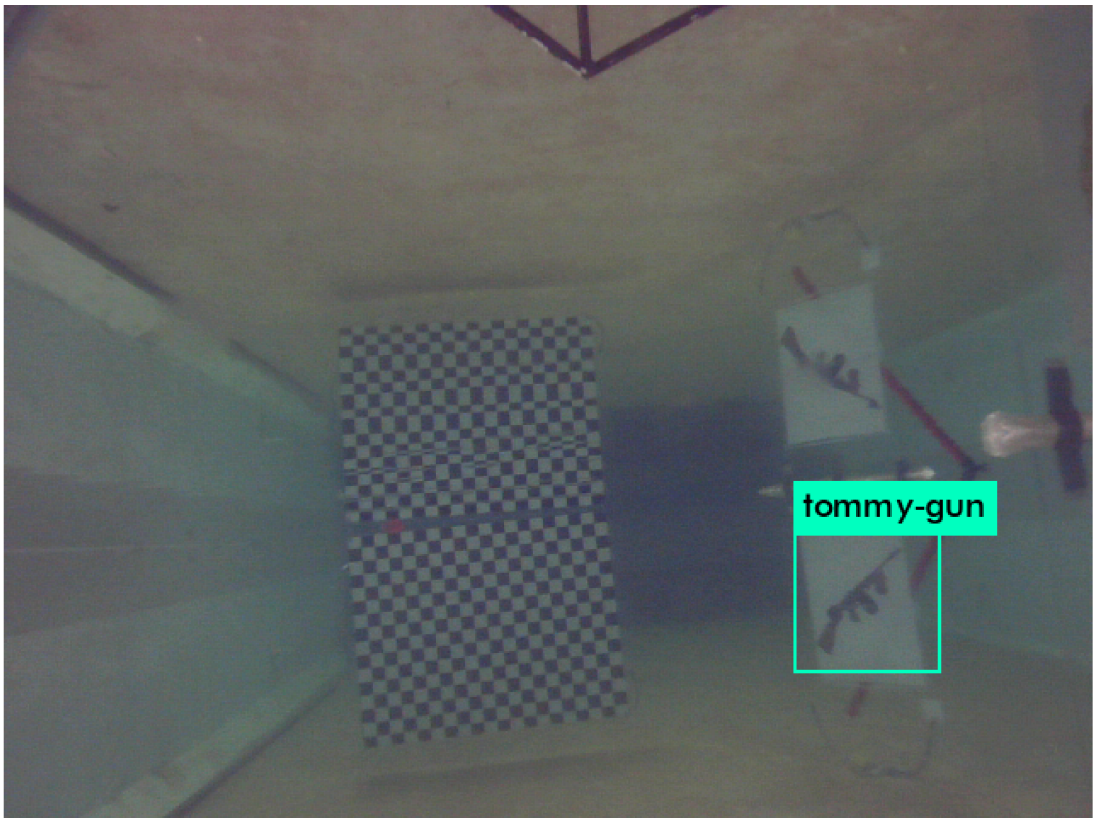


Figure 5.24: CNN results during the straight line test

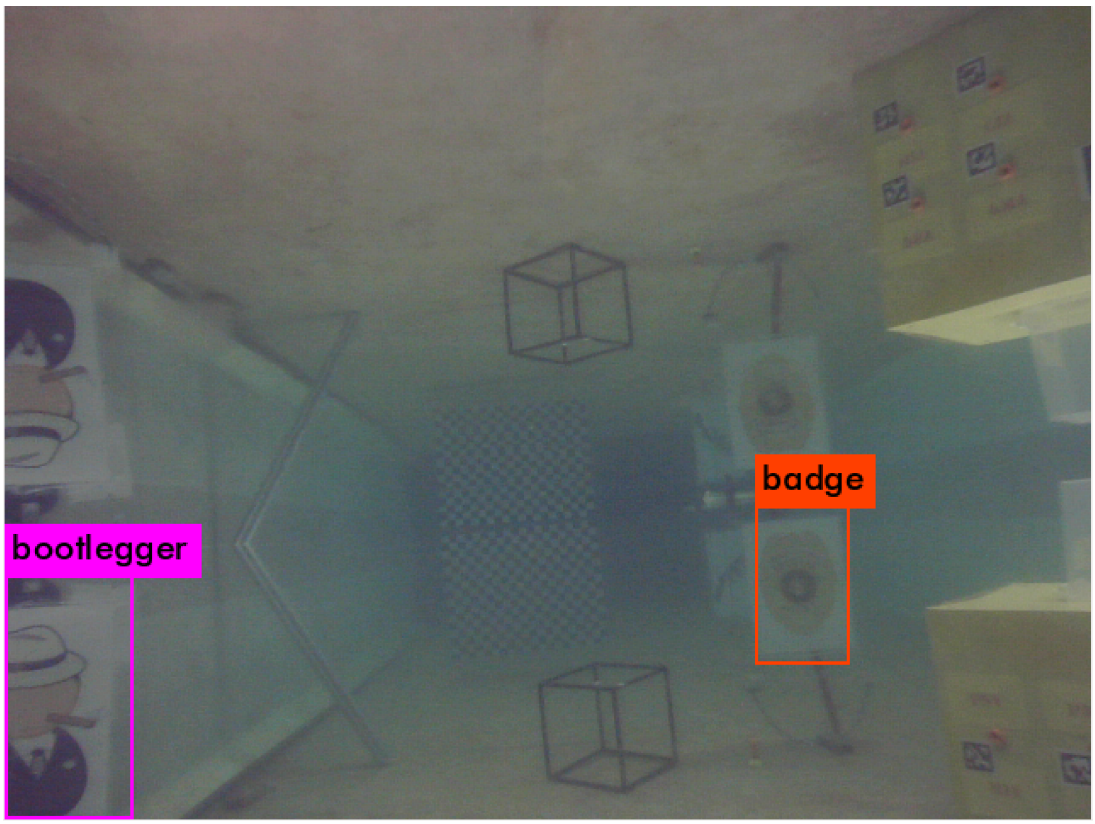


Figure 5.25: CNN results during the straight line test

Chapter 6

Discussion

6.1 Occupancy grid

As can be seen in figure 5.1 the map created by the simulated sonar is a good replica of the maze seen in figure 4.1. Here all the first local maximas over a certain threshold is plotted as obstacles(the black zones), and a line between the bin and the AUV are marked as passable terrain(the gray zones). Areas the system have no information over is visualised as white.

This was done as a prototyping to see that the mapping algorithm worked. However, this map is deterministic, all placed obstacles are permanent. Of course this is not realistic with noisy sensors and erroneous pose estimations, and for the map created by the SLAM algorithm only landmarks will be present in the map. The experiment nonetheless shows that an occupancy grid works well as a 2D-map when using a Sonar, and using the first local maximums seems to work well.

6.2 Pixel-mapping

As can be seen in figure 5.2 and 5.3 the trained YOLOv3 network had no problems locating the images in the simulator, however the bounding boxes somewhat extends outside the image as can especially be seen in the badge detection in figure 5.2. This might cause problems in some situations as the pixel-width is taken as the central pixel of the bounding box.

Figure 5.4 shows a map created with the same mapping algorithm as the maze with the registered location of the four landmarks visualised in the map as gray dots in the black walls highlighted by green boxes. The dots are not completely centralised on the bouys but are still placed close to their actual position. Overall the result of the prototyping in the simulator seemed to work really well. The original plan was to prototype the pixel-mapping in the simulator and if it worked well run experiments to test it in the physical world. However due to technical difficulties with the AUV a calibration set was not collected.

6.3 Straight line

It was discovered in the pre-processing that Qualsys had lost track of the AUVs body when it neared the end-point of the line, and was not able to relocate the body until a substantial amount of the way back was covered. There is however still value information one can get by analyzing the experiment for the stretches Qualsys was able to track.

In figure 5.5 and 5.6 the trajectories estimated by the EKF and EKF-SLAM are compared with regards to Qualsys. This is a simple case where the AUV were fixed to a bridge and pushed forward and backwards with almost no changes in yaw. In this case one would assume that a SLAM approach should be able to out perform the standard EKF as the same landmarks are visible for most of the travel, and all landmarks seen at start-up should also be present at the stop point. By observing the end points of both trajectory one can see that the EKF-SLAM indeed estimates an end position closer to Qualsys than the EKF does.

By looking at the figures 5.7 and 5.8 one can see that the estimated x and y for the EKF and EKF-SLAM follows the same patterns which indicates that the SLAM only performs small adjustments, as one would expect as the EKF estimation should be fairly accurate considering the short duration of the experiment. Looking at the absolute error in position over time one can see that both the EKFs and EKF-SLAMs

have basically an identical error at a duration in the start, but as times goes on the EKF error grows. When Qualsys looses track both errors are naturally getting large.

In figure 5.9 one can find the largest variations between the three estimations. There are some points on the rails the bridge the AUV was fixed to that had more resistance than others. To overcome these spots the bridge had to be pushed more forcefully leading to unintentional rotation in YAW because the poles used to hold the AUV in place slightly bends. This is the reason for peaks in in yaw that can be seen in the figure. One can also observe that the consistent yaw changes when the AUV is pushed backwards from a slightly negative yaw value to a slight positive value. This is due to the poles bending slightly in opposite directions based on which way the AUV is pushed with a constant velocity. Qualsys yaw estimations starts to behave strangely at around 55 seconds until it is able to relocate the body. This is because from around 55 seconds until tracking is lost it switches between two possible body orientations based on the balls fixed to the AUV, which makes it hard to use the yaw estimations in this time period for comparison with the navigation system. From the two navigation systems one can observe that the EKF-SLAM consistently have lower yaw values than the EKF.

The map created can be seen in figure 5.10 it has a good resemblance with the track set visualised in figure 4.5. The big box however is not registered at all in the map. The AUV were fixed to the bridge at its natural buoyancy which is slightly positive, and the sonar was therefore close to the surface, the big box had a gap of about 0.3 meter between its top and the surface. Apparently such a small gap was enough for the intensity of the bins it created to fall below the minimum threshold. Therefore the use of only landmarks used in the SLAM algorithm itself may be missing obstacles which could be a problem if the map is to be used for pathplanning in a structured environment, and a lot of the obstacles in the area do not form any sort of consistent line of high intensity bins. Another deviation is the angle of the large landmark furthest away from the starting point of the AUV. The large landmark were created by using a big calibration board, this board was taller than the depth of the pool and was therefore was not completely vertical, but standing at an angle. It seems like this really affected the sensor readings as instead of a straight line it reads like many different blobs, see the right side of figure 5.11. Compared to the other landmark seen in the image which can be read as a straight line it is understandable that the estimation of the calibration board landmark is not correctly being converted to the map.

Another observation one can do by looking at the map is that the EKF-SLAM is not using the walls of the pools as landmarks in this experiment. This could probably be changed by increasing the β value in the line extraction algorithm which would lead to all bins voting for lines in a bigger sector at the cost of requiring more computational power.

Overall the EKF-SLAM performance in this experiment seems on par with the EKF, with x, y and yaw-estimations following the exact same pattern. However there are some deviations especially in yaw-estimation, and the EKF-SLAM estimates an end-point closer to the end-point of Qualsys estimation than the EKF. There are some issues with only using landmarks when creating a map as obstacles that do not pass the implemented feature extraction are ignored. While this is a limitation for mapping in structured environments it should not hinder the EKF-SLAM algorithms state-estimation. It could also be resolved by using multiple different features extractions in the EKF-SLAM itself, or some other obstacle avoidance system.

6.4 Random-loop

In figures 5.13 and 5.12 the estimated trajectories of the random loop for the EKF and EKF-SLAM can be observed. While both the EKF and EKF-SLAM is able to somewhat follow the shape of the trajectory estimated by Qualsys they both deviates. This was the first experiment were the AUV were pushed by a human, and it lead to some quick-turns and unnatural movement for the AUVs body. A good example of this is the very sharp turn at about the point (-0.4,0.4) in the figure, here both the EKF and EKF-SLAM overestimates the turning and a deviation from Qualsys is created already here in the beginning of the experiment. It seems like the EKf and EKF-slam is not able to handle the gyro-measurements well during periods of high acceleration and deceleration. Theoretically one would expect the EKF-SLAM do be able to handle this better than a pure EKF as estimations is based on both gyro and sonar measurements. However there are two problems in this case. Firstly as the AUV is turning it has not yet actually discovered any landmarks in the area it is turning towards, giving no possibility for correction. Secondly the IMU/DVL and sonar data was recorded on two different systems, leading to the measurements not being perfectly synchronized. Therefore a quick turn will lead to some lag in either measurement. This may actually lead to a greater error than just using the gyroscope for estimations if a landmark is wrongly associated with a scan as a result of lag, and then used to update the estimations.

After the first deviation the trajectories of the EKF and EKF-SLAM seems to follow the same pattern

as the Qualsys well. During the last part of the trajectory one can see that the EKF-SLAM algorithm is doing some corrections leading to different behaviour than the EKF and Qualsys estimations. These corrections are most likely done on landmarks not initiated well as a result of the deviation in the start of the experiments.

As can be seen in the figures 5.14 and 5.15 the EKF have lower deviation in x and y position than the EKF-SLAM. Once can clearly see the jump in error for the y-estimation in the end of the experiments when the EKF-SLAM tries to correct its position based on landmarks. Interestingly enough at the same time one can observe a reduction in error in x-position. In figure 5.16 one can see that the EKF and EKF-SLAM ends up with about the same yaw-estimation, however both are X° of the Qualsys estimation.

During the EKF-SLAM tests on this experiment there were a problem with the same landmark being discovered multiple times when turning quickly due to the lag between sensors. Therefore the amount of votes needed in order to discover a landmarks had to be increased compared to the straight line experiment.

In figure 5.17 the map created by EKF-SLAM during the experiment can be seen. Is is quiet similar to the set-up visualised in 4.6. There are some white lines in the map that should not be there, and is a result of a bug in the code. Here the pool walls can be seen as the two longest obstacles. As one can see they are not perfectly parallel, which indicates that the initialization is not perfect.

Overall the implemented EKF-SLAM system did not seem to work as well in the present of quick rotations. However, the EKF had similar problems so a more accurate gyroscope or additional sensors to help measure yaw may fix the problem. The synchronizing between the different measurements should also be taken into consideration.

6.5 20 minute square

In figure 5.19 and 5.18 the trajectories estimated by the EKF and EKF-SLAM compared to Qualsys can be seen. Here it becomes clear that the EKF will drift over time and after 20 minutes have built up a substantial error from the AUvs actual position. This is one of the main issues for AUV navigation today and here EKF-SLAM really gets to shine. Using the walls of the pools together with some few artificially placed landmarks the EKF-SLAM is able to reduce almost all the drift in the y-direction, and most of the drift in x-direction. There seems to be a constant small deviation in x-directiom between the Qualsys estimation and the Qualsys estimations. This is most likely a result of a landmark have been initialized with a small deviation from the ground truth, and then are used to update the AUVs positions in x-direction keeping the deviation about constant throughout the experiment.

This can be seen in figures 5.20 and 5.21 where the error shows the EKf is slowly drifting away from 0 meanwhile the EKF-SLAM oscillates around 0. The big deviation occurring around 560 seconds into the experiment are due to the Qualsys system loosing track, and are corrected once QUalsys are able to locate the body of the AUV again, this is also the reason there are a straight blue line across the square in the top-down figures. One an also observe that the EKF, EKF-SLAM and Qualsys has the same patterns in their x- and y-estimation but the EKF estimations are drifting towards one direction due to the bias in the IMU and DVL.

Interestingly the yaw-estimations of the EKF seen in figure 5.22 follows the EKF-SLAM and QUalsys estimations well even after it has drifted substantially in x- and y-position meaning the orientation estimation of the EKF is still quiet accurate.

The map seen in figure 5.23 shows the map created by EKF-SLAM while running this experiment. The two walls of the pool can be seen as the black lines. Notice that also here they do not appear to be perfectly parallel. Once again a cause of this could be the unsyncronization between the IMU/DVL and sonar measurements. As with the random-loop a higher threshold for landmarks were needed compared to the straight line experiment. Some of the smaller artificial landmarks placed in the vicinity were therefore ignored. See figure 4.7 for the set-up of the experiment.

Overall the EKF-SLAM performed very well in this experiment. Even though the estimated trajectory is not identical to QUalsys with some error especially in the x-position the estimation it provides a much better estimate than the EKF.

6.6 Errors sources from the experiments and implementation of EKF-SLAM

One of the greatest error-sources came from the collected datasets as the small lag between the sensors. While not noticeable during most of the experiments, it was notable during qucik turns. This could lead to one of two situations, the first situation would be that the same landmark are added to the state vector multiple times as the change of its position between two scans are great. This may not be the biggest issue, as the first landmark added will not get any associations and therefore not influence the system in any other way than requiring slightly more computational power. The second case though would be the landmark being discovered with a wrong angle, and then used to update the estimated position in the next scan. This could lead to the entire system being rotated which could be really bad.

There were also a couple of weaknesses in the implementation that should be addressed. Firstly that the ρ and θ values needed to achieve compatibility with a landmark was set to a constant deviation from the landmarks estimated position. This is a bad approach as one should be looking for possible associations for landmarks with high uncertainty in a larger area than landmarks with low uncertainty. This should therefore rather be done by having values varying on the uncertainty of ρ and θ for each individual landmark.

Another that became clear after reprocessing experiments was that it seemed like good idea to have some sort of buffer where potential new landmarks were stored until they were seen a set amount of times, and then intitlaize that landmark as the average of the observed positions. In this implementation landmarks are added at once if a discovered line is not associated with any existing landmarks. This created situations were the landmark were initialized at slightly odd angles, which the EKF-SLAM then used to update the position of the AUV in the subsequent scans leading to an error. A more robust initializing method should be implemented to handle this.

6.7 YOLOv3 underwater

As can be seen in figures 5.24 and5.25 YOLOv3 had no problem finding the images in the underwater conditions of the pool. As mentioned the dataset were purposefully created without rotating objects 180°. If this had not been done the clear reflections of the images would also most likely have been detected by the network.

Chapter 7

Conclusion

7.1 Conclusion

In this report an EKF-SLAM has been implemented and run on an AUV during experiments conducted in the Marine Cybernetics Laboratory at Tyholt, NTNU. In addition a prototype of a direct mapping from image-pixels to sonar-angles have been tested in a world simulated in Gazebo.

EKF-SLAM gave results on par with the EKF for experiments conducted over short-period of times. This is a good result as both navigation systems were able to follow the estimated trajectory from Qualsys fairly well for the straight-line and random-loop experiment. However, there were some problems with the yaw-estimation during periods of high acceleration and deceleration which created some deviation between the estimations. The most interesting result came from the 20 minutes square experiment where the real power of SLAM became apparent. During the 20 minutes the EKF build up an substantial error with regards to Qualsys at the end having an absolute positional error of 2.5 meters. The EKF-SLAM on the other hand is almost able to eliminate the drift, having a maximal absolute positional error of under 0.5 meters. This error could probably have been even smaller with a more robust iniatilization scheme of new landmarks. Even though this experiment was conducted in a small pool in a simple scenario, it shows the potential for using sonar as a feature detector in underwater SLAM system in order to reduce the drift that occurs over time with an EKF.

One issue with this implementation is the lack of a specialised loop closure algorithm. If a substantial drift builds up in the estimation before the rediscovery of an existing landmark the estimated parameters of the "rediscovered" landmark will vary to much from the existing landmarks, and hence the algorithm will add a new landmark instead of updating based on the existing landmark.

The prototype of pixel-width to sonar-angle mapping worked well in the simulator, and allowed for placing specific objects within the occupancy-grid. Due to technical difficulties with the AUV a physical test of the system were not conducted. Even though the entire system could not be tested due to lack of a calibration dataset, YOLOv3 the object detector of choice were still tested in the pool were it performed well. However, the clear conditions of the still water in the pool can not be expected to work as well in the sea.

7.1.1 Future work

A more robust initialization of landmarks into the state vector should be implemented. A landmark should have to be observed over subsequent scans before being adopted into the state vector. As experienced during the post-processing of this report the first observation of a line was often not perfectly aligned with the actual object in the physical world, leading to wrong initialization and as a consequence undesired positional updates to the AUVs position while the landmarks position slowly are corrected to its actual position.

Compatibility between existing landmarks and new measurements should be found based on the variance of ρ and θ of the landmarks when looking for compatibility and not just have to be within a set-value. This would lead to a stricter match on landmarks with little uncertainty, and a more lenient match for landmarks with much uncertainty.

A specialised loop-closure should be implemented. In visual SLAM loop-closure are performed when the algorithm recognizes patterns of features it has seen before. There are a lot more features to extract from an camera image than a sonar scan, this makes it harder to ensure that enough patterns are present to

create such a loop closure in a sonar scan. However it might still be possible to find an algorithm that are able to recognize the patterns of landmarks discovered and then ensure that this is a robust enough match to initiate loop closure.

There is also potential in combining the sonar scan with the camrea image to get loop-closure in the SLAM system. In a scenario where the camera is able to detect an object of interest, and a pixel-to-angle mapping is used to find the same object in the sonar scan, the next time the same object is spotted by the camera, one already knows which landmark in the state vector it correlates to. This information can then also be used to correlate all other measurements to the found landmarks in the scan updating them all in a bundle adjustment.

As mentioned in the master contract the use for a 2D-SLAM system in underwater robotics is limited, as the ocean space is a 3D-environment. A natural next step would therefore be to take the algorithms into a 3D-space.

Appendices

GITHUB links to code

Written code:

EKF-SLAM, C++ - https://github.com/WaldumA/sonar_slam

Sonar driver and ROS msg, C++ - https://github.com/WaldumA/720i-sonar-linux-driver

Sonar-mapping and object placement, python - https://github.com/WaldumA/Sonar_mapping

Sonar-mapping, C++ - https://github.com/WaldumA/sonar_mapping_c

Object placement, C++ - https://github.com/WaldumA/object_placement_c

External code:

EKF - https://github.com/vortexntnu/manta-auv/tree/master/localization/robot_localization

UUV-simulator - https://github.com/vortexntnu/uuv-simulator

DarknetROS - https://github.com/leggedrobotics/darknet_ros

Link to datasheets of sensors

Sensornor STIM300 - https://www.sensonor.com/media/1132/ts1524r9-datasheet-stim300.pdf

Nortek DVL 1000 - https://www.nortekgroup.com/products/dvl-1000-300m/pdf

Tritech 720i imaging sonar - https://www.seascapesubsea.com/downloads/0685-SOM-00002-05_Gemini-720i.pdf

Bibliography

[1] Marine snow image link, 2019.

[2] Nortek DVL 1000 available at:. https://www.nortekgroup.com/products/dvl-1000-300m.

[3] Marko Bjelonic. YOLO ROS: Real-time object detection for ROS. `https://github.com/leggedrobotics/darknet_ros`, 2016–2018.

[4] Edmund Brekke. Fundamentals of sensor fusion - target tracking, navigation and slam. unpublished, 2019.

[5] Bluerobotics HD camera available at:. https://bluerobotics.com/store/sensors-sonars-cameras/cameras/cam-usb-low-light-r1/.

[6] Mariia Dmitrieva, Matias Valdenegro-Toro, Keith Brown, Gary Heald, and David Lane. Object classification with convolution neural network based on the time-frequency representation of their echo. In *2017 IEEE 27th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, sep 2017.

[7] Richard O. Duda and Peter E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM*, 15(1):11–15, jan 1972.

[8] Martin A. Fischler and Robert C. Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, jun 1981.

[9] Trygve Olav Fossum, Jo Eidsvik, Ingrid Ellingsen, Morten Omholt Alver, Glaucia Moreira Fragoso, Geir Johnsen, Renato Mendes, Martin Ludvigsen, and Kanna Rajan. Information-driven robotic sampling in the coastal ocean. *Journal of Field Robotics*, 35(7):1101–1121, jul 2018.

[10] Ross Girshick. Fast r-CNN. In *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE, dec 2015.

[11] William Rowan Hamilton. II. on quaternions or on a new system of imaginaries in algebra. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 25(163):10–13, jul 1844.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2016.

[13] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Transactions of the ASME–Journal of Basic Engineering*, 82(Series D):35–45, 1960.

[14] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, may 2017.

[15] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jul 2017.

[16] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: Single shot MultiBox detector. In *Computer Vision – ECCV 2016*, pages 21–37. Springer International Publishing, 2016.

[17] Martin Ludvigsen and Asgeir J. Sørensen. Towards integrated autonomous underwater operations for ocean mapping and monitoring. *Annual Reviews in Control*, 42:145–157, 2016.

[18] Lili Ma, Yangquan Chen, and Kevin L. Moore. A family of simplified geometric distortion models for camera calibration. *ArXiv*, cs.CV/0308003, 2003.

[19] Asgeir J. Sørensen Martin Ludvigsen, Geir Johnsen, Petter A. Lågstad, and Øyvind Ødegård. Scientific operations combining ROV and AUV in the trondheim fjord. *Marine Technology Society Journal*, 48(2):59–71, mar 2014.

[20] STIM300 Intertial measurement unit datasheet available at:. https://www.sensonor.com/media/1132/ts1524r9-datasheet-stim300.pdf.

[21] T. Moore and D. Stouch. A generalized extended kalman filter implementation for the robot operating system. In *Proceedings of the 13th International Conference on Intelligent Autonomous Systems (IAS-13)*. Springer, July 2014.

[22] J. Neira and J.D. Tardos. Data association in stochastic mapping using the joint compatibility test. *IEEE Transactions on Robotics and Automation*, 17(6):890–897, 2001.

[23] Sharmin Rahman, Alberto Quattrini Li, and Ioannis Rekleitis. Sonar visual inertial SLAM of underwater structures. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, may 2018.

[24] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2016.

[25] Joseph Redmon and Ali Farhadi. YOLO9000: Better, faster, stronger. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jul 2017.

[26] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *CoRR*, abs/1804.02767, 2018.

[27] David Ribas, Pere Ridao, and José Neira. *Underwater SLAM for Structured Environments Using an Imaging Sonar*. Springer Berlin Heidelberg, 2010.

[28] Stian Skaalvik Sandoy, Takumi Matsuda, Toshihiro Maki, and Ingrid Schjolberg. Rao-blackwellized particle filter with grid-mapping for AUV SLAM using forward-looking sonar. In *2018 OCEANS - MTS/IEEE Kobe Techno-Oceans (OTO)*. IEEE, may 2018.

[29] Ingrid Schjølberg and Ingrid Bouwer Utne. Towards autonomy in ROV operations. *IFAC-PapersOnLine*, 48(2):183–188, 2015.

[30] Mark Shortis. Calibration techniques for accurate measurements by underwater camera systems. *Sensors*, 15:30810, 12 2015.

[31] Kristoffer Solberg. Manta v1: A deliberative agent software architecture for autonomous underwater vehicles. Master thesis, 2020.

[32] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, jun 2015.

[33] Gazebo Website. http://gazebosim.org/.

[34] MATE website. https://materovcompetition.org/.

[35] Robosub website. https://robonation.org/programs/robosub/.

[36] ROS Website. https://www.ros.org/.

[37] UUV Website. https://github.com/uuvsimulator.

[38] Øyvind Denvik. A realtime inertial navigation comparison between eskf, nlo and ekf for an autonomous underwater vehicle. Master thesis, 2020.