Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Engineering Cybernetics

Vegard Haraldstad

A side-scan sonar based simultaneous localization and mapping pipeline for underwater vehicles

Master's thesis in Cybernetics and Robotics
Supervisor: Damiano Varagnolo
Co-supervisor: Simon Andreas Hagen Hoff
June 2023

NTNU
Norwegian University of
Science and Technology

Vegard Haraldstad

A side-scan sonar based simultaneous localization and mapping pipeline for underwater vehicles

Master's thesis in Cybernetics and Robotics
Supervisor: Damiano Varagnolo
Co-supervisor: Simon Andreas Hagen Hoff
June 2023

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

NTNU
Norwegian University of
Science and Technology

Abstract

The vast and unexplored ocean holds tremendous potential resources, yet sustainable and environmentally friendly harvesting of these requires a deeper understanding of the ocean's ecosystem. The advancement of autonomous capabilities of robotics presents new opportunities for effectively exploring and gathering valuable information about the ocean.

This thesis aims to develop and analyze an underwater simultaneous localization and mapping (SLAM) pipeline for autonomous underwater vehicles (AUV) that utilizes side scan sonar for navigation purposes. The goal is to improve navigation accuracy that, in turn, will extend the operational capabilities of AUVs, enabling long-time underwater missions.

The proposed SLAM pipeline consists of several key steps, including: • swath processing techniques that are employed to remove artifacts in the side-scan sonar data; • a novel probabilistic algorithm to generate a 2D cartesian map of the processed swaths; • a novel landmark detector that combines intensity thresholding, geometric filtering, classification, and height estimation to identify landmarks; • a probabilistic data association algorithm, that is utilized to associate the landmarks detected as the mission unfolds and data is collected; and • a multimodal inference scheme to generate new state and landmark estimates.

The proposed pipeline is also evaluated using real-world data, demonstrating promising results. Nevertheless, the results also show that the pipeline is improvable in some key steps: • the proposed landmark detector shows to be lacking robustness in detecting landmarks at different ranges and in detecting the full extent of the landmarks; • the swath processing shows a discrepancy between the measured and calculated first backscatter, potentially reducing the correctness of the pipeline; • loop closures do not have the expected effect on the estimated trajectory of states, and the approximations of the odometry and measurement uncertainty are thought to be the reason.

By developing a SLAM pipeline for side scan sonar, this thesis advances the field of underwater SLAM and lays the foundation for further development of a robust and accurate pipeline.

Sammendrag

Det enorme og uutforskede havet inneholder betydelige potensielle ressurser. Imidlertid krever bærekraftig og miljøvennlig utnyttelse av disse en dypere forståelse av havets økosystem. Fremskritt innen roboters autonome egenskaper åpner nye muligheter for effektiv utforskning og innhenting av verdifull informasjon om havet.

Denne masteroppgaven har som mål å utvikle og analysere en SLAM (simultan lokalisering og kartlegging) arbeidsflyt for autonome undervannsfarkoster (AUV) som bruker side-skannende sonar til navigasjonsformål. Målet er å forbedre navigasjonsnøyaktigheten, som igjen vil utvide AUV-ers operative evner og muliggjøre langvarige undervannsoppdrag.

Den foreslåtte SLAM-arbeidsflyten består av flere nøkkeltrinn, inkludert: • metoder for prosessering av sonarmålinger for å fjerne artefakter i dataen; • en ny probabilistisk algoritme for å generere et 2D kartesisk kart av de behandlede sonarmålingene; • en ny tilnærming for gjenkjenning av landemerker som kombinerer intensitetsterskling, geometrisk filtrering, klassifisering og høydeestimering for å identifisere landemerker; • en probabilistisk dataassosiasjonsalgoritme som brukes til å knytte sammen landemerkene som oppdages når oppdraget utfolder seg og data samles inn; og • en multimodal inferens metode for å generere nye tilstander og landemerke estimater.

Den foreslåtte arbeidsflyten blir også evaluert ved hjelp av virkelige data og viser lovende resultater. Likevel viser resultatene også at arbeidsflyten kan forbedres på noen nøkkeltrinn: • den foreslåtte landemerkedetektoren viser seg å mangle robusthet ved deteksjon av landemerker på forskjellige avstander og deteksjon av hele landemerkets utstrekning; • prosesseringen av sonarmålinger viser et avvik mellom målt og beregnet første mulige refleksjon, noe som potensielt reduserer nøyaktigheten til arbeidsflyten; • lukkinger av sløyfer har ikke den forventede effekten på de estimerte tilstandene, og antakelsene gjort ved utregning av odometri og målingsusikkerhet antas å være årsaken.

Ved å utvikle en SLAM-arbeidsflyt for side-skannende sonar fremmer denne masteroppgaven feltet for undervanns-SLAM og legger grunnlaget for videreutvikling av en robust og nøyaktig arbeidsflyt.

Preface

This master's thesis was conducted as part of the Masters of Science program in Cybernetics and Robotics at the Norwegian University of Science and Technology (NTNU). The work has been conducted under the supervision of Damiano Varagnolo and Simon Andreas Hagen Hoff.

I would like to express my appreciation to Simon Andreas Hagen Hoff for his support and countless hours of discussion. His guidance and assistance have been of great help. I also extend my gratitude to Damiano Varagnolo for his enthusiasm and support. In addition, I would like to thank AURLab, for providing the data used and Bjørnar Reitan Hogstad for providing the groundwork for the swath processing.

Lastly, I would like to thank my girlfriend, friends, and family. I am grateful to my girlfriend for her patience and unwavering support during the course of this thesis. My friends for their support and companionship. The shared moments of joy have been a source of inspiration. Finally, my family for always supporting and encouraging me.

Contents

Chapter 1

Introduction

This thesis builds on the work in [1] and [2], where the latter is the project thesis written at NTNU by the author. Because of the close relationship with this thesis, some parts of [2] will be similar or identical to the introduction, theories, and methods presented in this thesis.

This chapter will first present the motivation of the thesis, from the problems we want to solve to the means for solving them. Next, the chapter will introduce the problem formulation and the contributions made by this thesis. Lastly, the outline of the thesis is presented.

1.1 Motivation

Throughout history, humans have relied on the vast ocean for sustenance, commerce, and exploration, recognizing its immense potential. However, despite this, it is astonishing to note that a mere five percent of the ocean has been explored and charted [3]. This remarkable statistic highlights the vastness and complexity of the ocean, emphasizing the need for increased efforts in exploring and mapping it. The ocean affects all life forms on the planet, and knowing more about it can help safeguard its ecosystem and biodiversity. As the activity in the ocean increases, the balance of the ecosystem can the threatened by, for example, ocean pollution, and the more we know about the ocean, the more efficiently we can protect it.

The world is turning to the oceans for more resources as more and more land is occupied, increasing marine activity and, hand in hand, the demand for knowledge about the ocean. A significant increase in green energy and aquaculture production in the ocean is expected in the future [4]. One example of aquaculture production is fish farming, where the newest trend is to move the fish farms from the protected fjords and into the open ocean [5]. The establishment of fish farms in fjords has been shown to disrupt ecosystems, such as the threat posed by salmon lice to the wild salmon population and the ecological impact of fish farming waste on the ocean ecosystem [6]. However, the potential effects of offshore fish farming on surrounding ecosystems in the open ocean remain largely unknown.

Moreover, the emergence of offshore wind farms, exemplified by Hywind Tampen [7], and deep-sea mining, which offers the potential to extract rare-earth metals for electronic devices and vehicles [8], represents novel approaches to resource exploitation in the ocean. Consequently, a deeper understanding of the ocean and its ecosystems is crucial in these emerging contexts.

Autonomous underwater vehicles (AUVs) are widely used for monitoring and surveying the ocean and are more cost-effective and environmentally friendly than manned vessels [9, 10]. However, even though they are used in commercial applications, there are still challenges to solve to enhance their usability and persistence in underwater operations. One of the most significant hindrances to their use is the need for an increased ability to execute long-time operations without surfacing. More specifically, one of the main challenges left to solve is accurate underwater navigation.

Accurate underwater navigation poses a challenge due to the limitations of electromagnetic wave propagation in an underwater environment. As a result of the limitations, the global navigation satellite system (GNSS) is unavailable in underwater applications [9]. GNSS provides global position measurements, and without it, accurate navigation either needs expensive inertial sensors or an external position system, such as acoustic beacons for localization, that is expensive to install and restricts the operating area. However, a promising solution and a current research topic is simultaneous localization and mapping (SLAM).

SLAM has significantly impacted mobile robotics, improving the localization and mapping of unknown environments, making a large leap forward in their autonomous capabilities [11]. SLAM combines inertial navigation with the perception of the local environment to build a global representation of the environment and increase navigation and localization accuracy. It has primarily been applied to land and air vehicles, where cameras and/or lidars are typically used for perceiving the environment [12–14]. SLAM has also been utilized in several different underwater applications [15]. However, without the true position of the AUV and true maps of the seabed, or in other terms, a lack of ground truth, it is difficult to evaluate the performance of the different underwater SLAM methods. In addition, using a camera underwater is problematic because of the turbidity in the water and the lack of light at greater depths, significantly reducing sight, making it harder to utilize the same methods under water as above water [16].

Sonar is, for many purposes, the "eyes" of underwater robotics and uses acoustics to sense the robot's surroundings. A sonar transmits acoustic waves and "senses" the echoes reflected to give a perception of the environment. A great advantage of sonars compared to cameras is that it is not affected by turbidity and light conditions, making them an excellent alternative for underwater applications. Because of this, much of the research focuses on SLAM using sonars to perform long-time operations [15].

1.2 Problem formulation

The thesis attempts to investigate and explore the problem of performing underwater SLAM utilizing a side scan sonar as a whole. In other terms, it puts together all the pieces that are needed to transform raw data from a side scan sonar into a map that can actually be used for navigation purposes. This is a large and complex problem with several interconnected moving pieces. Since, to the best of our knowledge, there is no existing off-the-shelf complete solution on which we can build on top of, the main goal has thus been to fill this gap and develop a whole SLAM pipeline that other researchers may improve upon. This "initial" pipeline has no performance goal regarding real-time performance or accuracy, except that it should be able to perform loop closures. The secondary goal has been to iterate on the different parts of the solution as needed to increase computational performance and accuracy.

1.3 Thesis contributions

This thesis's main contribution is to build an underwater SLAM pipeline for side scan sonar and test it on real-world data. Besides this, the thesis advances the swath processing algorithm developed in [1]. More precisely, that algorithm is further developed to incorporate pitch and roll corrections to increase the overall performance. A novel probabilistic map generation algorithm is presented, based on the works in [17]. Importantly, the map generation algorithm achieves a computational performance that is compatible with an online real-time SLAM pipeline. Furthermore, a novel landmark detector is presented, building on the landmark detector from [18], tested in [2]. The novel landmark detector uses intensity thresholding, geometric filtering, landmark classification, and height estimation to perform landmark detection. Lastly, the work in [19] is implemented and adapted to perform probabilistic data association and multimodal inference for completing the SLAM pipeline.

1.4 Outline

This thesis will present the developed underwater SLAM pipeline utilizing side scan sonar and will test the pipeline on real-world data. This chapter has introduced the underwater navigation problem, presented the motivation for the thesis, and, lastly, the thesis contributions. Chapter 2 introduces the sonar and the concepts used to perceive the underwater environment using acoustic waves. Chapter 3 presents the foundations of state estimation. Chapter 4 explains how the sonar measurements are processed in the pipeline. Chapter 5 concerns how processed sonar measurements can be combined to build 2D cartesian maps of the seabed, and Chapter 6 presents how landmarks can be found in the maps generated. Chapter 7 concerns the last part of the SLAM pipeline, mainly the

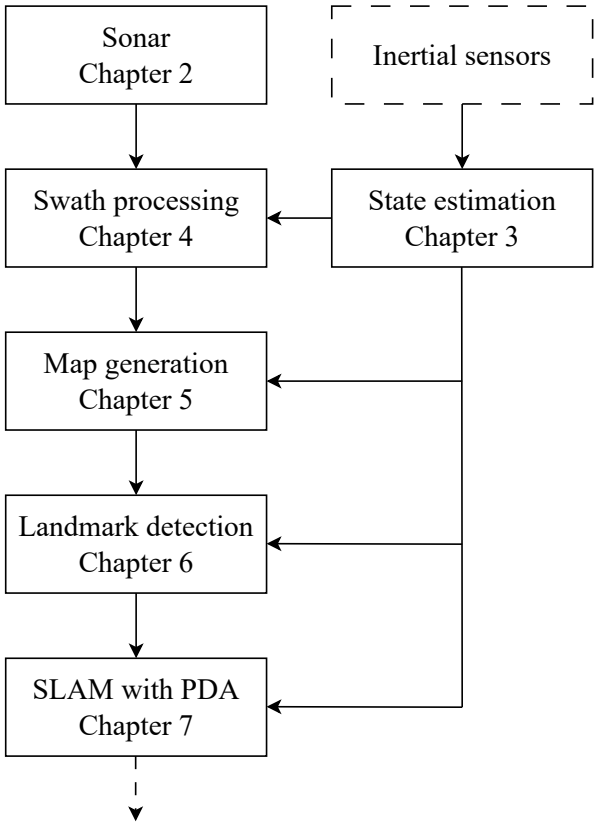


Figure 1.1: Flowchart of the different parts of the SLAM pipeline. Inertial sensors are not treated in this thesis but are needed for the state estimation. The pipeline output is new state estimates and landmark position estimates.

SLAM algorithm and an algorithm for probabilistic data association. Chapter 8 will present a review of the SLAM pipeline and discuss its strengths and weaknesses. In addition, the suggested further work will be put forward. Lastly, the thesis is concluded in Chapter 9. Figure 1.1 shows a flowchart of the SLAM pipeline and in which chapters the different parts of the pipeline are presented.

Chapter 2

Sonar

Sonar is a vital sensor for underwater robotics. It utilizes acoustics to "scan" the seabed, gathering information that can be used for further processing, such as creating a 2D map of the seabed. This chapter will treat the introductory physics of underwater acoustic, the sonar, and its measurement principle, and introduce the side scan sonar used in this thesis. Lastly, the dataset used in this thesis is presented.

2.1 Acoustic waves

Sonars utilize acoustic waves to obtain an "acoustic scan" of the seabed, and therefore acoustic waves and their properties must be examined to understand the sonar characteristics. Waves are, in their general physical form, propagating dynamical disturbances of one or more quantities and can be described by the two-way wave equation, stemming from the work of J. d'Alembert [20]

$$\frac{\partial^2 u}{\partial t^2} = k^2\left(\frac{\partial^2 u}{\partial x_1^2} + \frac{\partial^2 u}{\partial x_2^2} + \ldots + \frac{\partial^2 u}{\partial x_n^2}\right) \tag{2.1}$$

where $u = u(x_1, x_2, \ldots, x_n; t)$ describes scalar functions of time and space, and k is a real coefficient. For acoustic waves, the dynamical disturbance is a pressure wave propagating through the water column and can be described by the acoustic wave equation [21]

$$\nabla^2\phi - \frac{1}{c^2}\frac{\partial^2\phi}{\partial t^2} = Sf(t), \tag{2.2}$$

where c is the phase speed of the acoustic wave, $S = S(x, y, z)$ is the source strength of the acoustic wave, $f(t)$ is the time signature of the acoustic wave. $\phi = \phi(x, y, z; t)$ is defined as the velocity potential given by

$$\hat{u} = -\nabla\phi, \tag{2.3}$$

where $\hat{u}$ is the particle velocity vector. It is important to note that the particle velocity is not the same and is not necessarily related to the phase speed c.

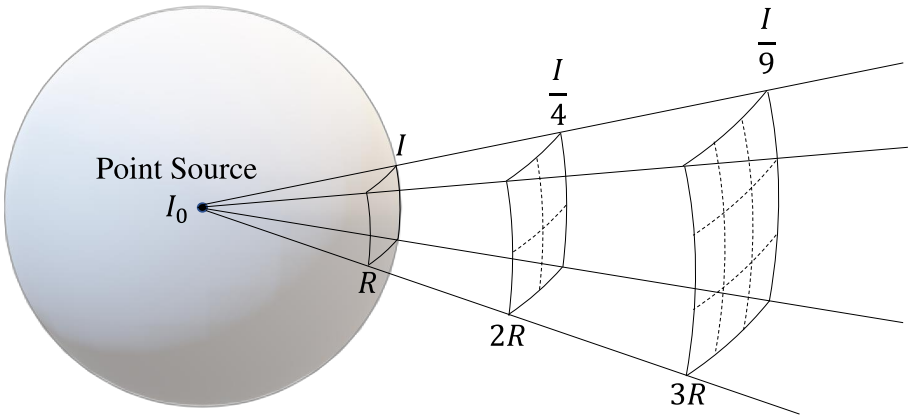


Figure 2.1: Figure showing a visualization of the intensity loss from an acoustic point source with intensity I_0, following the inverse square root law. Figure from [1], used with permission from the author.

The general description of an acoustic wave is given by (2.2), and depending on the wave type, different solutions of the wave equation can be found. In addition to the wave equation that describes the physics of the acoustic wave, we need to examine three important phenomena for underwater acoustics to understand the sonar's measuring principle and characteristics.

Firstly, as acoustic waves propagate the water collum, they experience transmission loss stemming from various sources, where the main one is acoustic intensity loss due to the geometrical spreading of the acoustic wave [21]. An acoustic wave stemming from an omnidirectional point source will spread uniformly in all directions, giving a spherical spreading and intensity loss. Assuming a reference range of 1 m from the source with a reference intensity of I_0, the intensity at range R is given by

$$I(R) = \frac{I_0}{R^2}, \tag{2.4}$$

also known as the inverse square root law. A visualization of the acoustic intensity loss is shown in Figure 2.1. In addition to acoustic intensity loss, scattering by inhomogeneities in the propagation path and frequency-dependent absorption caused by viscosity, heat conductivity, and relaxation effects attenuate the acoustic wave, thus contributing to transmission loss.

Secondly, a critical phenomenon experienced by acoustic waves is refraction and reflection at the boundary between two different mediums, also called an interface [22]. An example of such an interface can be the boundary between two water layers with different sound speeds, where the sea's sound speed will vary with parameters such as water temperature, water salinity, and whether it con-

tains heterogeneities (e.g., bubbles or suspended sediments). At such interfaces, acoustic waves will follow Snell-Descartes' laws of reflection and refraction at the interface. Examining an acoustic wave in 2D, it follows from Snells's law that the reflection is given by

$$\theta_i = \theta_r, \tag{2.5}$$

and that the refraction is given by

$$\frac{c_i}{\sin\theta_i} = \frac{c_t}{\sin\theta_t}, \tag{2.6}$$

where θ_i is the angle of incidence, c_i the speed of sound of the incoming wave, θ_r is the angle of reflection, θ_t the angle of the refracted wave and c_t the speed of sound of the refracted wave. This is shown in Figure 2.2a. Because of the phenomenon of refraction, an acoustic wave can experience bending as it traverses through the water collum [22]. However, since the distances measured by the sonar in this thesis are relatively short, an assumption of a constant sound speed is made, hence we assume no bending of the acoustic waves.

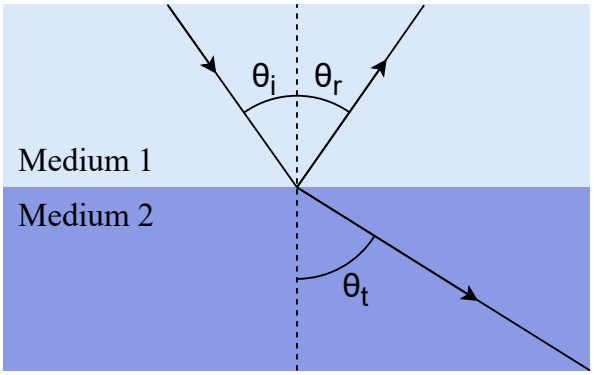
Lastly, for rough surfaces, such as the seabed, scattering of the acoustic wave occurs, in addition to refraction and reflection [21]. Because of the rough surface, the wave is not coherently reflected as for a planar interface, but rather a part of the acoustic wave is scattered in all directions. Figure 2.2b shows an example of scattering of an acoustic wave on a rough surface. The part of the acoustic wave scattered in the direction of the acoustic source is named *back scatterer*, and the part scattered away from the source is named *forward scatter*.

Further, the scattered intensity depends on several parameters, such as the surface's roughness and the incidence angle, where for example, a lower incident angle on the surface increases the backscattered intensity. The backscattering of the acoustic wave is one of the main principles sonars use to create an "acoustic scan" of the seabed.

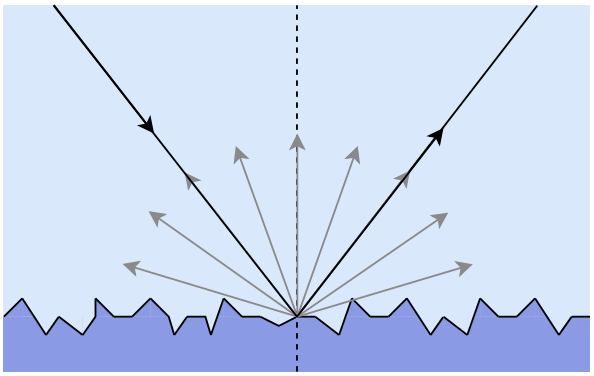
2.2 SONAR - SOund Navigation And Ranging

SONAR, which stands for "Sound Navigation and Ranging," is a sensor specifically designed for navigation and range measurement. It utilizes the backscattering of acoustic waves, or sound, on the seabed to generate an "acoustic scan" of the underwater environment. The fundamental principle of sonar involves transmitting an acoustic wave from a sonar transducer and subsequently measuring the intensity of the portion of the acoustic wave scattered by the surrounding objects. This measurement process forms the basis of the sonar's capability to gather information about the underwater surroundings.

Sonar systems where the transmitter and receiver are located in different positions are called *bistatic* and, depending on the receiver's position relative to the transmitter, can measure waves scattered in all directions [21]. *Monostatic* sonars have the transmitter and receiver in the same position and can only meas-



(a) Reflection and refraction of a 2D acoustic wave at an interface, following Snell's law. The figure is remade from [23].



(b) Scattering and reflection of an acoustic wave on a rough surface. The grey arrows show how the scattering of the acoustic wave is happening in all directions.

Figure 2.2: Refraction and reflection together with reflection and scattering of an acoustic wave.

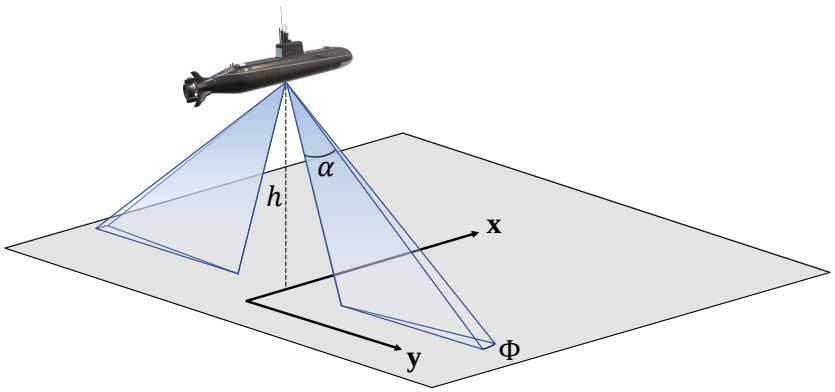


Figure 2.3: A illustrative figure showing an AUV with a side-scan sonar. Figure from [1], used with permission from the author.

ure backscattering. Monostatic sonars are the most common type and are used throughout this thesis. Hence, bistatic sonars will not be treated further.

Next, a differentiation between *continuous* and *pulsed* sonar has to be made, where the former continuously transmits an acoustic wave, and the latter transmits acoustic waves in pulses [24]. The continuous sonar is typically used for anti-submarine warfare, and since this is outside the scope of this thesis, continuous sonars are not treated further.

The sonar used in this thesis is a pulsed monostatic sonar. However, different types of pulsed monostatic sonars with different characteristics exist, such as single-beam, multi-beam, and side-scan sonar. A side-scan sonar mounted on an AUV is shown in Figure 2.3. Even though the different sonar types have different characteristics depending on their intended use, the same basic measurement principle applies to them all. The sonar characteristics are mainly governed by the transducer's design, used for transmitting the acoustic wave and measuring the backscatter, and govern both the frequency and the direction of the transmitted and measured acoustic wave.

The sonar transducers are typically made up of piezoelectric ceramics that can convert electric energy into acoustic pressure (for transmitting acoustic waves) and visa versa (for measuring the backscattering) [22]. A voltage is applied to the piezoelectric element to create acoustic waves, making the element stretch or contract, depending on the polarity of the voltage. The piezoelectric element's stretching and contraction generate pressure waves transmitted into the water surrounding the element to form an acoustic signal [21]. Figure 2.4 shows the principle of contraction and stretching of a piezoelectrical element.

The nominal frequency, the directivity, and the desired source level of a transducer are determined by the geometrical shape of the piezoelectric element and are essential design parameters [21]. The thickness of the transducer determines the nominal frequency. Transducers are typically driven at their resonance frequency, where half of the resonance frequency wavelength equals the thickness

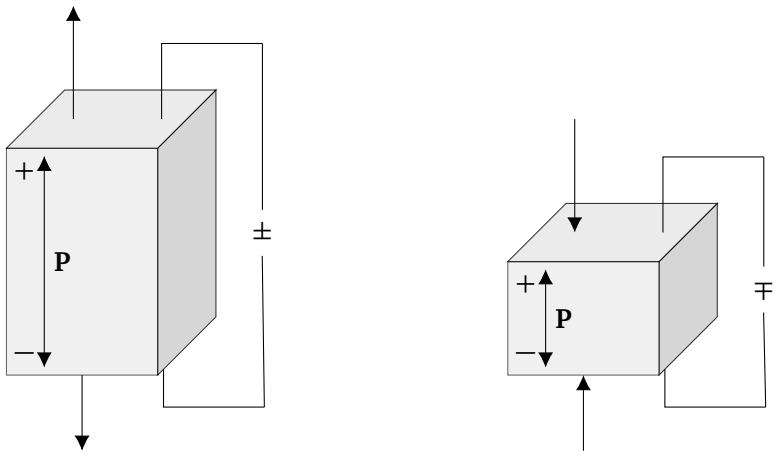


Figure 2.4: The figure shows how a piezoelectric ceramic changes shape or voltage when a voltage or a pressure is applied. Figure from [1], used with permission from the author.

of the transducer. Furthermore, the design of the transducer also governs the directivity pattern of the acoustic wave and is an important design parameter adapted depending on the intended use of the sonar [22].

The directivity pattern of a transducer typically exhibits a main lobe, accompanied by sidelobes on either side, which may not be symmetrically distributed around the main lobe. An illustration of the directivity pattern for a side-scan sonar is presented in Figure 2.5, demonstrating the presence of both the main lobe and sidelobes for the two transducers. However, these sidelobes are generally undesired characteristics in sonar systems as they can introduce anomalies in the recorded sonar data. Consequently, a common parameterization of the sonar transducer is the beamwidth of the main lobe, which is quantified by the point at which the transmitted signal drops by 3 dB from its peak amplitude.

The waveform of the acoustic pulse in sonar systems is an integral part of a system as it represents the "voice" of the system and will, to a great extent, determine the amount and quality of the information in the measured scan [25]. Two important waveforms are the constant waveform (CW) and the linear frequency modulated (LFM) waveform called chirp. The CW consists of a sinusoidal with a constant frequency, whereas LFM uses a sinusoidal with a frequency that varies linearly with time. Examples of both waveforms are shown in Figure 2.6. The advantage of using LFM is that pulse compression can be performed, increasing the range resolution of the sonar. Because of this, the LFM of chirp is common in modern sonar systems.

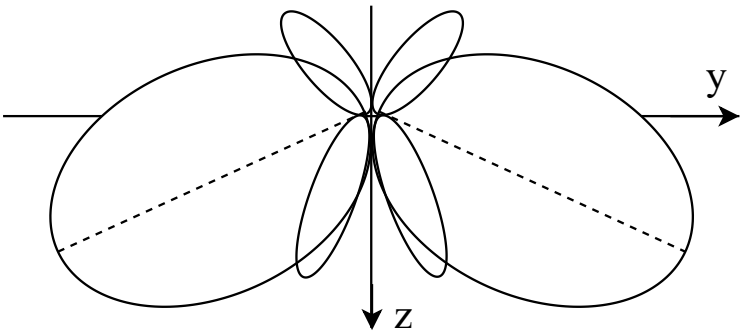


Figure 2.5: An exemplified directivity pattern of side-scan sonar. Each transducer has a main lobe and two unsymmetrical side lobes.

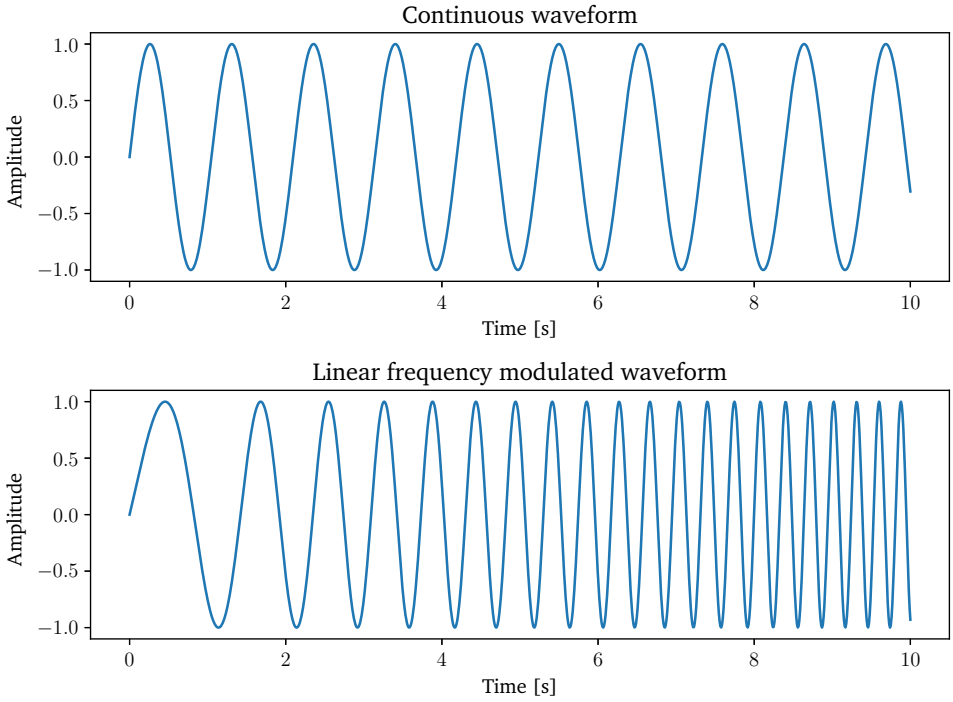


Figure 2.6: Example of the continuous waveform and the linear frequency modulated waveform.

2.3 Side scan sonar

The side scan sonar consists of transducers with narrow horizontal and wide vertical directivity, pointing perpendicular to the traveling direction [21]. A single-row side scan sonar consists of two transducers pointing to the port and starboard, as shown on an AUV in Figure 2.3. The transducer is tilted from the horizontal axis towards the seabed, creating a swath-like pattern on the seabed because of the narrow horizontal and wide vertical directivity. Because of this, a side scan sonar scan is often referred to as a *swath*. Side scan sonars measure the backscattered intensity at regular range intervals, where each intensity measurement is referred ta as a *bin*.

Multi-row side scan sonar utilizes two or more transducers at each side to determine the incidence angle of the backscattered acoustic wave and hence receive *bathymetric* data about the seabed. The multi-row side scan sonar is not used in this thesis and will not be examined further.

Since it's not possible to decide the incidence angle of the backscatter on a single-row side scan sonar, a *flat seafloor assumption* can be made to approximate the incidence angle of the echo returns [17, 26–28]. With the flat seafloor assumption, the geometry of the measurement and important quantities, such as the ground range and the slant range of the first bottom return, can be found.

Figure 2.7 shows the geometry of the starboard transducer of a side scan sonar measurement seen in the direction of travel. Here, β is the angle between the transducer's y-axis and the acoustic axis, α_v is the vertical beamwidth of the transducer, and h_t is the altitude of the transducer. For a point on the seafloor with height h_p, a slant range from the transducer, r_s, and a slant angle, β_s, we have that the ground range, r_g, is denoted

$$r_g(r_s, h_t, h_p) = \sqrt{r_s^2 - (h_t - h_p)^2}. \tag{2.7}$$

With the flat seafloor assumption, we assume that all measured points have a height of zero, reducing (2.7) to $r_g(r_s, h_t) = \sqrt{r_s^2 - h_t^2}$.

Another quantity of interest is the slant range of the first bottom return, r_{fbr}, given by

$$r_{fbr} = \frac{h_t}{\sin(\beta + \frac{\alpha_v}{2})}. \tag{2.8}$$

This thesis defines the first bottom return as the first backscatter received inside the transducer beamwidth opening, which does not necessarily coincide with the measured first bottom return.

2.4 Sonar and navigation data in this thesis

In this thesis, the sonar and navigation data used was acquired in the Trondheim fjord by *AURLab* using their vehicle *LAUV Fridtjof* [29]. Figure 2.8 shows where in the Trondheim fjord the data was collected, where the origin of the navigation

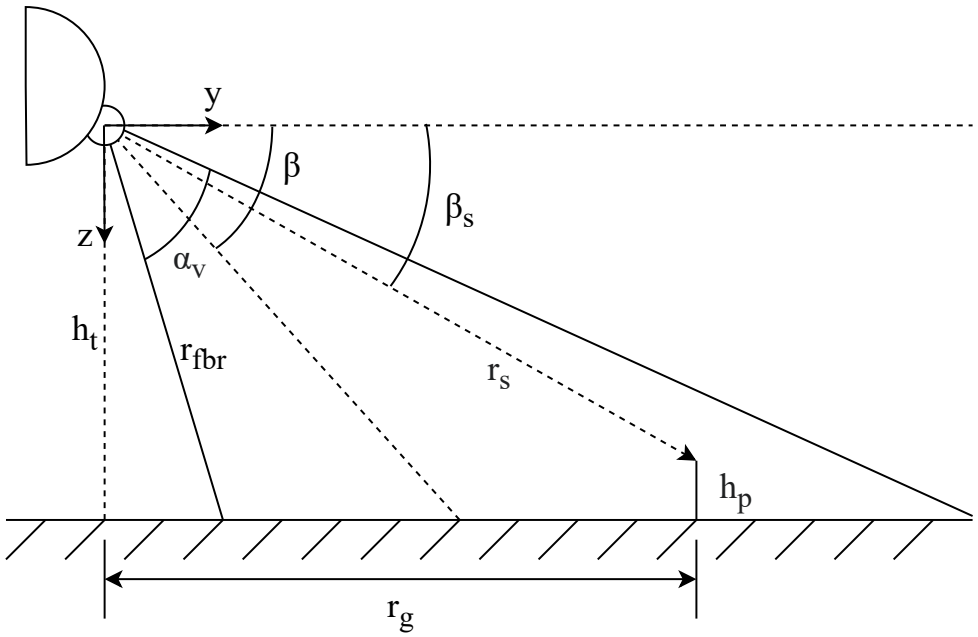


Figure 2.7: The geometry of the starboard transducer of a side scan sonar. β is the angle between the transducer's y-axis and the acoustic axis, α_v is the vertical beamwidth of the transducer, and h_t is the altitude of the transducer. An object with height h_p, slant range r_s, and slant angle β_s is also displayed.

data is at N63°24′16.685″ E10°24′5.8962″. The LAUV is equipped with a side scan sonar, consisting of two DeepVision DualChirp transducers [30] and a DeepVision OSM2 sonar module [29]. The parameter values used when collecting the data are shown in Table 2.1.

Only parts of the dataset are used in this thesis. The parts used are further divided into a training and a test dataset of 4890 and 3000 swaths, respectively. The training dataset is used for testing and tuning the different methods, whereas the test dataset is only used to test the final parameters on unknown ground.

Table 2.1: Parameters used by the LAUV when the data used in this thesis was collected.

Parameter	Symbol	Value
Frequency		640 kHz
Horizontal mounting angle	β	25°
Horizontal beamwidth	α_h	0.5°
Vertical beamwidth	α_v	60°
Sonar range	$r_{s_{max}}$	30 m
Slant range resolution	δ_s	0.03 m
Sound speed	c	1500 m s^{-1}

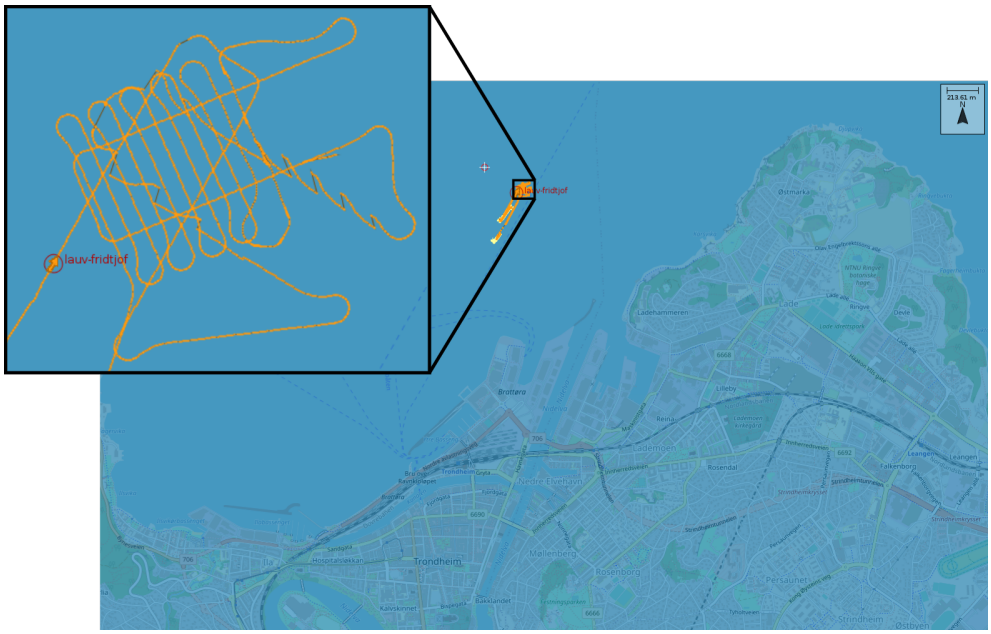


Figure 2.8: Map showing where in the Trondheim fjord the data used in this thesis was collected. The zoomed-in path shows the path the AUV took during the data collection. The origin of the data is at N63°24′16.685″ E10°24′5.8962″.

Chapter 3

State estimation

This chapter will present the state estimation part of the SLAM pipeline in Figure 1.1 and relevant background on three-dimensional geometry. State estimation is an integral part of robotics and is used for estimating the location and orientation of the robots. General knowledge of three-dimensional geometry is needed to perform state estimation for AUVs. Therefore, the theoretical foundations of poses and rotations, encompassing various parameterizations of rotations, as well as key properties associated with rotations and poses, are presented. Furthermore, three state estimation filters will be introduced to address the estimation of these states.

State estimation filters come in many flavors, but the presented filters here will be the Bayesian filter, the Kalman filter for linear systems, and the extended Kalman filter for non-linear systems. There are also many other versions of the Kalman filter, such as the error state Kalman filter, which is more suitable for inertial navigation than the extended Kalman filter [31]. However, the data used in this thesis was gathered with a LAUV [29] that implements an extended Kalman filter for state estimation [32]. Because of this and since state estimation using filters is not the main topic of this thesis, only the Bayesian filter, the Kalman filter, and the extended Kalman filter are presented. Lastly, the state interpolation implemented by the author in this thesis is presented.

3.1 Three-dimensional geometry

To perform state estimation in robotics, we need knowledge about three-dimensional geometry, especially rotations, and the different ways of representing them. This thesis presents a condensed summary with only selected themes in three-dimensional geometry, and the reader is referred to [33] for a more thorough explanation.

Robots that can move freely, both in terms of translation and rotation, have, mathematically speaking, six degrees of freedom. The combination of translation and rotation is also known as the *pose* of the robot [33], and we can use the concept of *reference frames* to represent them.

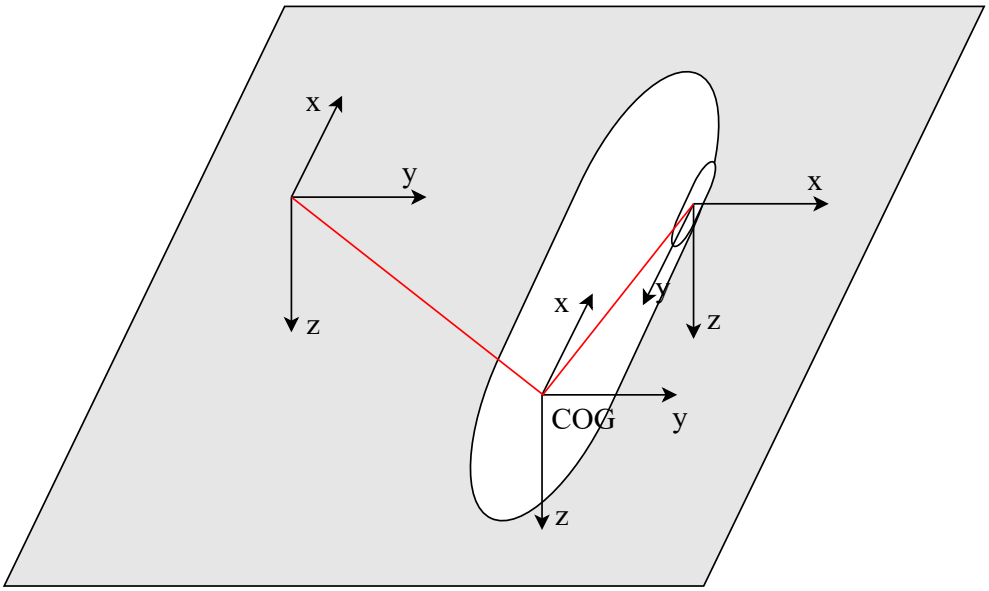


Figure 3.1: An AUV with a NED world frame, a body frame centered in the center of gravity (COG), and the starboard transducer frame shown. Note that the COG is placed strangely far back on the AUV for a clear presentation. The red lines show the relationship between the different frames.

A reference frame describes a reference coordinate system, and we use *coordinate frame transformations* to describe the relative transformation and orientation between two reference frames. For example, we typically define an inertial frame, sometimes referred to as a world frame, that describes the reference frame of the "world" for the robot. In addition, a body frame that presents the robot's reference frame and potentially several sensor frames are also defined. Using a coordinate transformation, we can describe the relationship between the different reference frames and transform poses and vectors represented relative to one frame to be represented relative to another reference frame.

This thesis uses the north-east-down (NED) coordinate system, a geographic coordinate system defined relative to the earth's reference ellipsoid [34]. In the world frame, the x-axis points north, the y-axis east, and the z-axis straight down. The body frame is typically centered in the center of gravity with the x-axis pointing forward on the AUV, the y-axis to starboard, and the z-axis straight down. This is shown in Figure 3.1 together with a sensor frame. It is important to note that a NED-coordinate system uses a tangent plane to the earth and will only be valid close to its origin.

Rotations have several different parameterizations, but this thesis will cover rotation matrices, Euler angles, and quaternions. All parametrizations seek to describe the three degrees of freedom nature of rotations but do so with different advantages and disadvantages. Rotation matrices are the most general representation and are defined as the dot product between the basis vectors of two frames

[33]. Given two frames with origin in the same point and their basis vectors, $\boldsymbol{\mathcal{F}_1} = [b_{1,1}, b_{1,2}, b_{1,3}]$ and $\boldsymbol{\mathcal{F}_2} = [b_{2,1}, b_{2,2}, b_{2,3}]$, the rotation matrix describing the rotation from frame $\boldsymbol{\mathcal{F}_1}$ to $\boldsymbol{\mathcal{F}_2}$ is

$$\boldsymbol{R_1^2} = \boldsymbol{\mathcal{F}_1} \cdot \boldsymbol{\mathcal{F}_2^T} \tag{3.1}$$

We can use this relation to describe the rotation of vectors by $\boldsymbol{v_2} = \boldsymbol{R_1^2 v_1}$. It is important to note that the notation used in this thesis is $\boldsymbol{v_{to}} = \boldsymbol{R_{from}^{to}} \boldsymbol{v_{from}}$.

Further, some restrictions are imposed for a rotation matrix to be valid. First, its determinant must equal one. Secondly, $\boldsymbol{RR^T} = \mathbf{1}$. This again leads to the relation $\boldsymbol{R_2^1} = \boldsymbol{(R_1^2)^T}$, which makes finding the inverse rotation very simple. The set of all valid three-dimensional rotation matrices belongs to the *special orthogonal group* $SO(3)$, formally given as

$$SO(3) = \{\boldsymbol{R} \in \mathbb{R}^{3x3} \mid \boldsymbol{RR^T} = \mathbf{1}, \det \boldsymbol{R} = 1\}. \tag{3.2}$$

Principal rotations are rotations around one axis and are given by only one angle [33]. Let the rotation angle around the x-axis be ϕ, the y-axis θ, and the z-axis be ψ. The principal rotations around the axis in the NED-coordinate system are [34]

$$\boldsymbol{R_x}(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & c\phi & -s\phi \\ 0 & s\phi & c\phi \end{bmatrix} \tag{3.3}$$

$$\boldsymbol{R_y}(\theta) = \begin{bmatrix} c\theta & 0 & s\theta \\ 0 & 1 & 0 \\ -s\theta & 0 & c\theta \end{bmatrix} \tag{3.4}$$

$$\boldsymbol{R_z}(\psi) = \begin{bmatrix} c\psi & -s\psi & 0 \\ s\psi & c\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}, \tag{3.5}$$

where $s(\cdot)$ and $c(\cdot)$ is abbreviations for $\sin(\cdot)$ and $\cos(\cdot)$.

Euler angles combine three principal rotations to form a three degree of freedom rotation [33]. The first rotation is a rotation around the original axis. The two next rotations, however, will not be around the original axes but rather around intermediate axes. Depending on the order of the rotations, the different axes are rotated in different orders. This thesis uses a Z-X-Y sequence, giving the following rotation matrix [34]

$$\boldsymbol{R_{z,y,x}}(\psi, \theta, \phi) = \boldsymbol{R_z}(\psi)\boldsymbol{R_y}(\theta)\boldsymbol{R_x}(\phi) = \begin{bmatrix} c\psi c\theta & -s\psi c\phi + c\psi s\theta s\phi & s\psi s\phi + c\psi c\phi s\theta \\ s\psi c\theta & c\psi c\phi + s\phi s\theta s\psi & -c\psi s\phi + s\theta s\psi c\phi \\ -s\theta & c\theta s\phi & c\theta c\phi \end{bmatrix}. \tag{3.6}$$

When ϕ, θ, and ψ are used to denote the roll, pitch, and yaw of a robot in the NED-coordinate system, the rotation from world to the body frame is $\boldsymbol{R_w^b} =$

$\boldsymbol{R}_{z,y,x}(\psi,\theta,\phi)$. An important drawback of using Euler angles is that all combinations of principal rotations have singularities, meaning that the orientation at the singularity is undefined. The sequence used in this thesis has a singularity at $\theta = \pm\frac{\pi}{2}$. This is chosen because AUVs typically won't have an orientation close to the singularity.

Another parametrization of rotations, using four parameters and without singularities, is the quaternion representation [33]. The quaternion can be found the following way. First, let us denote the axis of rotation by $\boldsymbol{a} = [a_1, a_2, a_3]^T$ and restrict $\boldsymbol{a}$ to be a unit vector. Next, let the angle of rotation be ϕ. The rotation matrix for the axis and angle of rotation is given by

$$\boldsymbol{R}(\phi,\boldsymbol{a}) = \cos\phi\,\mathbf{1} + (1-\cos\phi)\boldsymbol{a}\boldsymbol{a}^T - \sin\phi\,\boldsymbol{a}^{\boldsymbol{x}}, \tag{3.7}$$

where $\boldsymbol{a}^{\boldsymbol{x}}$ is the skew matrix of $\boldsymbol{a}$. The quaternion is then given by

$$\boldsymbol{q} = \begin{bmatrix}\eta\\ \boldsymbol{\epsilon}\end{bmatrix} = \begin{bmatrix}\eta\\ \epsilon_1\\ \epsilon_2\\ \epsilon_3\end{bmatrix}, \tag{3.8}$$

where

$$\begin{aligned}\eta &= \cos\frac{\phi}{2}\\ \boldsymbol{\epsilon} &= \boldsymbol{a}\sin\frac{\phi}{2} = \begin{bmatrix}a_1\sin\frac{\phi}{2}\\ a_2\sin\frac{\phi}{2}\\ a_3\sin\frac{\phi}{2}\end{bmatrix} = \begin{bmatrix}\epsilon_1\\ \epsilon_2\\ \epsilon_3\end{bmatrix}.\end{aligned} \tag{3.9}$$

The quaternion must be unit-length, $\boldsymbol{q}^T\boldsymbol{q} = 1$, to be valid.

As shown earlier, the special orthogonal group $SO(3)$ contains all valid rotation matrices. Poses, i.e., rotations and translations, are elements of the special Euclidean group $SE(3)$ [33] and are given by

$$SE(3) = \left\{\boldsymbol{T} = \begin{bmatrix}\boldsymbol{R} & \boldsymbol{t}\\ \mathbf{0}^T & 1\end{bmatrix} \in \mathbb{R}^{4x4} \mid \boldsymbol{R} \in SO(3), \boldsymbol{t} \in \mathbb{R}^3\right\} \tag{3.10}$$

where $\boldsymbol{T}$ is the transformation matrix, $\boldsymbol{R}$ the rotation matrix and $\boldsymbol{t}$ is the translation vector. The transformation matrix has nice properties in terms of presenting transformation between frames. If we use *augemented coordinate vectors* $\boldsymbol{v} = [v_1, v_2, v_3, 1]^T$, the transformation between two frames can be calculated by

$$\boldsymbol{v_2} = \boldsymbol{T_1^2}\boldsymbol{v_1}, \tag{3.11}$$

effectively performing the calculation $\boldsymbol{v_2} = \boldsymbol{R_1^2}\boldsymbol{v_1} + \boldsymbol{t_1}$, where the translation vector, $\boldsymbol{t_1}$, is given in the coordinates of the reference frame of $\boldsymbol{v_1}$ and $\boldsymbol{t_1}$, $\boldsymbol{v_1}$ and $\boldsymbol{v_2}$ is ordinary coordinate vectors.

$SO(3)$ and $SE(3)$ are not vector spaces but can be shown to be *matrix Lie groups* [33]. A matrix Lie group is a *differential manifold*, and operations on a differential manifold differ from those in vector spaces. However, associated with each Lie matrix group is a *Lie algebra*, consisting of a vector space where we can use ordinary operations from other vector spaces.

Matrix Lie groups are a great tool for robotics, especially in the context of SLAM, where it is used for inference. However, since the inner workings of the inference part of SLAM are outside the scope of this thesis, matrix Lie groups are not treated any further, and the reader may refer to [35] for a short introduction and to [33] for a more thorough presentation.

As the maps used in this thesis are 2D, we are also interested in two-dimensional poses. They make out the special Euclidian group $SE(2)$, which is defined by

$$SE(3) = \left\{ \boldsymbol{T} = \begin{bmatrix} \boldsymbol{R} & \boldsymbol{t} \\ \boldsymbol{0}^T & 1 \end{bmatrix} \in \mathbb{R}^{3x3} \mid \boldsymbol{R} \in SO(2), \boldsymbol{t} \in \mathbb{R}^2 \right\} \tag{3.12}$$

where $\boldsymbol{R}$ is the rotation matrix and $\boldsymbol{t}$ the translation vector [35]. If we define the rotation by the yaw angle ψ, the rotation matrix is given by

$$\boldsymbol{R} = \begin{bmatrix} c\psi & -s\psi \\ s\psi & c\psi \end{bmatrix}, \tag{3.13}$$

where $s(\cdot)$ and $c(\cdot)$ are abbreviations for $\sin(\cdot)$ and $\cos(\cdot)$.

3.2 Bayes filter

In the context of state estimation, the goal is to estimate the state of a stochastic dynamic system from a series of noisy measurements, also described as the filtering problem [31]. Furthermore, the state of a system is defined as all information needed to describe a system fully. A state vector, $\boldsymbol{x}$, containing all the states, is often used to denote the full state of a system. For robots, the state vector typically consists of its pose and velocities.

The filtering problem is formulated in terms of two models [31]. The first concerns how the state evolves in time and is called the kinematic prior. Let $\boldsymbol{x_{k-1}}$ be the state of the system at $t = k-1$. The state transition from $t = k-1$ to $t = k$ is the kinematic prior and is specified as $p(\boldsymbol{x_k}|\boldsymbol{x_{k-1}})$. The second model is known as the measurement model and concerns how the measurements observed are related to the state vector and is specified by $p(\boldsymbol{z_k}|\boldsymbol{x_k})$.

The solution to the filtering problem in the Bayesian approach is considered to be the posterior distribution $p(\boldsymbol{x_k}|\boldsymbol{z_{1:k}})$ [31]. The cyclic filtering consists of a prediction and an update step that is solved iterable to find the solution. The

prediction step is given by

$$\begin{aligned} p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k-1}) &= \int p(\boldsymbol{x}_k, \boldsymbol{x}_{k-1}|\boldsymbol{z}_{1:k-1}) d\boldsymbol{x}_{k-1} \\ &= \int p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) p(\boldsymbol{x}_{k-1}|\boldsymbol{z}_{1:k-1}) d\boldsymbol{x}_{k-1}, \end{aligned} \tag{3.14}$$

and the update step follows from Bayes's rule and is given by

$$p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k}) = \frac{p(\boldsymbol{z}_k|\boldsymbol{x}_k) p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k-1})}{p(\boldsymbol{z}_k|\boldsymbol{z}_{1:k-1})} \propto p(\boldsymbol{z}_k|\boldsymbol{x}_k) p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k-1}). \tag{3.15}$$

The prediction step and update step are known as the Bayes filter and are the foundation for the Kalman filter and extended Kalman filter.

3.3 Kalman filter

It is not expected to always find closed-form solutions to the Bayes filter. However, when the system at hand is linear and has a Gaussian noise model, a closed-form solution can be found in terms of the Kalman filter [31]. The kinematic prior, the measurement model, and the prior density in a Kalman filter are given by

$$\begin{aligned} p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) &= \mathcal{N}(\boldsymbol{F}\boldsymbol{x}_{k-1}, \boldsymbol{Q}) \\ p(\boldsymbol{z}_k|\boldsymbol{x}_k) &= \mathcal{N}(\boldsymbol{H}\boldsymbol{x}_k, \boldsymbol{R}) \\ p(\boldsymbol{x}_0) &= \mathcal{N}(\hat{\boldsymbol{x}}_0, \boldsymbol{P}_0), \end{aligned} \tag{3.16}$$

where $\boldsymbol{F}$ is the state transition matrix and $\boldsymbol{H}$ is the measurement matrix. $\boldsymbol{Q}$ is the kinematic prior noise, and $\boldsymbol{R}$ is the measurement noise, where both are assumed to be symmetric positive definite matrices. $\hat{\boldsymbol{x}}_0$ and $\boldsymbol{P}_0$ are the initial state and initial covariance of the system, respectively.

Only the closed-form solution of the Kalman filter will be stated here, and the reader is referred to [31] for the proof. The prediction step of the Kalman filter is

$$\begin{aligned} p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k-1}) &= \mathcal{N}(\hat{\boldsymbol{x}}_{k|k-1}, \boldsymbol{P}_{k|k-1}) \\ &= \mathcal{N}(\boldsymbol{F}\boldsymbol{x}_{k-1}, \boldsymbol{F}\boldsymbol{P}_{k-1}\boldsymbol{F}^T + \boldsymbol{Q}), \end{aligned} \tag{3.17}$$

and the update step is

$$\begin{aligned} p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k}) &\propto \mathcal{N}(\hat{\boldsymbol{x}}_k, \boldsymbol{P}_k) \\ &= \mathcal{N}(\hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{W}_k(\boldsymbol{z}_k - \boldsymbol{H}\hat{\boldsymbol{x}}_{k|k-1}), (\boldsymbol{I} - \boldsymbol{W}_k\boldsymbol{H})\boldsymbol{P}_{k|k-1}), \end{aligned} \tag{3.18}$$

where $\boldsymbol{W}_k$ is the Kalman gain at $t = k$, given by

$$\boldsymbol{W}_k = \boldsymbol{P}_{k|k-1}\boldsymbol{H}^T(\boldsymbol{H}\boldsymbol{P}_{k|k-1}\boldsymbol{H}^T + \boldsymbol{R})^{-1}. \tag{3.19}$$

The Kalman filter is an optimal solution to the filtering problem but only applies to linear systems with Gaussian noise models. Other types of Kalman filters, such as the extended Kalman filter, are needed for non-linear systems.

3.4 Extended Kalman filter

The extended Kalman filter (EKF) builds on top of the Kalman filter and can be used for state estimation of a non-linear system with Gaussian noise models [31]. The central concept behind the EKF is to linearize the system around the most recent estimate for the prediction step and the most recent prediction for the update step to be able to use the Kalman filter equations.

Typically, the kinematics prior, measurement model, and the prior density for the EKF are given by

$$\begin{aligned} p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}) &= \mathcal{N}(\boldsymbol{f}(\boldsymbol{x}_{k-1}), \boldsymbol{Q}) \\ p(\boldsymbol{z}_k|\boldsymbol{x}_k) &= \mathcal{N}(\boldsymbol{h}(\boldsymbol{x}_k), \boldsymbol{R}) \\ p(\boldsymbol{x}_0) &= \mathcal{N}(\hat{\boldsymbol{x}}_0, \boldsymbol{P}_0), \end{aligned} \tag{3.20}$$

where $\boldsymbol{f}$ is the non-linear state transition function and $\boldsymbol{h}$ is the non-linear measurement function [31].

To linearize $\boldsymbol{f}$ and $\boldsymbol{h}$, error variables are first introduced as

$$\begin{aligned} \Delta\boldsymbol{x}_{k-1} &= \boldsymbol{x}_{k-1} - \hat{\boldsymbol{x}}_{k-1} \\ \Delta\boldsymbol{x}_k &= \boldsymbol{x}_k - \hat{\boldsymbol{x}}_{k|k-1}. \end{aligned} \tag{3.21}$$

Then, Taylor expansions in terms of the error variables are performed

$$\begin{aligned} \boldsymbol{f}(\boldsymbol{x}_{k-1}) &\approx \boldsymbol{f}(\hat{\boldsymbol{x}}_{k-1}) + \boldsymbol{F}(\hat{\boldsymbol{x}}_{k-1})\Delta\boldsymbol{x}_{k-1} \\ \boldsymbol{h}(\boldsymbol{x}_k) &\approx \boldsymbol{h}(\hat{\boldsymbol{x}}_{k|k-1}) + \boldsymbol{H}(\hat{\boldsymbol{x}}_{k|k-1})\Delta\boldsymbol{x}_k. \end{aligned} \tag{3.22}$$

$\boldsymbol{F}$ and $\boldsymbol{H}$ are the Jacobians of $\boldsymbol{f}$ and $\boldsymbol{h}$ with respect to $\boldsymbol{x}_{k-1}$ and $\boldsymbol{x}_k$ and are given by

$$\begin{aligned} \boldsymbol{F}(\hat{\boldsymbol{x}}_{k-1}) &= \frac{\partial}{\partial \boldsymbol{x}_{k-1}} \boldsymbol{f}(\boldsymbol{x}_{k-1}) \bigg|_{\boldsymbol{x}_{k-1}=\hat{\boldsymbol{x}}_{k-1}} \\ \boldsymbol{H}(\hat{\boldsymbol{x}}_{k|k-1}) &= \frac{\partial}{\partial \boldsymbol{x}_k} \boldsymbol{h}(\boldsymbol{x}_k) \bigg|_{\boldsymbol{x}_k=\hat{\boldsymbol{x}}_{k|k-1}}. \end{aligned} \tag{3.23}$$

Using the same equations as for the Kalman filter, the prediction step is then given as

$$p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k-1}) = \mathcal{N}(\boldsymbol{f}(\boldsymbol{x}_{k-1}), \boldsymbol{F}\boldsymbol{P}_{k-1}\boldsymbol{F}^T + \boldsymbol{Q}), \tag{3.24}$$

where $\boldsymbol{F}$ now is the jacobian of $\boldsymbol{f}$. Furthermore, the update step is

$$p(\boldsymbol{x}_k|\boldsymbol{z}_{1:k}) \propto \mathcal{N}(\hat{\boldsymbol{x}}_{k|k-1} + \boldsymbol{W}_k(\boldsymbol{z}_k - \boldsymbol{h}(\hat{\boldsymbol{x}}_{k|k-1})), (\boldsymbol{I} - \boldsymbol{W}_k\boldsymbol{H})\boldsymbol{P}_{k|k-1}), \tag{3.25}$$

where $\boldsymbol{H}$ is the jacobian of $\boldsymbol{h}$. $\boldsymbol{W}_k$ is the Kalman gain in (3.19) calculated with $\boldsymbol{F}$ and $\boldsymbol{H}$ as found in (3.23).

As mentioned in Section 2.4, the data used in this thesis is collected with a LAUV [29], where the LAUV implements an EKF as its state estimation filter [32].

The EKF implemented is a standard EKF and uses measurements from global positioning system (GPS), long baseline (LBL) acoustic positioning, inertial measurement unit (IMU), attitude, heading and reference system (AHRS), and Doppler velocity logger (DVL) to estimate the state of the LAUV.

The GPS is only available when the AUV is surfaced, and the LBL acoustic positioning needs two baseline transponders fixed at known locations, where both are able to provide global position measurements. However, staying on the surface or relying on external transponders is not always an option, and in the case of no global position measurements, the filter will drift over time. This is the main reason for performing SLAM: correcting drift by reobserving landmarks and achieving accurate navigation over longer timespans.

3.5 State interpolation

The state estimation filters presented above are discrete-time state estimation filters, meaning they estimate the state at discrete timesteps and are run at a constant rate. But what if the state at the time between two timesteps is needed? This could be the case for sensor measurements, such as sonar measurements, that aren't synchronized with the discrete-time state estimation and where the pose at sonar measurement time is needed for further processing. Different solutions exist, such as continuous-time estimation or interpolating discrete-time state estimates.

In [33], continuous-time estimation utilizing *Gaussian processes* are outlined, providing means to query the filter for the state at any time. The filtering problem is formulated as a Gaussian process and then solved at the time of measurements as an optimization problem. With the continuous-time formulation, Gaussian process interpolation can find the state at any given time and is not restricted to discrete timesteps.

Another solution to unsynchronised measurements is to use a polynomial interpolation scheme to interpolate the state estimates obtained from the discrete-time state estimation. One solution would have been to use cubic spline interpolation to interpolate the pose [36]. Since the cubic spline is continuous in its second-order derivative, this would yield a continuous and smooth acceleration. However, for simplicity, a linear interpolation scheme was implemented for this thesis.

The linear interpolation scheme used in this thesis to find the state x at time t is given by

$$x = x_{n-1} + \frac{t - t_{n-1}}{t_n - t_{n-1}} x_n, \tag{3.26}$$

where x_{n-1} is the state at time t_{n-1}, x_n is the state at t_n, and t is restricted to $t_{n-1} \leq t \leq t_n$. This simple linear interpolation scheme gives a piecewise constant velocity but a non-defined acceleration. An example of linear interpolation in the xy-plane is shown in Figure 3.2, where the figure shows interpolation of state estimates with long timesteps. The state estimation during the data collection was

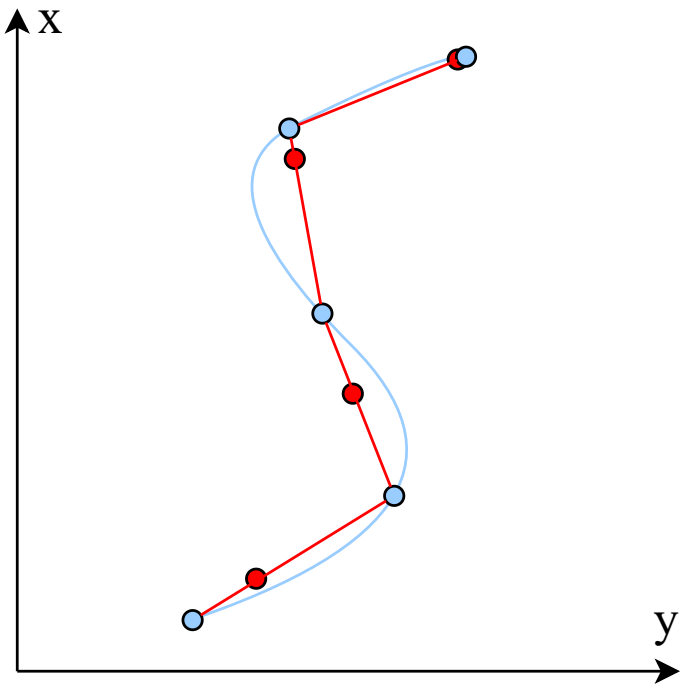


Figure 3.2: The figure shows how linear interpolation is performed on a trajectory in the xy-plane. The blue line is the true trajectory, and the blue dots are the state estimates. The red trajectory shows how linear interpolation leads to a non-smooth trajectory, where the red dots are the interpolated positions.

performed at 20 Hz, and the AUV moved slowly during the course of the collection. Hence, the errors from the linear interpolation will be much smaller than those shown in the figure.

The non-smooth trajectory is not optimal and should be considered to be revised in further work, for example, implementing a cubic spline interpolation method. The linear interpolation scheme is used to interpolate the position of the AUV and the altitude between available state estimates. However, due to rotations not being a vector field, as mentioned in Section 3.1, interpolation of the rotation is handled differently.

To interpolate the rotation of the AUV, *spherical linear interpolation* of quaternions, or slerp in shorthand, is used [37]. Slerp is also a linear interpolation scheme, but the interpolation is performed on the sphere of quaternions. The interpolated quaternion q at time t is given by

$$q = q_{n-1}(q_{n-1}^{-1}q_n)^u, \tag{3.27}$$

where q_{n-1} is the attitude at time t_{n-1}, q_n is the attitude at t_n, and u, similar to the linear interpolation above, are found by

$$u = \frac{t - t_{n-1}}{t_n - t_{n-1}}. \tag{3.28}$$

The slerp interpolation scheme is linear, yielding a piecewise constant angular velocity and a non-defined angular acceleration, and should be considered to be improved in further work.

In addition to the state of the AUV, the covariance of the states from the discrete-time state estimation has to be interpolated. The only data available in the dataset is the variance of the states and no covariances. This is not optimal since the covariance information is omitted, but it makes the interpolation simpler as the states become independent.

The interpolation of the variance is done in the same manner as with the linear state interpolation and is given by

$$\sigma^2 = \sigma_{n-1}^2 + \frac{t - t_{n-1}}{t_n - t_{n-1}} \sigma_n^2, \tag{3.29}$$

where σ_{n-1}^2 is the variance at time t_{n-1}, σ_n^2 is the state at t_n, and we must have $t_{n-1} \leq t \leq t_n$. This is done for all states, where the resulting variances are put together in a diagonal matrix.

Chapter 4

Swath processing

Before a sonar measurement can generate a cartesian map, unwanted artifacts must be removed, and equations for mapping the sonar measurement to the coordinates on the seabed from which it originates need to be derived. This chapter will present the three steps used in this thesis to correct the sonar measurements and derive the abovementioned equations. These three steps for processing the sonar measurements build on the methods presented in [1], but an additional roll and pitch correction is derived and added. Throughout the chapter, the methods are tested on the real-world training dataset.

4.1 Pitch and roll correction

An assumption of zero roll and pitch is sometimes made for AUVs mapping the seafloor, as this is simple and often a valid assumption [17, 26]. However, this thesis derives swath processing steps that consider a pitch and roll different from zero for increased accuracy. This section will derive the equations for pitch and roll correction, also considering a sensor offset between the sonar transducers and the body frame of the AUV. In addition, equations for finding the world coordinate of each bin are derived.

Firstly, we find the transformation between the body frame and the sonar transducers. We assume that the acoustic axis of both transducers lay in a plane parallel to the yz-plane of the AUVs body frame and that the transducers are mounted symmetrically around the xz-plane of the body frame. First, looking at the starboard transducer, the transformation between the body frame and the starboard transducer, such that the acoustic axis lies in the xz-plane of the transducer frame, is

$$\boldsymbol{T}_{b}^{s_{stb}} = \begin{bmatrix} 0 & 1 & 0 & x_s \\ -1 & 0 & 0 & y_s \\ 0 & 0 & 1 & z_s \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.1}$$

For the port transducer, the same transformation is

$$\boldsymbol{T}_{\boldsymbol{b}}^{\boldsymbol{s}_{port}} = \begin{bmatrix} 0 & -1 & 0 & x_s \\ 1 & 0 & 0 & -y_s \\ 0 & 0 & 1 & z_s \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.2}$$

In these transformations, $\boldsymbol{x}_{\boldsymbol{s}_{stb}} = [x_s, y_s, z_s]^T$ is defined as the sonar offset, the translation between the origo of the body frame and the starboard transducer. Since the transducers are assumed to be mounted symmetrically around the xz-plane of the AUV, the translation between the origo of the body frame and the port transducer is $\boldsymbol{x}_{\boldsymbol{s}_{port}} = [x_s, -y_s, z_s]^T$.

Secondly, the altitude of the transducers, h_t, has to be corrected for pitch and roll. The altitude of the AUV is typically measured in the direction of the body's z-axis, as this is typical for DVLs. Therefore, the measured altitude, h_m, should be transformed from the body frame to world coordinates, and the z-coordinate extracted. Using the transformation defined in (3.6), here denoted as $\boldsymbol{R}_{\boldsymbol{b}}^{\boldsymbol{w}}$, the true altitude of the AUV is given by

$$h = (\boldsymbol{R}_{\boldsymbol{b}}^{\boldsymbol{w}} \boldsymbol{z}_{\boldsymbol{h}})^{\boldsymbol{T}} \boldsymbol{e}_{\boldsymbol{3}} = h_m \cos(\phi) \cos(\theta), \tag{4.3}$$

where $\boldsymbol{z}_{\boldsymbol{h}} = [0, 0, h_m]^T$ is the altitude measurement vector, $\boldsymbol{e}_{\boldsymbol{3}} \equiv [0, 0, 1]^T$, ϕ is the roll angle and θ the pitch of the AUV. Using the sensor offset $\boldsymbol{x}_{\boldsymbol{s}}$, the height of the transducers can then be found by

$$h_{t_{stb}} = h_m - (\boldsymbol{R}_{\boldsymbol{b}}^{\boldsymbol{w}} \boldsymbol{x}_{\boldsymbol{s}_{stb}})^{\boldsymbol{T}} \boldsymbol{e}_{\boldsymbol{3}} = (\boldsymbol{R}_{\boldsymbol{b}}^{\boldsymbol{w}} (\boldsymbol{z}_{\boldsymbol{h}} - \boldsymbol{x}_{\boldsymbol{s}_{stb}}))^{\boldsymbol{T}} \boldsymbol{e}_{\boldsymbol{3}}, \tag{4.4}$$

and

$$h_{t_{port}} = (\boldsymbol{R}_{\boldsymbol{b}}^{\boldsymbol{w}} (\boldsymbol{z}_{\boldsymbol{h}} - \boldsymbol{x}_{\boldsymbol{s}_{port}}))^{\boldsymbol{T}} \boldsymbol{e}_{\boldsymbol{3}}. \tag{4.5}$$

Thirdly, a change in roll angle will affect the ground range r_g and the range of the first bottom return r_{fbr}. Equations for calculating these quantities under the influence of roll have to be derived. The roll angles are assumed to be small, making the derived ranges both positive and finite. The situation is shown for the starboard transducer in Figure 4.1, where Figure 4.1a shows the situation with zero roll. Figure 4.1b shows the situation with a non-zero roll, and it is evident from the figure that the ground range of the acoustic axis is shorter than in the situation with zero roll. If we use the corrected transducer altitude in (4.4) and (4.5), the calculation of the ground range in (2.7) is still valid. However, (2.8) has to be corrected for roll. Looking at Figure 4.1b and making geometrical considerations, the corrected slant range of the first bottom return for port and starboard can be written as

$$r_{fbr_{stb}} = \frac{h_{t_{stb}}}{\sin(\phi + \beta + \frac{\alpha_v}{2})}, \tag{4.6}$$

and

$$r_{fbr_{port}} = \frac{h_{t_{port}}}{\sin(-\phi + \beta + \frac{\alpha_v}{2})}, \tag{4.7}$$

where ϕ is the roll angle of the AUV, β is the mounting angle from the body y-axis of the sonar, and α_v is the vertical beamwidth of the sonar.

Lastly, we want to find the world coordinates on the seafloor of each of the bins in the sonar. To do this, we define a reference frame, $p_s \in SE(3)$, for the starboard and port transducer, such that we only need the pose of the AUV and the ground range of each bin to find its world coordinates on the seabed.

The frame, p_s, is found by first projecting the pose of the transducer in the direction of the z-axis of the world frame in the xz-plane of the sonar. Secondly, a projection in the direction of the sonar's z-axis in the sonar's yz-plane is performed. This is shown in Figure 4.1 and Figure 4.2, respectivly. In addition, the frame p_s is transformed to have zero roll and pitch. This way, the ground range of a bin will coincide with the x-axis of the p_s frame. Using geometrical considerations when looking at Figure 4.2b, it is evident that the area on the seabed covered by the sonar beam changes with the pitch of the AUV. However, since both the pitch angle and the vertical beamwidth, α_v, are assumed to be small, we approximate this area to be constant with changes in pitch. The transformation $p_{s_{stb}}$ to the starboard transducer is

$$\boldsymbol{T}^{\boldsymbol{s}}_{\boldsymbol{p}_{s_{stb}}} = \begin{bmatrix} \boldsymbol{R}(\theta)\boldsymbol{R}(\phi) & \boldsymbol{0} \\ \boldsymbol{0}^{\boldsymbol{T}} & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & h_t \tan(\theta) \\ 0 & 0 & 1 & -h_t \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{4.8}$$

and for the port, we have

$$\boldsymbol{T}^{\boldsymbol{s}}_{\boldsymbol{p}_{s_{port}}} = \begin{bmatrix} \boldsymbol{R}(\theta)\boldsymbol{R}(\phi) & \boldsymbol{0} \\ \boldsymbol{0}^{\boldsymbol{T}} & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & -h_t \tan(\theta) \\ 0 & 0 & 1 & -h_t \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.9}$$

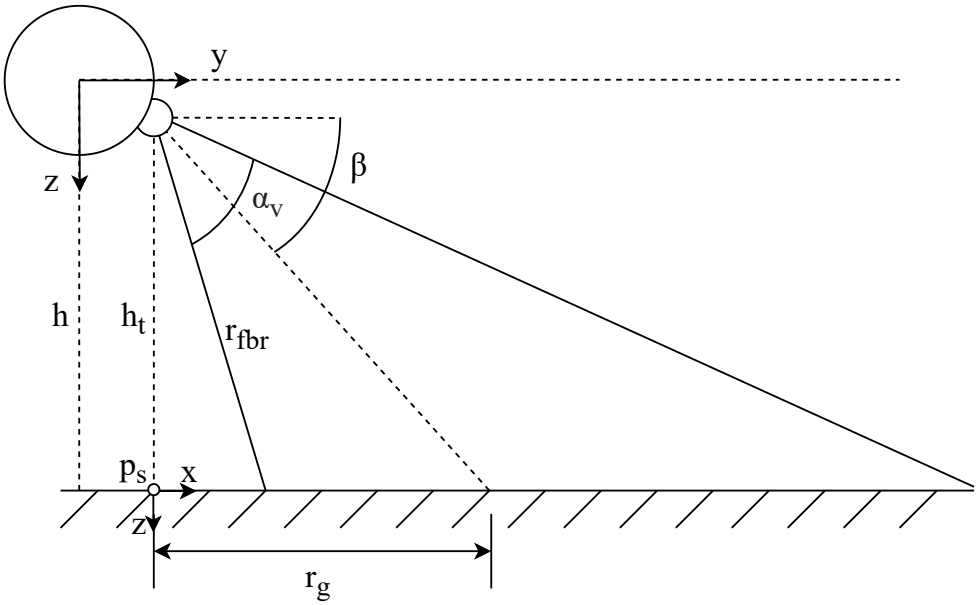
Then, the world coordinate of a bin from the starboard transducer, given its ground range, r_g, is

$$\boldsymbol{b}_{\boldsymbol{w}} = \boldsymbol{T}^{\boldsymbol{w}}_{\boldsymbol{b}} \boldsymbol{T}^{\boldsymbol{b}}_{\boldsymbol{s}_{stb}} \boldsymbol{T}^{\boldsymbol{s}_{stb}}_{\boldsymbol{p}_{s_{stb}}} \boldsymbol{z}_{\boldsymbol{r}_g}, \tag{4.10}$$

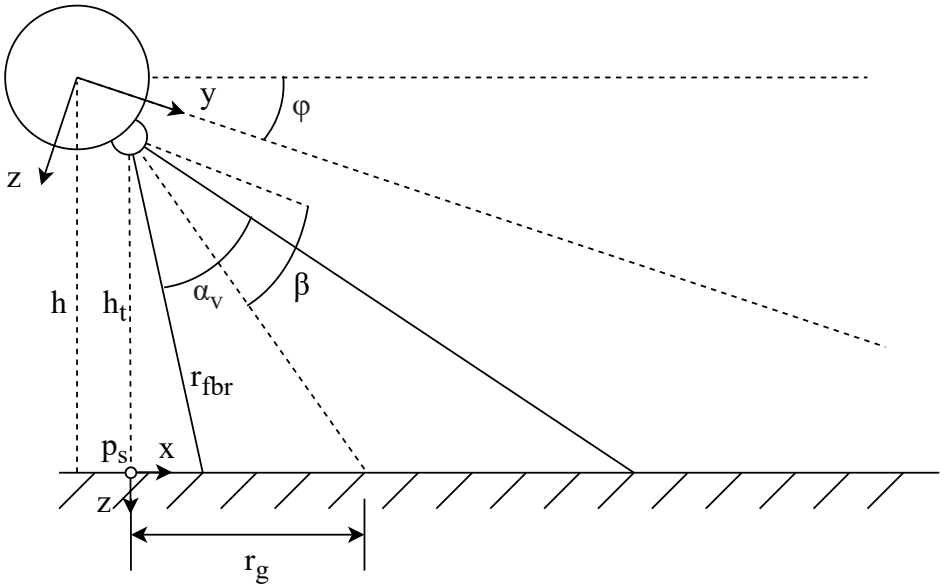
where $\boldsymbol{z}_{\boldsymbol{r}_g} = [r_g, 0, 0, 1]^T$, and $\boldsymbol{T}^{\boldsymbol{b}}_{\boldsymbol{s}_{stb}} = (\boldsymbol{T}^{\boldsymbol{s}_{stb}}_{\boldsymbol{b}})^{-1}$.

4.2 Blind zone removal

The sonar starts to record the backscatter as soon as the sonar pulse is transmitted. Hence, the first bins will only contain background noise, as some time will be needed for the acoustic pulse to travel to the seabed and for the backscatter to travel back to the transducer. These bins are often referred to as the *blind zone* and are characterized by low intensity. Since the bins only contain background noise, we want to remove them before further processing the swath.

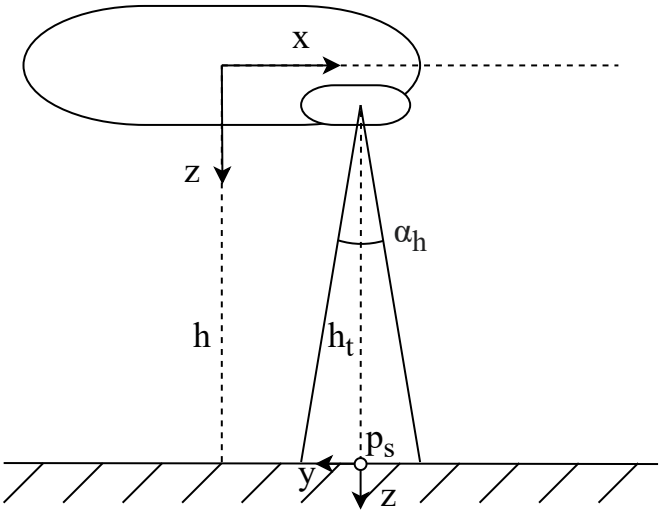


(a) Starboard transducer and AUV with zero roll.

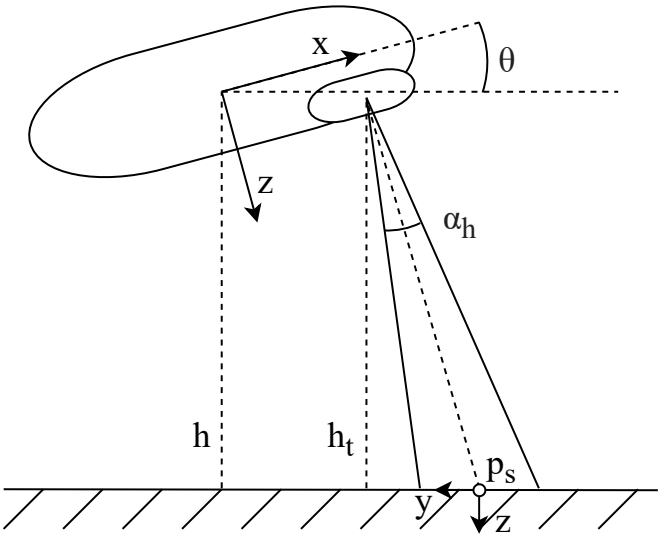


(b) Starboard transducer and AUV with non-zero roll.

Figure 4.1: The figure shows the sonar's measurement geometry with a zero and non-zero roll angle of the AUV. Note how the ground range is shorter in Figure 4.1b.



(a) Side scan sonar mounted on AUV with zero pitch.



(b) Side scan sonar mounted on AUV with non-zero pitch.

Figure 4.2: The figure shows the sonar's measurement geometry with a zero and non-zero pitch angle of the AUV. Note how the frame p_s changes position with the pitch angle θ.

The blind zone starts in the first measured bin and ends with the first bottom return. It is, therefore, typical to remove the blind zone by detecting the first bottom return and removing the bins up to this point.

In [38], two different methods are proposed for finding the first bottom return, both relying on the sonar measurement itself. In the first method, a cubic spline regression is applied to the log of the measured intensity to estimate the nadir center. The first bottom return is found as the first bin above a tuning threshold. The second method utilizes a moving average filter and a tuning threshold similar to the first method.

In [17], instead of relying on the data itself, a pure geometrical blind zone removal algorithm is presented. The method presented calculates the range of the first bottom return, similar to (2.8). However, instead of the slant range of the first bottom return, the ground range of the first bottom return is used, as the blind zone removal is performed on slant range corrected swaths. Such blind zone removal, based on purely geometrical considerations, has the advantage of being robust against errors and anomalies in the sonar data and is therefore used in this thesis.

In this thesis, the blind zone removal algorithm presented in [17] is adopted for swaths that are not slant range corrected, and pitch and roll correction is implemented. First, (4.6) and (4.7) are used to calculate the slant range of the first bottom returns for port and starboard. Then, bins with a slant range shorter than $r_{fbr_{stb}}$ in the starboard part of the swath and $r_{fbr_{port}}$ in the port part of the swath are removed. This is performed by simply changing out the intensity values in the bins with an invalid value.

The results of both roll and pitch corrected and non-corrected blind zone removal of the 1000 first swaths in the training dataset are shown in Figure 4.3. The actual removal is not performed; rather, the area to be removed is displayed with the red and green lines in the plot. In addition, two light blue colored lines are shown in the image. These lines represent the bins where we expect the first backscatter to appear. These bins are found by calculating the bin corresponding to a slant range that equals the altitude of the AUV. As the figure shows, the calculation of the first possible backscatter does not match the first measured backscatter.

The reason for the discrepancy between the calculated first possible backscatter and the measured first backscatter return in Figure 4.3 was sadly not possible to find. First, the calculation was thoroughly tested by contacting the LAUV manufacturer to verify the sonar offsets and mounting angles. Secondly, the sound speed was investigated. The LAUV is not measuring the sound speed, but the gathered data was used to estimate it. It was found to be approximately 1 % below the constant sound speed used by the sonar of $1500\,\mathrm{m\,s^{-1}}$. If the sound speed were the reason for the discrepancy, the actual sound speed would need to be higher than the constant sound speed used by the sonar. Since that was not the case, the sound speed is unlikely to be the reason for the discrepancy. Lastly, the sonar manufacturer was contacted, but unfortunately, they could not point out any plausible reason for the results.

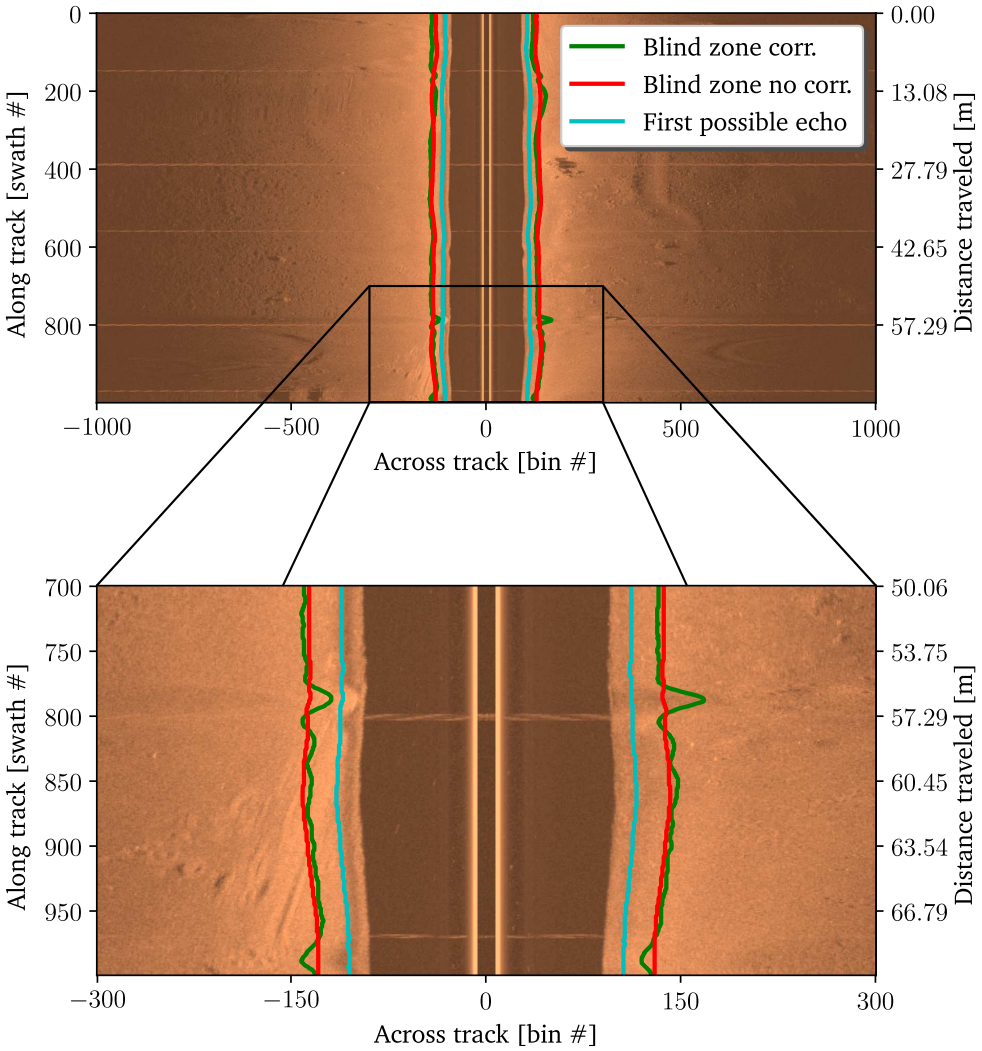


Figure 4.3: The figure shows the part of the sonar image to be removed by the blind zone removal, both with and without pitch and roll correction. The sonar image is constructed from the 1000 first swaths of the training dataset. In addition to the blind zone removal, the light blue line represents the bin where we expect the first backscatter to appear. This is done by finding the bin corresponding to a slant range that equals the altitude of the AUV.

The natural next step would have been to test the setup in a pool to verify the results in a controlled environment. However, as AURLab collected the data and the author has no knowledge of how to operate the LAUV, this step would demand a lot of time and the possibility that no solution would be found. Therefore, this was not performed.

As the error appears to be constant or close to constant, a solution could have been to correct the blind zone removal by a constant. However, as we don't know

the reason for the error, the solution would have needed to be verified for different altitudes. Therefore, further investigation of the problem is left as further work, and the method is kept as presented.

Investigating the blind-zone removal with and without pitch and roll correction in Figure 4.3, we find the pitch and roll correction to give the best results. Examining the sonar image at the light blue line in the left portion of the zoomed-in image, we see a darker line appearing after the first backscatter return. The hypothesis is that this darker line stems from a directivity that lies between the side lobe and the main lobe of the sonar, something also suggested by the sonar manufacturer [30]. This is not evident in the right part of the image, but this is most likely due to a small roll angle of the AUV stemming from the spinning propeller's torsion forces.

Again examining the left portion of the image, the bright bins on the left side of the darker strip most likely stem from the main lobe. If a right translation of the starboard blind zones were to be performed, we see that the corrected blind zone is a better fit for the pattern appearing. This shows that the pitch and roll corrected blind zone is likely to better represent the actual first bottom return than the non-corrected one. Therefore, pitch and roll correction is used in the rest of this thesis.

4.3 Intensity normalization

Intensity attenuation of the acoustic wave is an unwanted artifact in the measured signal that we want to counteract with intensity normalization. Intensity normalization can be performed on both a sonar image and an individual swath, where the latter is performed in this thesis. The framework presented next is for general sonar images but applies to individual swaths by considering one swath as a sonar image with a height of one. Some of the methods presented would not work, as they are considering pixels from several swaths simultaneously, but some are also possible to adapt to work on a single swath.

In [39], it is proposed that the sonar image $I(x, y)$ can be decomposed into a reflectance map $R(x, y)$ and an illumination map $L(x, y)$, resulting in $I(x, y) = R(x, y) * L(x, y)$, where the operator $*$ denotes element-wise multiplication. Since the information about the seabed lies in the reflectance map, we want to find it by estimating the illumination map and combining it with the original image. Thus for some estimated illuminance map $\hat{L}(x, y)$, the intensity normalized sonar image can be found by

$$I'(x, y) = \frac{I(x, y)}{\hat{L}(x, y)}. \tag{4.11}$$

In [39], it is proposed to estimate the illuminance map by $\hat{L}(x, y) = I(x, y) \otimes F(x, y) + a$, where F is a smoothing filter function and a is a constant. A mean filter and a bilateral filter are proposed as the filtering function. This was adapted to work on individual swaths in this thesis and briefly tested. However, the initial

results did not improve compared to the chosen intensity normalization method, and further testing was halted. The work in [17] proposed to model the seabed as a Lambertian surface and use the resulting equation to estimate the illuminance map.

This thesis uses smoothing cubic splines to approximate the illuminance map, as proposed by [38]. Unlike the abovementioned methods, the method uses polynomial fitting to estimate the illuminance map and is performed on individual swaths. The illuminance map is estimated by minimizing

$$p\sum_{i=1}^{n}|I(i)-L(i)|^2+(1-p)\int_1^n|D^2L(t)|^2dt, \tag{4.12}$$

where the first term is the *error measure* and the second the *roughtness measure*. Furthermore, p is the smoothing parameter, D^2L denotes the second derivative of L, and n is the number of bins. The smoothing cubic spline intensity normalization was tested for the same transducer used to collect the data in this thesis by [1].

Figure 4.4 shows the result of performing intensity normalization using smoothing cubic splines on the blind-zone removed sonar image from the 1000 first swath of the training dataset. As in [1], we use $p = 1 \cdot 10^{-6}$ since the implementation uses the same sonar transducer as in this thesis. The normalized image at the bottom appears to have normalized intensities, and the horizontal lines in the unnormalized top image are almost invisible. These artificial lines stem from acoustic communication with a surface vessel from when the data was collected. For the experienced eye, it is also evident that the dynamic range in the normalized image is lower, further away from the AUV, than close to the AUV. This is natural, as the dynamic range in the original data also changes depending on the across-track distance to the AUV and the fact that the method cannot estimate the reflectance map perfectly.

4.4 Slant range correction

The final step of the swath processing pipeline is to perform slant range correction of the swaths. The swaths in the earlier steps, and hence the displayed sonar images, have their bins equally spaced in their slant range with the slant range resolution δ_s. To be able to map the swaths to the seabed, we want to correct the swath such that the bins are equally spaced relative to their ground range, i.e., transform the spatial spacing between the bins to a ground range resolution δ_g.

In [17], a slant range correction algorithm using interpolation is suggested, and the algorithm is adapted to use in this thesis with pitch and roll correction. As showed in Section 2.3, the ground range, assuming a flat seafloor, can be found by $r_g(r_s,h_t)=\sqrt{r_s^2-h_t^2}$. Flipping the equation, the slant range can be found by $r_s(r_g,h_t)=\sqrt{r_g^2+h_t^2}$. The algorithm presented here performs slant range correction individually on the port and starport part of the swath, as the range of the first bottom return r_{fbr}, and the transducer height h_t, are not necessarily equal

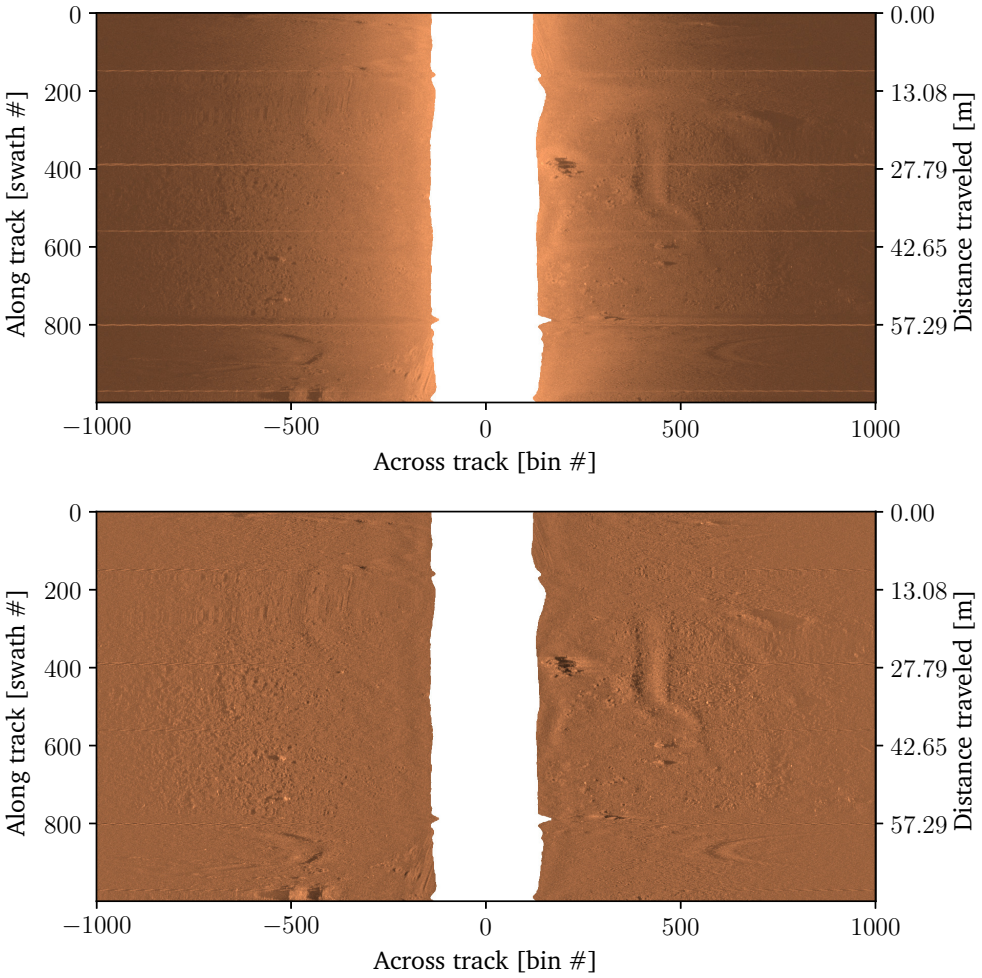


Figure 4.4: The top image shows the blind-zone removed sonar image with raw intensity constructed from the 1000 first swaths of the training dataset. The bottom image shows the same image normalized using smoothing cubic spline normalization. Note how the artificial lines are almost invisible in the normalized image.

for each side. Providing either the starboard or port portion of the swath as S, the corresponding first bottom return r_{fbr}, transducer height h_t, and the sonar range $r_{s_{max}}$, the slant range corrected swath S' can be found by Algorithm 1.

Figure 4.5 shows the sonar image before and after the slant range correction, and it is evident the image has been decompressed close to the sonar. We again see how the pitch and roll correction affects the image. However, as there is no ground truth of the seabed, it is not possible to say anything more about the correctness of the image. Two banana-formed shapes can be seen by examining the image around swath 900. As we will see in the next chapter, the AUV is turning at this point, and we need cartesian map generation to more accurately represent the map of the seabed.

Algorithm 1 Slant range correction

Input: $S, r_{fbr}, r_{s_{max}}, h_t$
Output: S'

$S' \leftarrow \varnothing$
$r_{g_{min}} \leftarrow r_g(r_{fbr}, h_t)$
$r_{g_{max}} \leftarrow r_g(r_{s_{max}}, h_t)$
$\delta_g \leftarrow \delta_s$
for $r_g \leftarrow r_{g_{min}}$ to $r_{g_{max}}$ step δ_g **do**
    $r \leftarrow \frac{r_s(r_g, h_t)}{\delta_s}$
    $w_1 \leftarrow r - \lfloor r \rfloor$
    $w_2 \leftarrow 1 - w_1$
    $S'(|\frac{r_g}{\delta_g}|) \leftarrow w_2 S(\lfloor r \rfloor) + w_1 S(\lceil r \rceil)$
end for

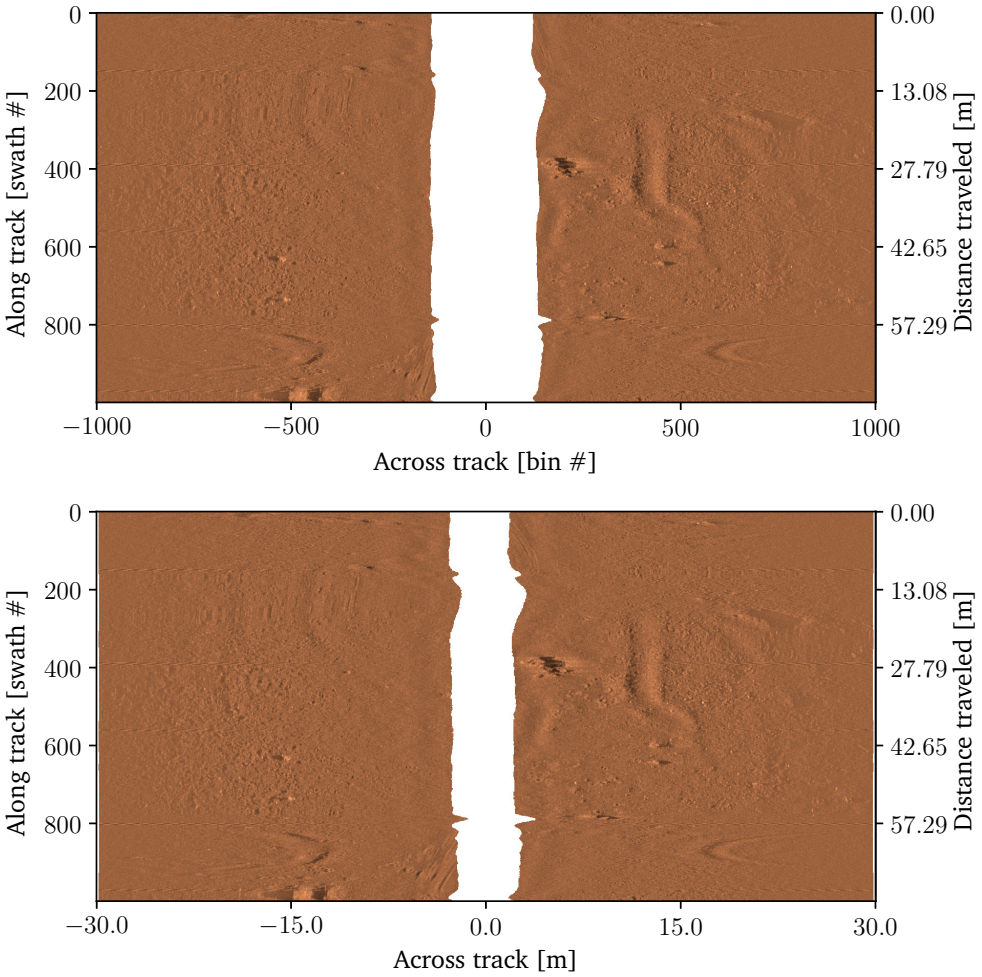


Figure 4.5: The top image shows the normalized image constructed from the 1000 first swaths of the training dataset. The lower image shows the result of slant range correction on the top image. Note how each side of the image is decompressed towards the middle.

Chapter 5

Cartesian map generation

This chapter will present a motivation about why we want cartesian maps compared to traditional waterfall images, three different cartesian map generation algorithms, and a comparison of the algorithms. The first algorithm is based on Kd-trees and nearest neighbors search. The second algorithm makes probabilistic considerations when building a cartesian map. The last algorithm is a novel probabilistic map generation algorithm that builds on the second algorithm. It has improved computational time and is capable of real-time performance. Lastly, all three algorithms are compared against each other.

5.1 Why cartesian map generation?

Traditionally, waterfall sonar images have been widely used in side scan sonar applications, typically disregarding sonar data when heading changes are performed [40–43]. This is simple, and as AUVs usually travel in a lawnmower pattern as they survey the seabed, only a small portion of the trajectory is disregarded due to turning. Waterfall images are created by stacking swaths together, one after the other, to create an image, as shown in Figure 5.1.

Further, good-quality waterfall images also depend on a constant altitude, speed, roll, and pitch, where small deviations from constant values can lead to anomalies in the waterfall image. However, if AUVs are to be used not just for surveying but also in more general applications, putting such constraints on the movement of the AUV may not be optimal from a navigation point of view. This thesis, therefore, seeks a solution that does not depend on constant states for good results.

Reconstruction of the seafloor's bathymetry, or the elevation of the seafloor, has been performed [44, 45], and is a possible solution for map generation. However, the reconstruction typically involves several steps that might be computationally intensive, and a reconstruction method runs the risk of not being fast enough in a SLAM pipeline. Therefore, reconstruction of the seafloor's bathymetry is not chosen to generate maps for the SLAM pipeline.

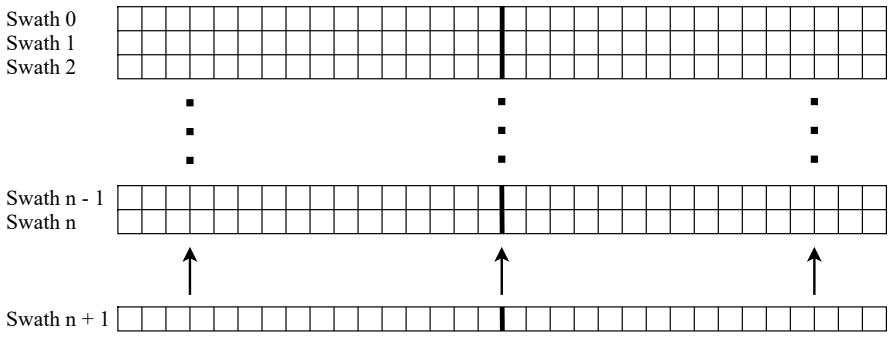


Figure 5.1: The figure shows how swaths are stacked together to form a waterfall image.

Cartesian map generation in this thesis is the process of generating a 2D map of the seabed from side scan sonar swaths. Unlike waterfall sonar images, Cartesian map generation does not require a steady survey heading, speed, altitude, pitch, or roll. Instead, it maps each bin to its corresponding geometric position on the seafloor to create a 2D cartesian intensity map. Figure 5.2 shows how one swath maps into the seafloor and the overlayed map cells.

For Cartesian map generation, the mapping of each bin to the seafloor is performed by combining the state estimation with the measurement geometry and the flat seafloor assumption presented in Section 4.2. It is important to note that since the map generation relies on the state estimation, so does the correctness of the generated map. In addition to mapping the bins to the seafloor, some type of interpolation or filtering is needed to find the intensity of the map cells. It is first needed to determine the cell intensity in scenarios where several bins map to the same cell, and second, to fill in the visual gaps between swaths where no bin covers the cells.

Figure 5.3 shows both a cartesian map and a waterfall sonar image constructed by the 1000 first swaths of the training dataset. As we can see, there is a big difference in the parts of the trajectory where turning occurs. In addition, the scaling is drastically different, making it much easier to infer the scale of objects in the Cartesian map as the axes are in meters.

Looking at the waterfall image at around swath number 900, two banana-like shapes appear. Examining the top of the cartesian map, we see that the banana-like form has been mapped to the straight, almost horizontal line spanning around 40 m. From the author's perspective, inferring the cartesian map's geometrical information is easier than for the waterfall image. The following sections will present different methods of generating such cartesian maps.

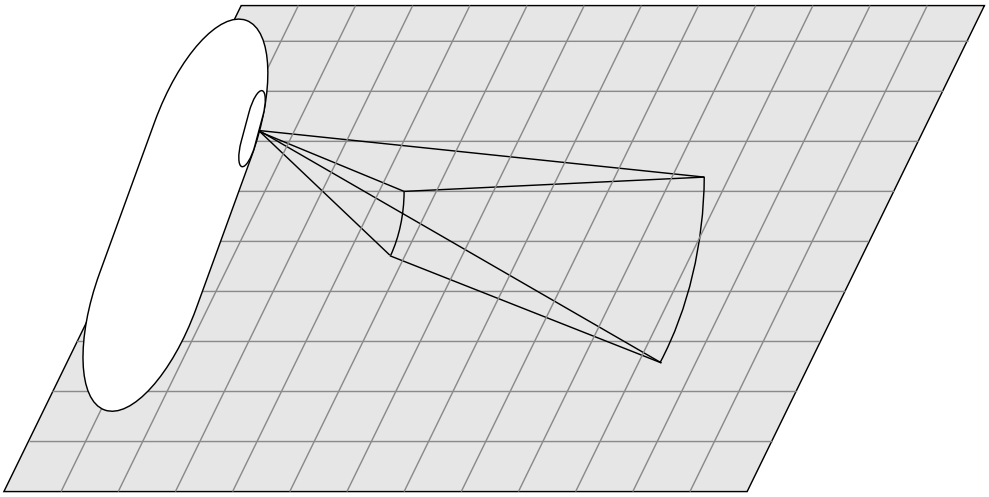
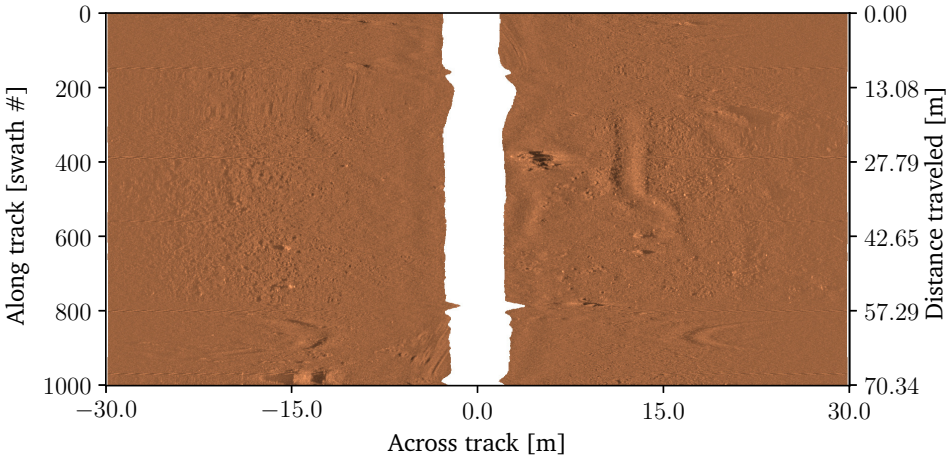


Figure 5.2: The figure shows how the starboard portion of the swath maps to the seabed and to the 2D grid on the seafloor that represents the cells of the cartesian map.

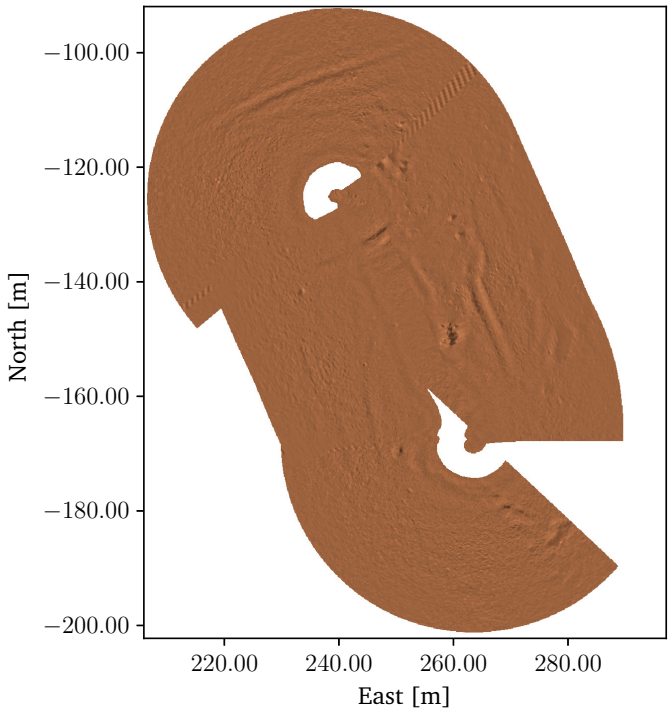
5.2 k nearest neighbor cartesian map generation

In [1], a k Nearest Neighbor (kNN) cartesian map generation algorithm is presented. The algorithm involves two main steps. First, a K-d tree consisting of the world coordinates of the bins in all swaths is built. Second, the k_{nn} nearest neighbors are found for each cell in the map. To only include intensities spatially close intensities, only the nearest neighbors inside each cell's range threshold, r_{max}, are considered. Further, a maximum variance threshold, σ^2_{max}, is imposed on the nearest neighbors inside the range threshold. If the variance is above the threshold, the intensity is set to the nearest neighbors' 10^{th} percentile. This is done to prevent high-intensity outliers in the map. On the contrary, if the variance is below the threshold, the resulting intensity value of the cell is found as the mean of the nearest neighbors inside the range threshold. Let the map M be defined by its number of rows n_r, the number of columns n_c, its origin M_0, and its resolution δ_m. Furthermore, let S be the set containing all swaths to generate a map from. The cartesian map can then be generated by Algorithm 2.

Computational time is essential when a cartesian map generation algorithm is to be used in a real-time online SLAM pipeline, and a computational complexity analysis can help analyze the algorithm at hand. A central element of the kNN cartesian map generation algorithm is the K-d tree [46]. A K-d tree represents a set of N points in a K-dimensional space. In the case of the tree being *semidynamic*, meaning that only deletion and undeletion of points are allowed, we can build the



(a) The figure shows a waterfall sonar image processed by the swath processing in Chapter 4.



(b) The figure shows an example of a cartesian map generated with swaths processed by the methods in Chapter 4.

Figure 5.3: The figure shows both a waterfall sonar image and a cartesian map constructed from the same first 1000 swaths of the training dataset. There is a great difference between the two maps, even though they are both generated from the same swaths, especially in how the scale differs. In addition, it is easier to infer the geometry of the seabed in the cartesian map.

Algorithm 2 kNN map genaration

Input: $S, M_0, \delta_m, n_r, n_c, r_{max}, \sigma^2_{max}, , k_{nn}$
Output: M

$Q \leftarrow \varnothing$
$V \leftarrow \varnothing$
$M \leftarrow \varnothing$

for $m \in S$ **do**
  for $b \in m$ **do**
    $Q \leftarrow Q \cup$ GETBINWORLDPOS(b, m) ▷ Get pos. of bin in world.
    $V \leftarrow V \cup$ GETBININTENSITY(b)
  end for
end for
$T \leftarrow$ BUILDKDTREE(Q)

for $r \leftarrow 1$ to n_r **do**
  for $c \leftarrow 1$ to n_c **do**
    $q \leftarrow$ GETCELLWORLDPOS(r,c,M_0,δ_m) ▷ Get pos. of cell in world.
    $N \leftarrow$ GETNEARESTNEIGHBOURS(T,q,k_{nn})
    $L \leftarrow \varnothing$
    for $n \in N$ **do**
      if GETDISTANCE$(n,r,c) < r_{max}$ **then**
        $L \leftarrow V(n)$
      end if
    end for
    if GETVARIANCE$(L) < \sigma^2_{max}$ **then**
      $M(r,c) =$ GETMEAN(L)
    else
      $M(r,c) =$ GETQUANTILE$(L,0.1)$
    end if
  end for
end for

tree with a computational complexity of $O(KN + N\log(N))$ and perform a nearest neighbors search in constant time, $O(1)$.

In the first step of the kNN map generation algorithm, a kNN tree is built. Before the tree is built, data vectors of all the bin coordinates and intensities are put together. Because both obtaining the bin coordinates and intensity are constant in time, the computational complexity of this step is $O(SB)$, where S is the number of swaths, and B is the number of bins per swath. The number of points in the K-d tree built of the coordinates found is then $N = SB$. Additionally, as the kNN map generation algorithm generates 2D seabed maps, we have $K = 2$. Typically the number of swaths needed to build a meaningful map is a minimum of ten and typically larger, and we expect each swath to contain more than ten bins. Therefore, we can safely assume that $\log(SB) > K$ and, hence, the computational complexity of setting up the data vectors and building the K-d tree in the kNN map generation algorithm reduces to $O(SB + SB\log(SB)) = O(SB\log(SB))$.

The second step of the kNN map generation algorithm is to perform a nearest neighbor search for each cell in the map. In addition, range thresholding and variance calculation of the nearest neighbors are performed. Since the number of nearest neighbors used is small (in [1], $k_{nn} = 4$ is used), these operations, performed for each cell, are assumed to have constant time complexity. Then, the second step of the algorithm has a computational complexity of $O(RC)$, where R is the number of rows, and C is the number of columns in the map. To sum up, combining the computational complexity of the first and second sted results in a computational complexity of $O(SB\log(SB) + CR)$ for the kNN map generation algorithm.

Figure 5.4a shows a map generated with the kNN map generation algorithm. The first 500 swaths of the training data were used together with a map resolution of $\delta_m = 0.1\,\text{m}$, a range threshold of $r_{max} = 0.3\,\text{m}$, a variance threshold of $\sigma^2_{max} = 5 \cdot 10^{-3}$ and $k_{nn} = 4$. Since the same sonar transducer is used, the three latter parameters are identical to those used in [1]. Furthermore, a variance map is shown in Figure 5.4b. Each cell in the map corresponds to the variance of the nearest neighbors in the intensity map.

In Figure 5.5, maps generated with two different resolutions of $\delta_m = 0.05\,\text{m}$ and $\delta_m = 0.2\,\text{m}$ are shown. Comparing the two with the intensity map in Figure 5.4a, there is a minimal visual difference between a resolution of $\delta_m = 0.05\,\text{m}$ and $\delta_m = 0.1\,\text{m}$. However, for a resolution of $\delta_m = 0.2\,\text{m}$, the decreased resolution leads to a visually poorer map.

Since the number of cells, and potentially the computation time, increases quadratically with the map resolution, a tradeoff between computation time and resolution has to be made. In this thesis, a map resolution of $\delta_m = 0.1\,\text{m}$ is chosen as the preferred resolution and is used for the rest of the maps shown in this chapter.

5.3 Probabilistic map generation

Buguera et al. [17] present a method of probabilistic generation of a cartesian map, where a probabilistic map is created, and the information in it is combined with the measured intensity to form an intensity map. In this algorithm, the map consists of n_r rows and n_c columns of cells, where each cell, q, is a square with sides of size δ_m. Each cell has four corners, q_0, q_1, q_2, and q_3, starting with q_0 in the upper left corner, going clockwise around the corners. This is shown in Figure 5.6.

The probability map generated represents the probability of a cell being observed by the sonar [17]. It assumes that the slant range is deterministic and that the only stochastic variable is the horizontal observation angle, θ_{obs}. The intensity map represents the intensity of the cells. The different parts of the probabilistic map generation algorithm will be presented below.

The first step is to find an expression for the probability of measurement m observing cell q [17]. Let $q_i^m = (r_{g,i}^m, \theta_i^m)$ be the polar coordinates describing the position of corner i relative to the body frame of the AUV at the time measurement m was acquired. The algorithm here assumes that the sonar offset, as presented in Section 4.1 is zero, such that $\boldsymbol{p}_{\boldsymbol{s}_{stb}} = \boldsymbol{p}_{\boldsymbol{s}_{port}}$. The situation is shown in Figure 5.6 where the maximum observation angle, $\theta_{obs_{max}}^m$, and minimum observation angle, $\theta_{obs_{min}}^m$ of a cell, are shown. In addition, the minimum ground range, $r_{g_{min}}^m$, and the maximum ground range $r_{g_{max}}^m$ are denoted in the figure.

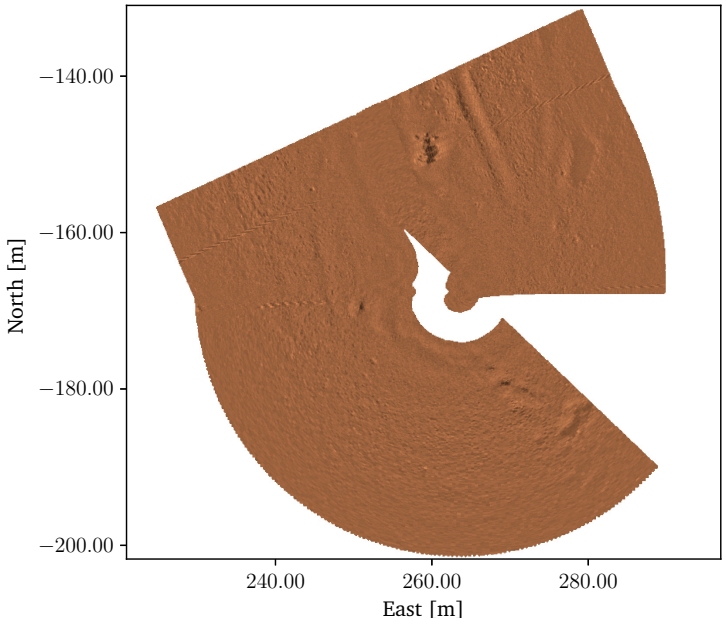
Since the probability of observation only depends on the minimum and maximum observation angle of a cell, the probability of measurement m observing a cell, q, can be written as

$$P^m(q) = \int_{\theta_{obs_{min}}^m}^{\theta_{obs_{max}}^m} p(\theta_{obs}) d\theta_{obs}. \tag{5.1}$$

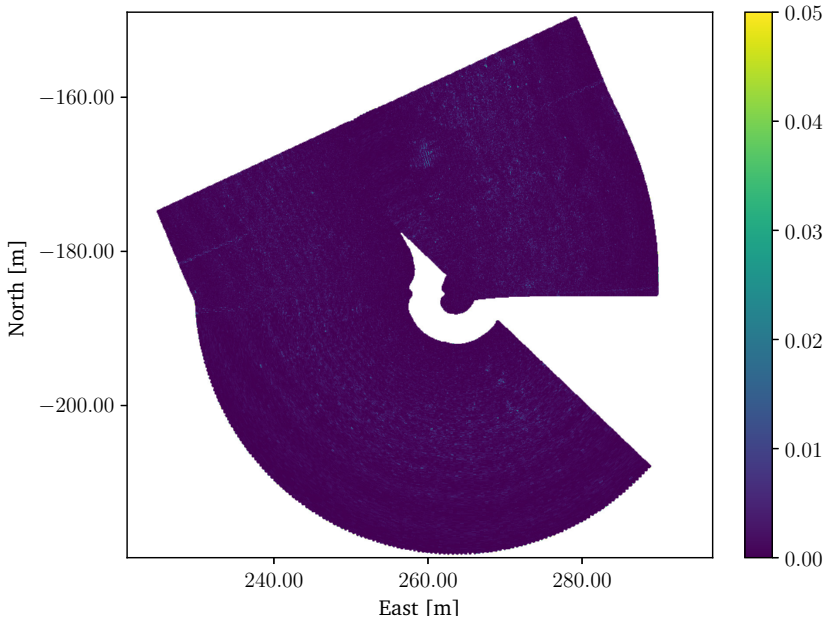
where $p(\theta_{obs})$ is the probability distrbution of θ_{obs}. In some situations, one or more of the corners of a cell can fall outside the minimum or maximum ground range. However, as long as one of the corners is inside the observed range, the cell is considered to have been observed and should therefore be considered.

Different probability distributions can be used for θ_{obs}. A Gaussian, triangular, and uniform distribution is suggested in [17]. The simpler triangular and uniform distributions are suggested to reduce the computational cost. However, in the work of this thesis, calculating the probability of observation using a Gaussian distribution had minimal impact on the computation time, so only a Gaussian distribution is presented and used.

The Gaussian distribution presented in [17] is centered around the acoustic axis, and the standard deviation is set to be $\alpha_h/2$. The distribution is then given

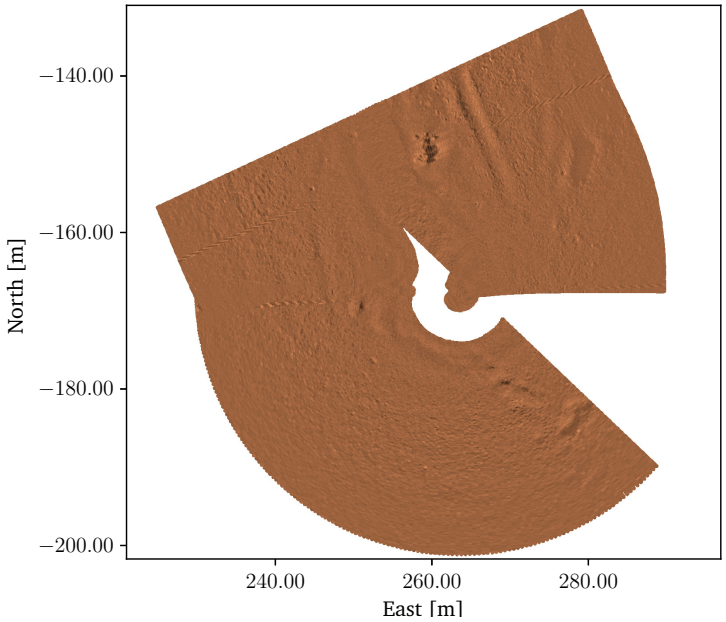


(a) Intensity map generated with kNN map generation algorithm.

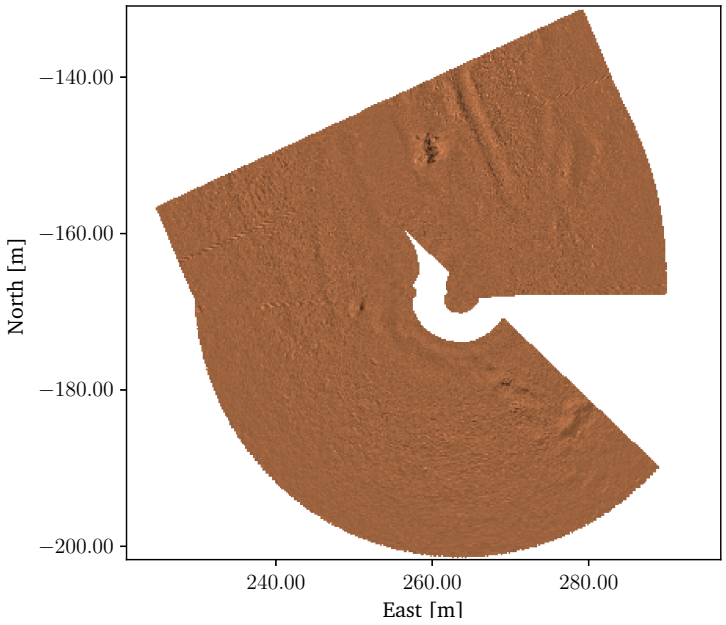


(b) Variance map from kNN map generation algorithm, where the color bar maps colors and variance.

Figure 5.4: The figure shows the resulting intensity and variance map generated from the 500 first swaths of the training dataset. The maps were generated with a map resolution of $\delta_m = 0.1\,\text{m}$, a range threshold of $r_{max} = 0.3\,\text{m}$, a variance threshold of $\sigma^2_{max} = 5 \cdot 10^{-3}$ and $k_{nn} = 4$.



(a) Intensity map generated with a map resolution of $\delta_m = 0.05\,\text{m}$.



(b) Intensity map generated with a map resolution of $\delta_m = 0.2\,\text{m}$.

Figure 5.5: The figure shows the resulting intensity maps generated with the kNN map generation algorithm, using the 500 first swaths of the training dataset and two different map resolutions. The maps were generated with a range threshold of $r_{max} = 0.3\,\text{m}$, a variance threshold of $\sigma^2_{max} = 5 \cdot 10^{-3}$ and $k_{nn} = 4$.

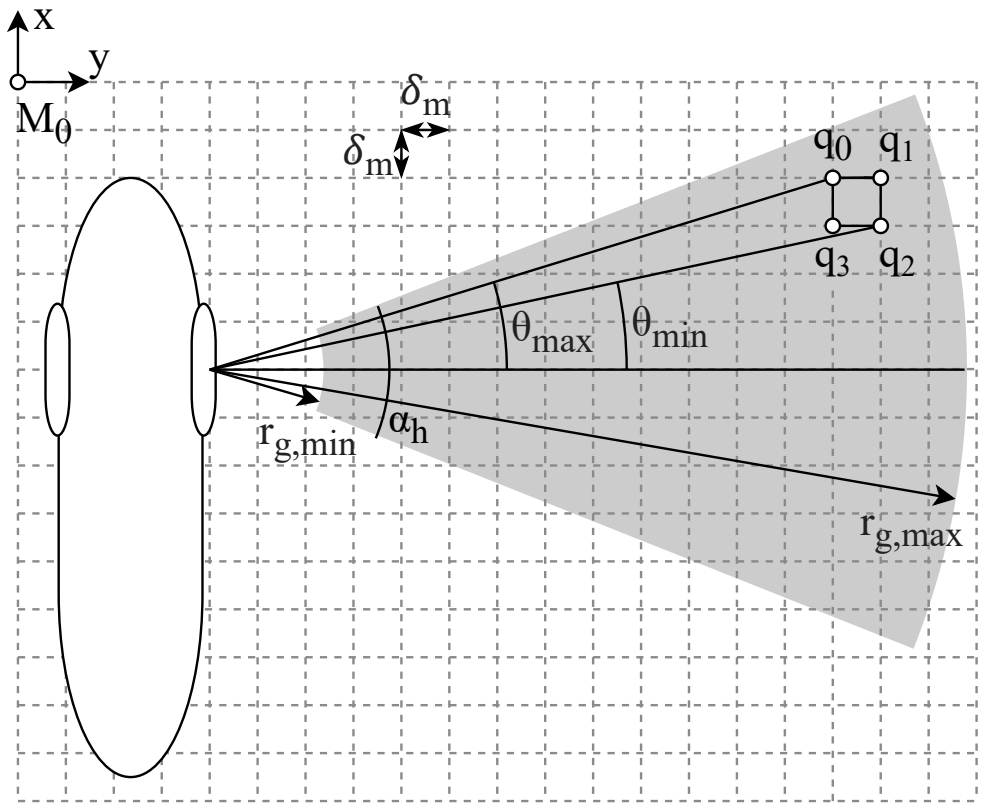


Figure 5.6: The figure shows an AUV from above, how its starboard transducer maps to the seabed, and the definitions used when generating cartesian maps using the probabilistic map generation algorithms. Remade from [17].

by

$$p(\theta_{obs}^{m}) = \frac{2}{\alpha_h \sqrt{2\pi}} \exp\left\{ \frac{-2(\theta_{\text{obs}}^{m})^2}{\alpha_h^2} \right\}. \tag{5.2}$$

As the Gaussian distribution is defined to have a non-zero probability on the interval $[-\infty, \infty]$, only cells with a probability of observation above a small threshold ϵ are considered to make the problem computationally tractable. Hence, if the observation probability falls below ϵ, the probability is set to zero. It is important to note that this is an assumption, and choosing a different ϵ could yield a different result.

Next, we want to find the probability of cell q being observed by all measurements used to generate the map [17]. This step is not actually needed to build the intensity map and is simply used for building the probability map. As described above, the probability of measurement m observing cell q is $P^m(q)$. Let all swats with a probability of observing cell q greater than ϵ be the set S_q. Considering all measurements in S_q and denoting the event that measurement m has observed cell q with E_q^m, where the probability of the event occurring is $P^m(q)$, the probability

of cell q being observed is given by

$$P(q) = P\left(\bigcup_{m \in S_q} E_q^m \right). \tag{5.3}$$

Assuming that the events E_q^m are mutually independent but not mutually exclusive, the probability in (5.3) can be rewritten as

$$P(q) = P\left(\bigcap_{m \in S_q} (E_q^m)^c \right)^c = 1 - \prod_{m \in S_q} (1 - P^m(q)), \tag{5.4}$$

where c denotes the complementary event. The probability map is created by calculating $P(q)$ for all cells in the map.

The intensity map uses the probabilities found by (5.1) and the measured intensities to generate a cartesian intensity map [17]. First, the measured intensity I^m in measurement m is linearly interpolated for each of the four corners of cell q, resulting in

$$V^m(q) = \begin{cases} 0, & m \notin S_q \\ \frac{1}{4} \sum_{i=0}^{3} \left(w_1 I^m \left(\lceil \frac{r_{g,i}^m}{\delta_g} \rceil \right) + w_2 I^m \left(\lfloor \frac{r_{g,i}^m}{\delta_g} \rfloor \right) \right), & m \in S_q, \end{cases} \tag{5.5}$$

where δ_g is the ground resolution of the measurements, $w_1 = \frac{r_{g,i}^m}{\delta_g} - \lfloor \frac{r_{g,i}^m}{\delta_g} \rfloor$, and $w_2 = 1 - w_1$.

The resulting intensity of cell q is given by

$$V(q) = \Big(\sum_{m \in S_q} P^m(q) \Big)^{-1} \sum_{m \in S_q} P^m(q) V^m(q), \tag{5.6}$$

where the normalization factor is not part of the algorithm presented in [17], but was found to be essential to get reasonable results. Picture the AUV being stationary, meaning that the number of measurements observing cell q grows linearly with time. Without the normalization factor, the same would happen to the cell intensity, giving an unreasonable high intensity. This is prevented by adding the normalization factor. The resulting intensity is a weighted average of the measurements that have observed the cell q, where the weight is the probability of observation.

The last step of the map generation is to fill the visual gaps in the map. Some cells have a probability of being observed below ϵ but lie in between two swaths, and hence, their intensity can be interpolated from the surrounding cells. In [17], this is done by creating polygonal meshes between the acoustic axis of all consecutive swaths. The cells inside the polygons that haven't been assigned an intensity value get interpolated. This is done by projecting a cell's four corners to the involved acoustic axes, and an intensity value is computed for each of them through

linear interpolation. In addition, the perpendicular distance from each corner to the acoustic axes is computed, and the resulting intensity is the weighted mean of the intensities, where the weights are inversely proportional to the corresponding perpendicular distance.

The implementation of the map generation algorithm from [17] in this thesis has a few differences from the original implementation. Firstly, the sonar offset is not assumed to be zero, and roll- and pitch-corrected swaths are used in the algorithm. Secondly, instead of the presented way of filling the visual gaps, this is done by using a Kd-tree and nearest neighbor search to all the non-assigned cells inside the min and max ground range, similar to the kNN map generation algorithm. This was chosen to save time in implementing the algorithm. Although it's probably not the most efficient way of filling the visual gaps, the total computation time spent on filling them is mostly small compared to the rest of the probabilistic map generation algorithm. Let the map M, as earlier, be defined by its number of rows n_r and columns n_c, its origin M_0, and its resolution δ_m. Furthermore, let all the swaths used to generate a map be the swaths in the set S. Then, the cartesian intensity map can be generated using Algorithm 3.

As with the kNN map generation algorithm, we want to analyze the computational complexity of the probabilistic map generation algorithm. We see from Algorithm 3 that the algorithm contains three nested for loops. The innermost contains two for loops, but both are at most performed S times, where S is the number of swaths used to generate the map. The two outer for loops are governed by the number of rows, R, and columns, C. Hence the computational complexity of the three nested for loops is $O(RCS)$. In addition, a KnnFill procedure is used to fill the visual gaps in the map. The procedure has the same computational complexity as the kNN map generation algorithm, $O(N\log(N)+RC)$, where N is the number of cells to be filled. However, as only a small part of the map needs to be filled, it is assumed that $N\log(N) < RCS$. Therefore the computational complexity of the probabilistic map generation algorithm is $O(RCS)$.

Figure 5.7 shows the resulting map using the probabilistic map generation algorithm on the 500 first swaths of the training dataset. For the map generation, a map resolution of $\delta_m = 0.1\,\text{m}$ and a probability threshold of $\epsilon = 0.1$ was used. For the KnnFill procedure, a distanse threshold of $r_{max} = 0.2\,\text{m}$ and a variance threshold of $\sigma^2_{max} = 5\cdot 10^{-3}$ was used. Comparing with the result in Figure 5.4a, it is not easy to spot any difference.

In Figure 5.8, the intensity map before interpolation by KnnFill and the probability map are shown. The probability threshold of $\epsilon = 0.1$ gave good results. A higher probability threshold than $\epsilon = 0.1$ led to many cells being interpolated. This is unwanted, as it, in sharp turns, can lead to missing intensities. On the contrary, a lower probability led to either none or only a few cells being interpolated. This is also unwanted because we do not want the swath to cover too many cells. Therefore, a probability threshold of $\epsilon = 0.1$ was chosen.

Algorithm 3 Probabilistic map generation

Input: $S, M_0, \delta_m, n_r, n_c$
Output: M

$M \leftarrow \varnothing$

for $r \leftarrow 1$ to n_r **do**
  for $c \leftarrow 1$ to n_c **do**
    $V \leftarrow \varnothing$
    $P \leftarrow \varnothing$
    $q_w \leftarrow$ GETCELLWORLDPOS(r,c,δ_m, M_0) ▷ Get pos of cell in world.
    for $m \in S$ **do**
      $m_w \leftarrow$ GETLOCALIZATION(m) ▷ Get pos. of AUV at meas. time.
      $q \leftarrow$ GETCELLBODYPOS(q_w, m_w) ▷ Get pos of cell in body.
      if $P^m(q) > \epsilon$ **then**
        $P \leftarrow P \cup P^m(q)$
        $V \leftarrow V \cup V^m(q)$
      end if
    end for
    $V_c \leftarrow 0$
    $P_c \leftarrow 0$
    for $v \in V, p \in P$ **do**
      $V_c \leftarrow V_c + vp$
      $P_c \leftarrow P_c + p$
    end for
    $M(r,c) \leftarrow P_c^{-1} V_c$
  end for
end for

$M \leftarrow$ KNNFILL(M)

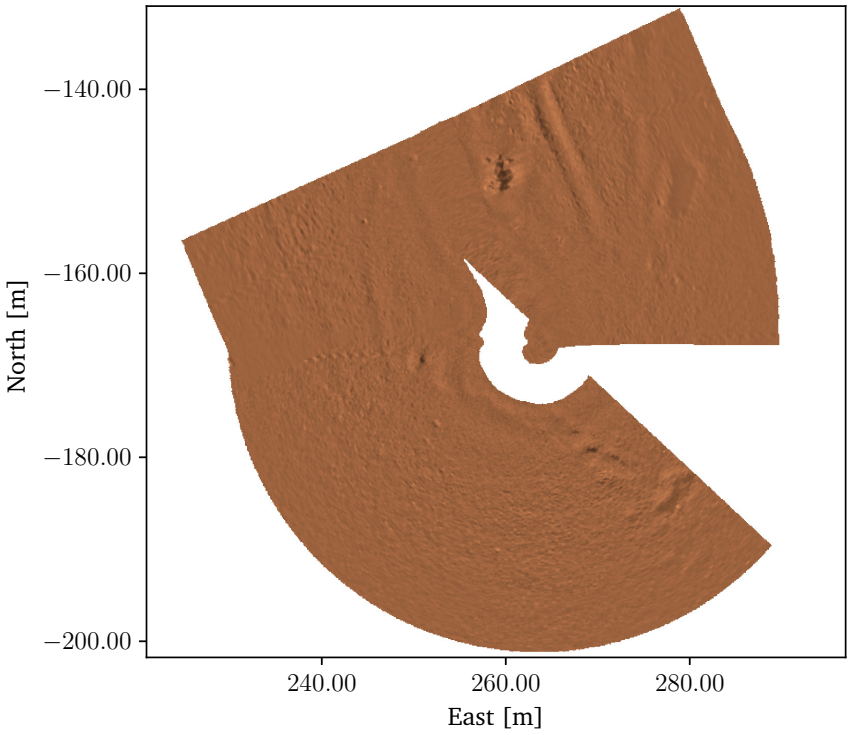
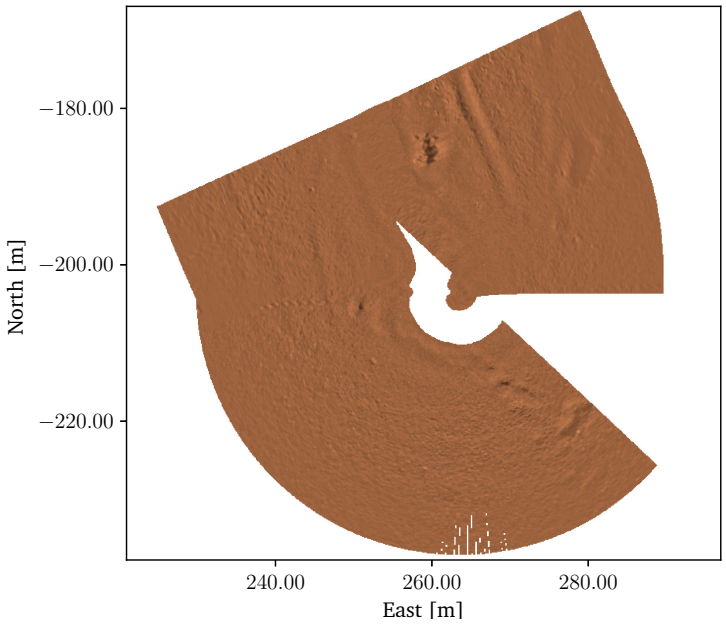
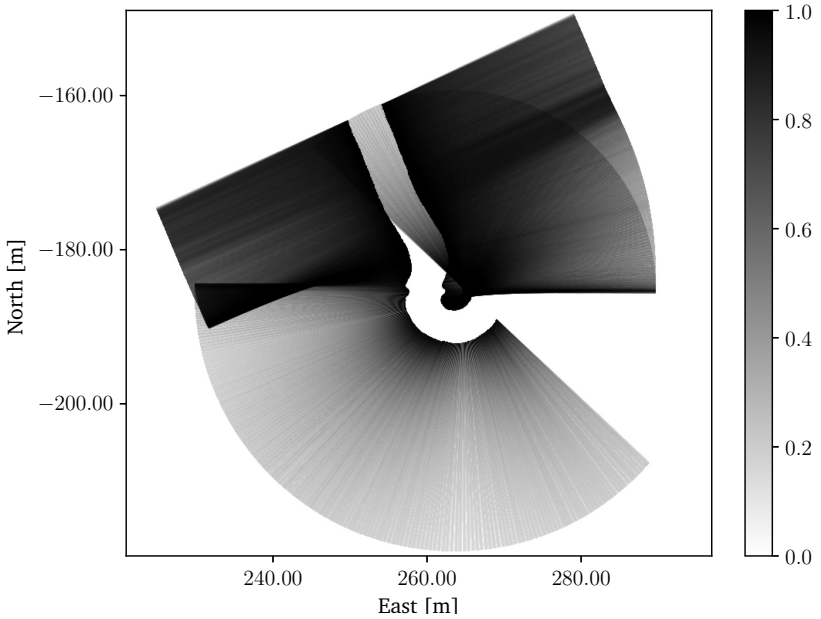


Figure 5.7: The figure shows the resulting intensity map generated from the 500 first swaths of the training dataset using the probabilistic map generation algorithm. The map was generated with a map resolution of $\delta_m = 0.1\,\text{m}$ and a probability threshold of $\epsilon = 0.1$.



(a) The map before the interpolation is performed.



(b) The probability map, where the color bar maps colors and probabilities.

Figure 5.8: The figure shows the intensity map before interpolation is performed and the probability map. The missing cells in the intensity map occur because the cells do not achieve a probability of being observed by a swath above the probability threshold ϵ and hence are not assigned an intensity value. The map is generated from the 500 first swaths of the training dataset using the probabilistic map generation algorithm, with a map resolution of $\delta_m = 0.1$ m and a probability threshold of $\epsilon = 0.1$.

5.4 Probabilistic map generation with decreased computational complexity

To achieve a probabilistic map generation algorithm that can be used in online real-time applications, improvements in the computational complexity of the probabilistic map generation algorithm in Section 5.3 have to be derived.

Examining Algorithm 3, we see that for each swath, the transformation from the body frame to a cell has to be calculated for every corner in each cell in the map. However, as the swath only covers a smaller portion of the map, great improvements can be achieved by reducing the number of transformations needed to be found.

In addition, neighboring cells share corners, meaning that the number of transformations can be further reduced for all neighboring cells by not calculating the transformations for all four corners for all cells but rather sharing the results between the cells. Using these observations, this section presents a novel probabilistic map generation algorithm with reduced computational complexity, building on the ideas from [17], to achieve a far better computational time.

The novel map generation algorithm presented turns the computation procedure of Algorithm 3 inside out, and instead of iterating the cells, calculating one and one intensity, one and one swath are iterated. The intermediate results from iterating the swaths are saved for calculating the resulting intensities at the end of the map generation. This is achieved by making use of the observations presented above.

Firstly, a flood fill method is utilized to only calculate the transformations between cells and the body frame for the cells with a probability of observation greater than ϵ (and the border around them). The calculated transformations are saved in a 2D array of size $(n_r + 1) \cdot (n_c + 1)$, representing all the corners in the map. In this way, no corner transformations are calculated more than once, as opposed to Algorithm 3.

Secondly, the novel map generation algorithm also considers the offset between the body and transducer frames. Lastly, since linear interpolation is done both for the bins in the slant range correction presented in Section 4.4 and in (5.5), the former interpolating step is dropped.

By dropping the slant range correction in Section 4.4, (5.5) has to be reformulated. Using the slant range to each of the four corners instead of the ground range, (5.5) is reformulated to

$$V^m(q) = \begin{cases} 0, & m \notin S_q \\ \frac{1}{4}\sum_{i=0}^{3}\left(w_1 I^m\left(\lceil \frac{r^m_{s,i}}{\delta_s} \rceil\right) + w_2 I^m\left(\lfloor \frac{r^m_{s,i}}{\delta_s} \rfloor\right)\right), & m \in S_q, \end{cases} \tag{5.7}$$

where δ_s is the slant range resolution of the measurements, $w_1 = \frac{r^m_{s,i}}{\delta_s} - \lfloor \frac{r^m_{s,i}}{\delta_s} \rfloor$, and $w_2 = 1 - w_1$.

Removing the slant range correction step is not done to speed up the algorithm itself but rather to remove one unnecessary interpolation step. Let the map M, as earlier, be defined by its number of rows n_r and columns n_c, its origin M_0, and its resolution δ_m. Furthermore, let all the swaths used to generate a map be the swaths in the set S. Probabilistic map generation can then be performed by Algorithm 4.

In Algorithm 4, see that the flood fill gets initialized in both the starboard and the port side, as the two portions of the swath aren't connected. The initialization is done with an 8-connectivity to the initial cell to ensure that border cases don't make the algorithm fail, but during the flood fill, a 4-connectivity is used to add new cells. The probability map can be generated similarly as for Algorithm 3.

Algorithm 4 Probabilistic map generation - optimized

Input: $S, M_0, \delta_m, n_r, n_c$
Output: M

$M \leftarrow \varnothing$
$T \leftarrow \varnothing$
$V \leftarrow \varnothing$
$P \leftarrow \varnothing$

for $m \in S$ **do**
    $Q \leftarrow \varnothing$
    $Q \leftarrow Q \cup$ GetInitialCellsStb(m, M_0, δ_m)
    $Q \leftarrow Q \cup$ GetInitialCellsPort(m, M_0, δ_m)
    while $Q \neq \varnothing$ **do**
        $q \leftarrow \text{pop}(Q)$
        $T(q) \leftarrow$ GetCellBodyPos(q, m, M_0, δ_m) ▷ Get pos. of cell in body.
        if $P^m(q) > \epsilon$ **then**
            $V(q) \leftarrow V(q) + P^m(q)V^m(q)$
            $P(q) \leftarrow P(q) + P^m(q)$
            $Q \leftarrow Q \cup$ GetFourConnectedCells(q)
        end if
    end while
end for

for $r \leftarrow 1$ to n_r **do**
    for $c \leftarrow 1$ to n_c **do**
        $q \leftarrow$ GetCell(r,c)
        $M(r,c) \leftarrow P(q)^{-1}V(q)$
    end for
end for

$M \leftarrow$ KnnFill(M)

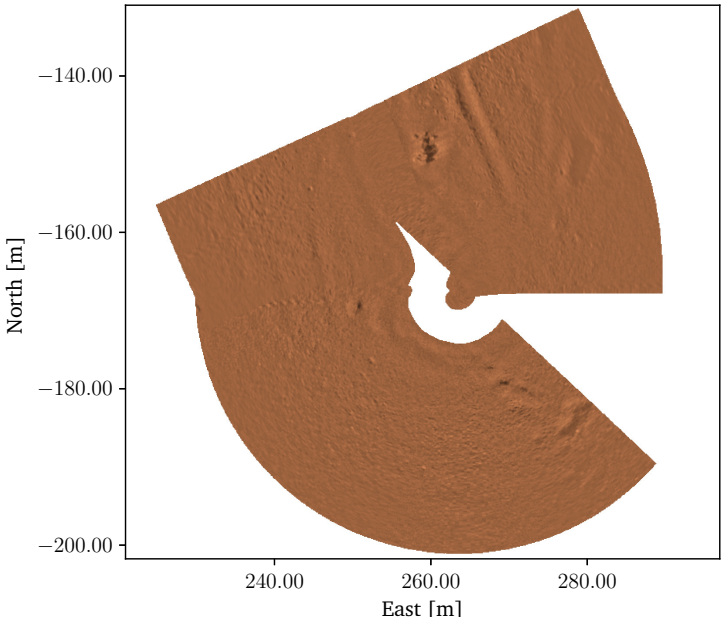
Again, we want to analyze the computational complexity of the algorithm. For the first part of the algorithm, calculating all intensities and probabilities of the swaths, we have an outer for loop that iterates swaths S times. The number of iterations on the inner while loop depends on the number of cells with a probability greater than ϵ. However, the number of cells a swath will cover is upper-bounded, and the exact upper limit depends on the sonar range, ϵ, and the map resolution δ_m. This is found to be proportional to the number of bins per swath B, such that the computational complexity of the algorithm's first part is $O(SB)$.

For the last part of the algorithm, the same arguments as for Algorithm 3 applies. The double for loop has a computational complexity of $O(CR)$, where C is the number of columns, and R is the number of rows, as both operations inside the loops can be completed in constant time. Further, the KNNFILL procedure is only performed on a small number of cells such that it is assumed that $N\log(N) < SB + RC$. The resulting computational complexity of the algorithm is then $O(SB + CR)$, a great improvement from the original algorithm.

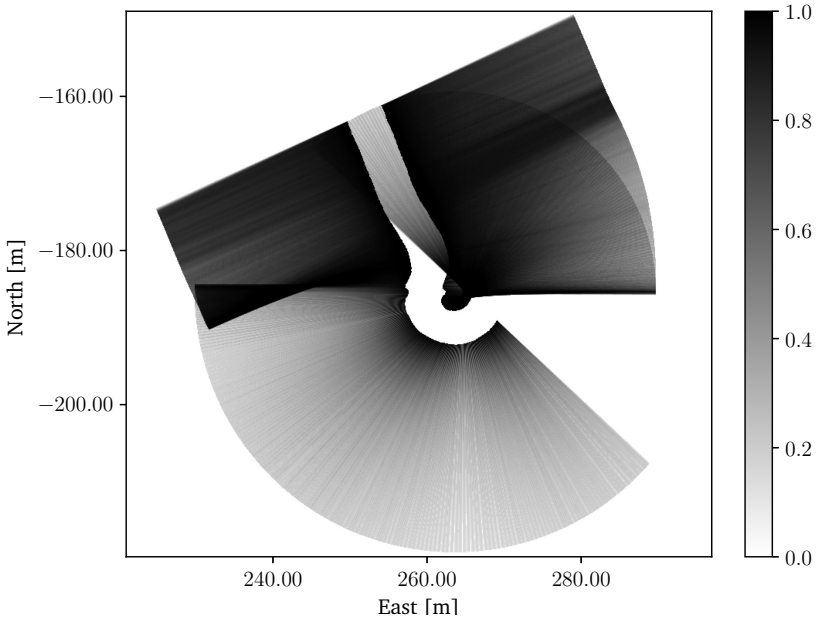
In Figure 5.9, we see the generated intensity and probability map using the probabilistic map generation algorithm with reduced computational complexity. The 500 first swaths of the training dataset were used to generate the map, together with $\delta_m = 0.1\,\text{m}$, $\epsilon = 0.1$, $k_{nn} = 2$, $r_{max} = 0.2\,\text{m}$ and $\sigma^2_{max} = 5 \cdot 10^{-3}$. Comparing the resulting intensity map in Figure 5.4a and Figure 5.7 with the intensity map in Figure 5.9a, we again find little difference.

As the number of cells in the flood fill depends on the map resolution δ_m and the probability threshold ϵ, different parameters are tested in Table 5.1 to examine how they affect the computation time. The sonar range would also affect the results, but it was not tested as it is constant for the dataset used. Only the computations of the algorithm were measured, meaning that all objects possible were preallocated, as would be the case for a real-time implementation of the algorithm. The algorithm was implemented in Julia [47], and only the intensity map was generated when the computational time was measured, not the probability map. The implementation ran on an Intel i5 CPU with Ubuntu 22. The tests were performed using the 100 first swaths of the training dataset, and each map was generated ten times, such that the presented results are the average of the ten runs. The results are presented in Table 5.1.

From Table 5.1, it is evident that increasing or decreasing the map resolution with a factor of two has a bigger impact on the computational time than doing the same for the probability threshold. It is also important to note that some combinations of parameters could not fully generate a map, displayed with "-" in Table 5.1. Examining the computation times and requiring a map resolution of at least 0.1 m, we see that a probability threshold of $\epsilon = 0.1$ gives the lowest computational time. Therefore, a probability threshold of $\epsilon = 0.1$ is used for the rest of this thesis.



(a) The intensity map.



(b) The probability map, where the color bar maps colors and probabilities.

Figure 5.9: The figure shows the intensity map and the probability map generated from the 500 first swaths of the training dataset using the probabilistic map generation algorithm with reduced computational complexity. A map resolution of $\delta_m = 0.1\,\text{m}$ and a probability threshold of $\epsilon = 0.1$ was used.

Table 5.1: Table showing the average computation times using different parameters for the probabilistic map generation algorithm with reduced computational complexity. The 100 first swaths of the training dataset are used, and for each setting, the map was generated ten times, such that the results displayed are the average over the ten runs. The missing values are due to the current parameters failing to produce a full map, leaving some or all cells out. The computation time is only measured for the computation, meaning that all objects possible were preallocated, as would be the case for a real-time implementation of the algorithm.

Prob. threshold	$\delta_m = 0.05\,\text{m}$	$\delta_m = 0.10\,\text{m}$	$\delta_m = 0.20\,\text{m}$
$\epsilon = 0.05$	353 ms	74.9 ms	23.1 ms
$\epsilon = 0.10$	–	67.1 ms	20.7 ms
$\epsilon = 0.15$	–	–	19.6 ms

Table 5.2: The table shows the computation time used for generating a map of n_s swaths from the training data set. All three algorithms presented in this thesis are tested and use a map resolution of $\delta_m = 0.1\,\text{m}$. The map was generated ten times for each number of swaths, such that the results displayed are the average computation time over the ten runs. The computation time is only measured for the computation, meaning that all objects possible were preallocated, as would be the case for a real-time implementation of the algorithm

Algorithm	$n_s = 100$	$n_s = 500$	$n_s = 1000$	$n_s = 4890$
kNN [1]	236 ms	596 ms	842 ms	2.65 s
Probabilistic [17]	3.92 s	30.9 s	91 s	615 s
Proposed probabilistic	67.1 ms	322 ms	908 ms	5.53 s

5.5 Comparison of map generation algorithms

This section will present an evaluation and comparison of the performance of the presented algorithms. Although the computational complexity can indicate the performance, it's not enough to determine if an algorithm is fast enough for an online SLAM pipeline, and we, therefore, measure the computational time. The properties of the algorithms will also be discussed in this section.

The same procedure as in Section 5.4 was used to compare computation time. All objects possible were preallocated, as would be the case if the algorithm ran in an online SLAM pipeline. The three algorithms were implemented in Julia [47], and only the intensity map was generated when the computational time was measured, not the variance or probability maps. The test was run on an Intel i5 CPU with Ubuntu 22. The tests were performed on n_s swaths from the training dataset with a map resolution of $\delta_m = 0.1\,\text{m}$. The other parameters are the same as the ones found earlier in Section 5.2, Section 5.3 and Section 5.4. The results presented are the average over ten runs and are stated in Table 5.2.

Examining the results in Table 5.2, we see, as expected, that the probabilistic map generation algorithm takes significantly longer to generate the map than the

other algorithms. Comparing the computational times between the novel probabilistic algorithm and the kNN map generation algorithm, we see that the novel probabilistic algorithm is faster up to $n_s = 500$ swaths and that the kNN map generation algorithm is faster for $n_s = 1000$ and $n_s = 4890$. Looking at the computational complexity of the algorithms, we would expect the novel probabilistic algorithm to be faster. However, as several significant constants could be embedded in the computation complexity expression, testing like this might not be directly comparable to the computational time. For example, is $\log(SB) \approx 7$ for the number of swaths $S = 4890$ and bins $B = 2000$, a constant that could be a part of the computational complexity for the novel probabilistic algorithm.

As we will see in Section 6.8, $n_s = 100$ is chosen as the map size to perform landmark detection, together with an overlapping of $n_o = 50$ swaths, generating a new map for every 50^{th} swath. To be able to perform this in an online fashion, we need a map generation algorithm that has a low enough computation time. From Table 5.2, we see that the novel probabilistic algorithm is the fastest for the chosen parameters and hence the most promising. Examining the measurement times in the dataset, we find that it takes 3.67 s to generate 50 new swaths. With a computation time of 67.1 ms, the novel probabilistic algorithm only uses only 1.83 % of the total time, leaving the rest to landmark detection, data association and inference.

Examining the resulting intensity maps in Figure 5.4a, Figure 5.7 and Figure 5.9a, it is hard to spot any difference. However, the three map generation algorithms have two important properties in common that have to be discussed.

The first property is how the algorithms, as they all rely on the state estimation, are vulnerable to drift in the state estimates. This means that generating maps from long trajectories and revisiting the same areas within the same map will likely produce poor results in the revisited areas due to the drift. However, the maps generated from 100 swaths used in the SLAM pipeline are affected to a very small extent. This is because the drift in state estimates is negligible due to the small timeframe.

The second property is how the non-homogeneous dynamic range of the swaths, as discussed in Section 4.3, propagates to the maps. This is not a property of the map generation algorithms themselves but rather a property propagated from the individual swaths. As we will see in Section 6.8, this property reduces the landmark detector performance.

This chapter has presented three cartesian map generation algorithms and compared them against each other. The novel optimized map generation algorithm is found to have the fastest computational time for the number of swaths used in the pipeline. It is, therefore, the preferred method for map generation in the SLAM pipeline. The next chapter will present how landmarks are detected in the generated maps.

Chapter 6

Landmark detection

Landmark detection is a critical part of a landmark-based SLAM pipeline, and a robust and accurate landmark detector is the main goal for landmark detection in this thesis. This chapter will first present what a landmark is, regarding landmarks in side scan sonar cartesian maps, some different techniques to detect landmarks, and lastly, the landmark detection method used in this thesis. The landmark detector used in this thesis has earlier been utilized for waterfall images but is here adapted to be used on 2D cartesian maps. In addition, several steps are taken to improve its robustness, like height estimation and height filtering of the landmarks.

6.1 What is a landmark and how to detect it?

What makes out a landmark on a cartesian map generated from side scan sonar data, and how should we choose landmarks to ensure robust navigation? This is an important question to answer before diving into the landmark detector.

In the context of sparse landmark-based (or feature-based) SLAM, landmarks are considered salient and distinct environment features that can be reliably recognized [11]. This contrasts dense spatial SLAM, which aims to build descriptors of surfaces or occupied space. With the era of machine learning, semantic landmark detection has become an active field of research [11], where a landmark is no longer just a distinct environmental feature; it can also be assigned a semantic class, improving the ability to recognize it reliably.

What characteristics define landmarks in Cartesian maps generated from side-scan sonar data? For visual SLAM, SIFT [48], SURF [49], and ORB [12] features are widely used. In [18], SIFT, among others, is tested for landmark detection in waterfall images and is found to generate a high number of false positives and false negatives. This indicates that directly applying visual SLAM techniques to side-scan sonar maps may not produce satisfactory results, necessitating the use of alternative approaches.

Examining the maps in Chapter 5, we see that the maps are mostly homogenous, except for smaller regions with increased intensity, decreased intensity,

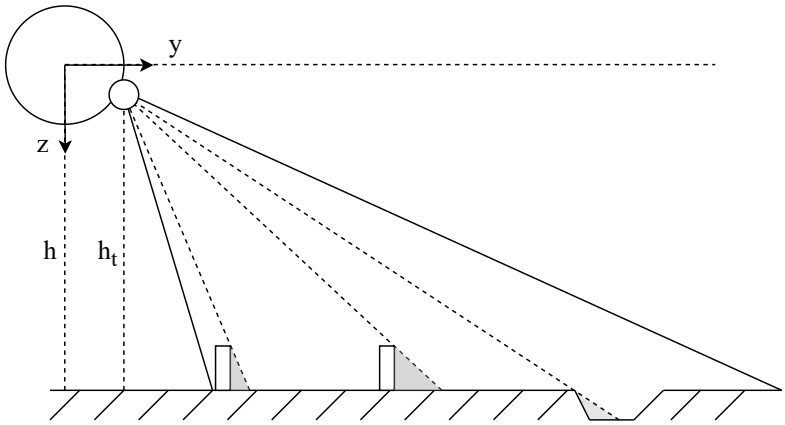


Figure 6.1: The figure shows how an elevated object and a pothole can create shadows when observed by a side scan sonar. It should also be noted how the same elevated object, observed with two different ground ranges, will give shadows of different lengths.

or both. These regions typically stem from elevated or lowered objects, such as boulders and potholes. Elevated objects are characterized by a high intensity, or echo, due to the decrease in the incidence angle of the acoustic beam, followed by an area of low intensity, a shadow. This is typically the opposite for potholes; we get a shadow followed by an echo. In addition, other inhomogeneities in the seabed can also create shadows and echoes in sonar maps, such as sand ripples, wrecks, and other objects. A pothole and two identical objects, observed by a side scan sonar, are shown in Figure 6.1.

Shadows and echoes are distinct environmental features and hence are candidates for use as landmarks. It is important to consider that objects on the seabed exhibit varying shadow lengths when observed by sonar at different ranges, as can be seen in Figure 6.1. Consequently, this creates shadows with different areas in the 2D map. In addition, their appearance on the map differs depending on the observed range. This is due to the incident angle changing together with the range. Therefore, reliably recognizing them across various ranges may prove challenging. Nevertheless, in the lack of other good alternatives, echoes, and shadows remain the most viable choice for landmarks and are therefore selected for this thesis. The next step is to find a landmark detector that can reliably detect the landmarks.

In side scan sonar images, shadows alone or in combination with echoes are widely used as landmarks. The shadows and echoes are either found by directly examining the intensity in the image [18, 40, 50, 51] or by using machine learning to detect the landmarks [43, 52, 53]. In addition to landmark position, a landmarks semantic class can be classified. In [53], machine learning is utilized to detect boulders on the seabed.

For simplicity, a landmark detector that directly examines the intensity in the map was chosen. Such landmark detectors generally consist of two steps. The first one is to use an intensity threshold, either found online or chosen offline, to filter

the sonar image to find landmark candidates. The second step is to use different techniques to filter the landmark candidates to be left with detected landmarks. The next section will present the landmark detector used in this thesis.

6.2 Landmark detection using intensity threshold, geometric filtering, and height estimation

In [18], a landmark detector for waterfall images using intensity thresholding and geometric filtering to find shadow landmarks is proposed. The method is also recreated and tested on the data used in this thesis in [2].

The first step of the method is to use an intensity threshold, chosen offline, to find landmark candidates in the sonar image. In [2], a morphological closing operation was performed on the landmark candidates found by the intensity threshold and was found to increase the method's robustness.

The second step of the landmark detector is to perform a series of geometric considerations to filter the landmark candidates. Firstly, the landmark candidates are filtered on their along-track length. Secondly, a threshold on the area of the image a landmark covers is imposed. To counteract the fact that the length of the shadow depends on the observation range, each area gets corrected by the landmark candidates' observation range. Thirdly, the landmark candidates are filtered concerning the percentage of the bounding box surrounding the landmark that is covered by the detected landmark. This is done to filter out thin landmarks or landmarks containing holes.

This thesis adapts the detector in [18] to detect landmarks in cartesian maps, adds filtering with two thresholds for improved robustness, and adds height approximation, filtering, and correction to the landmark detector for increased performance.

The proposed algorithm consists of five steps. The first step is to perform bilateral filtering, intensity thresholding with two thresholds, and morphological operators to find two sets of landmark candidates in the 2D cartesian map. The second step is to filter these two sets of landmark candidates geometrically before an intersection operation between the two sets is performed, only keeping the landmarks detected in both sets. The third step is classifying the landmark candidates as either elevated or lowered objects. The fourth step is to estimate the height of the landmark candidates and perform a final filtering based on the estimated height. The last step is to use the information available to find the position of the landmark, both in the world frame and relative to the AUV. The following sections will explain the different steps of the landmark detector in depth.

6.3 Step 1: Finding landmark candidates

The first step of the landmark detector is to find initial landmark candidates in the generated map. The map is filtered with a bilateral filter [54] to smooth out

the intensities but keep the sharp edges characterizing the boundary between the map's background and the shadows. The bilateral filter takes two parameters, σ_s for the similarity term of the filtering and σ_c for the closeness term. The reason for the filtering is that we are only interested in relatively large shapes, and filtering will remove small areas or single pixels with low intensity, reducing the number of false positives in the landmark candidates and hence the number of landmark candidates needed to process.

Intensity thresholding with two different thresholds, followed by a morphological opening and a morphological closing, is then performed on the filtered map. The intensity thresholds $T_{i,l}$ and $T_{i,h}$ are simply checked against the intensity of each cell in the map. Using a kernel size of $K \cdot K$, the morphological operations are performed on all the cells below the threshold. The result is the two initial sets of landmark candidates used in the rest of the landmark detector, where one landmark candidate is the cluster of all neighboring cells that results from the intensity threshold and morphological operators.

The idea of using two thresholds in parallel came from a duality when tuning the method with only one intensity threshold. A low threshold would yield very few false positives, but it also left out some pixels that belonged to the landmark. The opposite was the case for a higher threshold. False positives were generated, but the true positives were a larger extent detected. Combining the thresholds should give the best of both, resulting in better performance.

Figure 6.2 shows the resulting landmark candidates performing intensity thresholding and morphological operators on the 100 first swaths of the training dataset. The map is filtered by a bilateral filter using $\sigma_s = 2.0$ and $\sigma_c = 2.0$. The green landmark candidates are then generated using a high-intensity threshold of $T_{i,h} = 0.97$, and a kernel size of $K \cdot K = 3 \cdot 3$. In addition, the projected path of the AUV is displayed in grey. As pitch and roll correction is used, the path is slightly offset compared to the blind zone in the map.

6.4 Step 2: Geometric filtering of landmark candidates

The second step of the landmark detector is to perform geometric filtering on the two landmark candidate sets and combine the results. Using the landmark candidates, the geometric filtering consists of two steps, filtering the area of the landmark and the percentage of the bounding box covered by the landmark candidate.

First, the landmark candidate area gets filtered against a minimum area threshold, A_{min}, and a maximum area threshold, A_{max}. No correction of the shadow area is done, in contrast to [18], as this step is only used to prune out the smallest and largest landmark candidates and rather rely on other filtering steps for the detailed filtering of landmarks.

Second, the landmark candidates are filtered based on the percentage of the bounding box of the landmark candidate covered by the landmark. The bounding box filtering uses a threshold T_{bb}. This thesis uses a square bounding box to emphasize circular or circular-like shadows. The hypothesis is that circular shadows

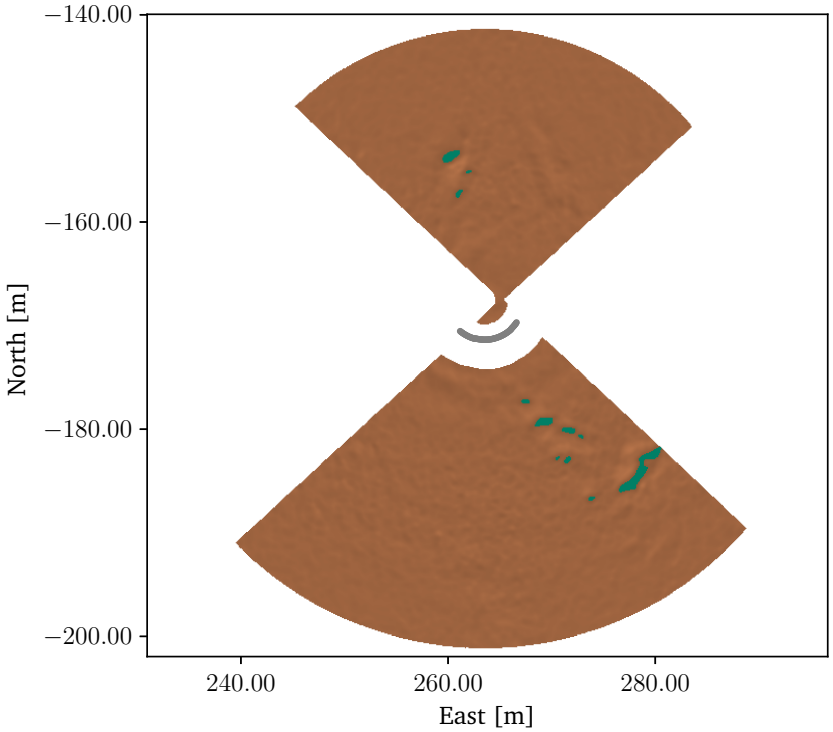


Figure 6.2: The figure shows the resulting landmark candidates after performing bilateral filtering, intensity thresholding, and morphological operators on the map generated from the 100 first swaths of the training dataset. The resulting landmark candidates are shown in green, and the path of the AUV is in grey. For the bilateral filter, $\sigma_s = 2.0$ and $\sigma_c = 2.0$ was used. The intensity threshold was performed with a high-intensity threshold of $T_{i,h} = 0.97$, and for the morphological operations, a kernel size of $K \cdot K = 3 \cdot 3$ was used.

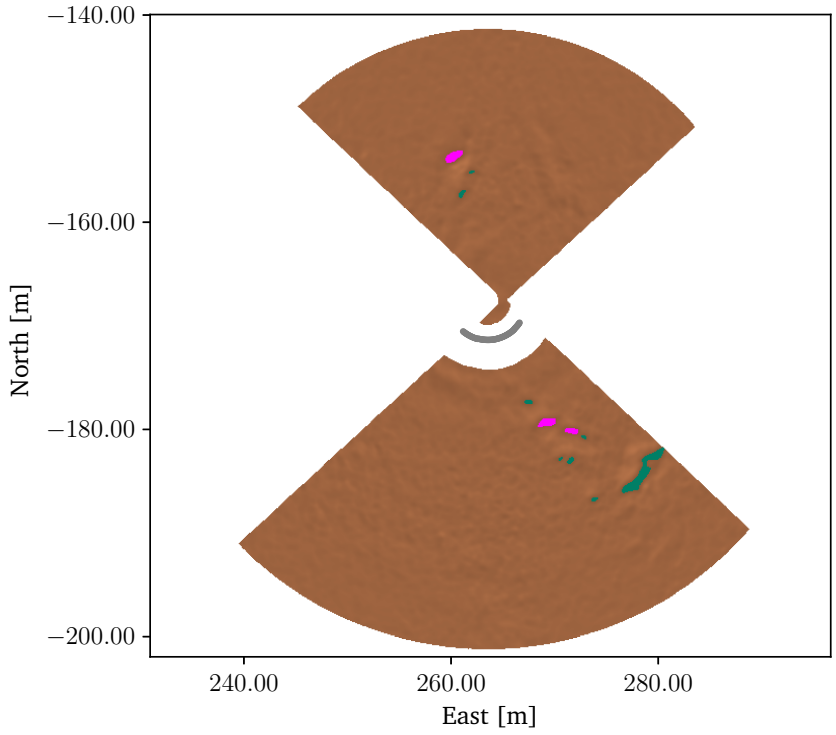


Figure 6.3: The result of performing geometric filtering on the landmark candidates in Figure 6.2. The pink and green landmark candidates make out the initial landmark candidates from Figure 6.2, where the pink is the kept landmark candidates after the geometric filtering. Note how small and thin landmarks are filtered out. For the geometric filtering, the parameters $A_{min} = 0.5\,\mathrm{m}^2$, $A_{max} = 10.0\,\mathrm{m}^2$ and $T_{bb} = 0.3$ was used.

stem from circular objects, such as boulders and potholes, and that their circular form makes the form and size of the shadow less dependent on the angle of observation. This, again, should lead to the landmarks being easier to recognize and detect reliably from different viewing angles.

Figure 6.3 shows the resulting landmarks candidates after performing geometric filtering on the landmark candidates in Figure 6.2. The parameters $A_{min} = 0.5\,\mathrm{m}^2$, $A_{max} = 10.0\,\mathrm{m}^2$ and $T_{bb} = 0.3$ was used. The pink and green landmark candidates make out the initial landmark candidates from Figure 6.2, where the pink is the kept landmark candidates after the geometric filtering.

The same thresholds and parameters are used for geometrically filtering both sets of landmark candidates, and an intersection of the two sets is performed only to keep the landmark candidates detected by both the low and high thresholds. As the high-intensity threshold is thought to be best for picking up all the pixels of the landmark, the pixels found by the high-intensity landmarks are used to represent the landmark further in the detector.

Figure 6.3 shows the result of performing the intersection of the two sets of

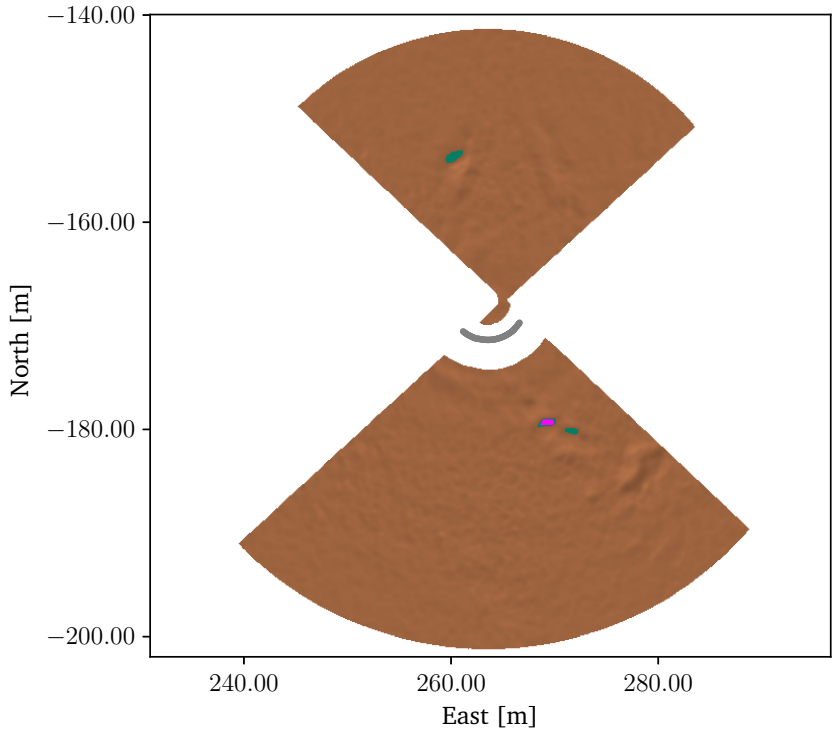


Figure 6.4: Geometric filtered landmarks from high and low-intensity thresholding showed in the same figure. The green landmark candidates are the kept landmarks of Figure 6.3, and the pink landmark candidate stems from a low-intensity threshold of $T_{i,h} = 0.95$ geometric filtered with the same parameters as the high-intensity landmark candidates. The landmark detector performs an intersection between the two sets, and here is only one landmark candidate kept.

genetically filtered landmark candidates. Again the first 100 swaths of the training dataset are used, together with a low-intensity threshold of $T_{i,h} = 0.95$, and a high-intensity threshold of $T_{i,h} = 0.97$. The green landmarks are found with the high-intensity threshold, and the pink ones with the low-intensity threshold. Both are geometric filtered with $A_{min} = 0.5\,\mathrm{m}^2$, $A_{max} = 10.0\,\mathrm{m}^2$ and $T_{bb} = 0.3$. As can be seen, only one landmark candidate has overlapping detection and is the only one kept.

6.5 Step 3: Landmark classification

The third step of the landmark detector is to perform landmark classification. Two extra steps are performed when the map is generated to do this. First, a range map is computed, where each cell contains the average ground range between the cell and the body frame of the AUV for the swaths used to generate the cell. Secondly, the swaths used to generate each cell are saved. Using this information, a minimum ground range, $r_{g_{min}}$, and a maximum ground range, $r_{g_{max}}$ of each

landmark candidate can be obtained by iterating over the cells in the landmark candidate. In addition, a set of all the swaths used to generate the cells of the landmark candidate, S_l, can be found.

As presented in Section 6.1, elevated or lowered objects are categorized by order of the echo and shadow, where an elevated object first will have an echo, followed by a shadow, and a lowered object the opposite. Here "first" points to the attribute with the lowest ground range. The landmark detector uses these properties to classify the object type. The object type is then used for height estimation of the landmark candidate. We assume a flat seafloor everywhere on the map, except for the landmark itself, such that both the seabed in front of and behind the landmark are assumed to have the same elevation.

The classification of the landmark candidates is performed by estimating the order of the echo and the shadow in each individual swath used to generate the landmark. This is performed by fitting a Gaussian derivative to the swath using non-linear least squares fitting to estimate the order of the shadow and echo. Furthermore, the estimates are combined into an ensemble score for classifying the landmark candidate.

To fit a Gaussian derivative to the swath, a portion of the swath around the landmark has to be chosen to use for the fitting. The maximum slant range, $r_{s_{max}}$, and minimum slant range $r_{s_{min}}$ of the landmark candidate are found using the minimum and maximum ground range found earlier together with $r_s = \sqrt{r_g^2 + h_t^2}$. Here, h_t is the transducer height. Let the slant length of the landmark candidate be $l_s = r_{s_{max}} - r_{s_{min}}$. We then choose to include the bins with a slant range of $[r_{s_{min}} - l_s, r_{s_{max}} + l_s]$, such that we include bins equal to the slant length of the landmark, both before and after the actual landmark. Since the individual swaths are not filtered, we filter the chosen bins with a Gaussian low-pass filter with a standard deviation of σ_f to reduce noise in the swaths.

The Gaussian derivative has the same form as an object that appears in a swath, a top followed by a low or opposite, depending on the sign. For function fitting, we use the Gaussian derivative given by

$$\frac{ac}{\sqrt{2\pi}}(x-b)\exp\left\{\frac{-(c(x-b))^2}{2}\right\}+d, \tag{6.1}$$

where a, b, c, and d are shape parameters to be determined by the non-linear least squares function fitting. We bound $c \geq 0$ such that the sign of a gives the order of the top and the bottom and hence does the sign if a becomes the estimate of the classifier.

The fitting of the Gaussian derivative is done for each swath used to generate the landmark. Figure 6.5 shows the result of fitting the Gaussian derivative in (6.1) to one of the swaths in a landmark candidate. If the function fittings fail, the swath is dropped. Each swath's results are combined to create an ensemble score for estimating the object's class. The class with the most "votes" are chosen. In the case of an equal amount of votes, it is not possible to determine the landmark class, and the landmark candidate is dropped.

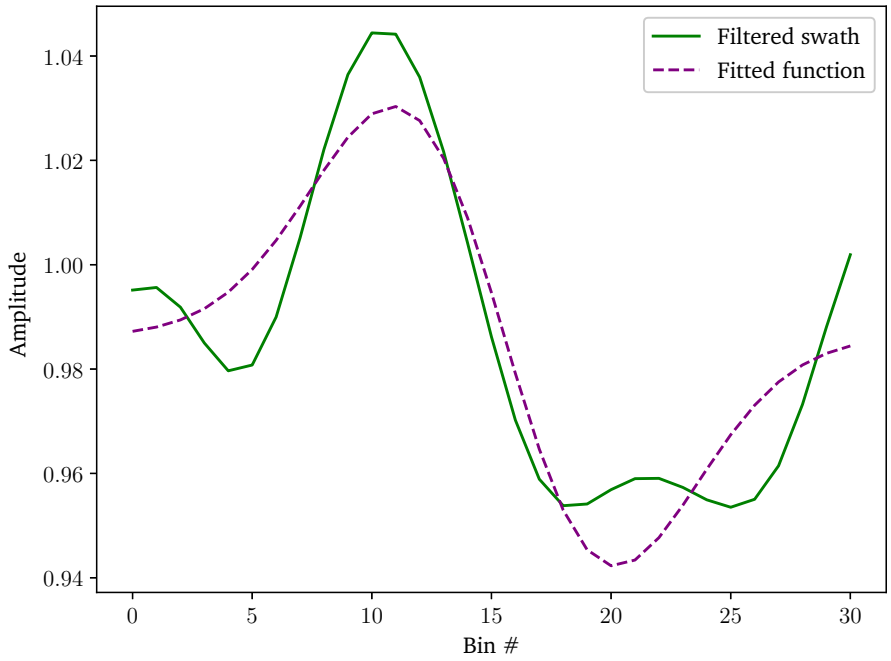


Figure 6.5: The results on fitting the function in (6.1) to a part of a filtered swath with a slant range of $[r_{s_{min}} - l_s, r_{s_{max}} + l_s]$. The low-pass filtering is performed with $\sigma_f = 2.0$. The resulting parameters from the function fitting were $a = -0.182$, $b = 15.540$, $c = 0.214$, and $d = 0.986$. Since $a \leq 0$, the results "vote" for the landmark candidate to be an elevated object.

It is important to note that the only element missing to be able to use the landmark detector in a semantic SLAM pipeline is to reason about the uncertainty of the classification. This could, for example, be done by finding the detector's confusion matrix by running the detection on several datasets and finding the probabilities for detection and misdetections. However, finding the confusion matrix requires both time and data and since especially the former was limited, finding the confusion matrix is left as further work.

6.6 Step 4: Height estimation of landmark candidates

The fourth step of the landmark detector is to perform height estimation and filtering of the landmark candidates. For this step, a "reference" pose and altitude of the AUV for each landmark candidate are needed. However, there are several candidate poses and altitudes in the set S_l. One solution would have been to average over the swaths in S_l, but since averaging over attitudes are not straightforward, a simpler solution is used in this thesis.

To find the "reference pose" for a landmark, we first assume that the AUV maintains a constant speed and heading rate for the short time the landmark is observed. Then the swaths in S_l are sorted by the time of measurement, and the reference pose and altitude are chosen as the pose and altitude of the AUV when the middle swath of S_l was acquired. The middle swath is chosen because, given the assumptions, this middle swath will be the one observing the middle of the landmark. Choosing the middle of the landmarks should lead to the same reference position being chosen when the landmark is observed from a different heading. To simplify, the first of the two middle swaths are used for an even number of swaths, but an interpolation would have been more accurate.

To estimate the height of the landmark, we use the retrieved minimum and maximum slant ranges together with the triangle similarity of the measurement geometry. The following relation holds for elevated objects

$$\frac{h_l}{h_t} = \frac{r_{s_{max}} - r_{s_{min}}}{r_{s_{max}}}. \tag{6.2}$$

where h_l is the height of the landmark. The estimated height can then be found by

$$\hat{h}_l = h_t\left(1 - \frac{r_{s_{min}}}{r_{s_{max}}}\right). \tag{6.3}$$

For lowered objects, we have the relation

$$\frac{h_l}{h_t} = \frac{r_{s_{max}} - r_{s_{min}}}{r_{s_{min}}}, \tag{6.4}$$

such that the height estimation becomes

$$\hat{h}_l = -h_t\left(\frac{r_{s_{max}}}{r_{s_{min}}} - 1\right), \tag{6.5}$$

where the sign of the landmark height estimate is changed such that the height of lowered objects becomes negative.

It is important to note that the slant range and not the ground range have to be used for estimating landmark height. To be able to create a shadow, a landmark has to violate the flat seafloor assumption locally. Hence, the landmark cells' ground range has not been correctly calculated. However, by retracing back to the original slant range, the height of the landmark can be correctly estimated.

Using the absolute of the estimated landmark height, the landmarks candidates get filtered, removing landmarks below a height threshold, T_h. The height filtering is the final filtering step, and the resulting landmarks are the detected landmarks.

6.7 Step 5: Where in the world is the landmark?

The last step of the landmark detector is to find both the position of the landmark in the world frame and relative to the AUV. In addition, for use in the SLAM pipeline, we reason about the uncertainty of the position of the landmark relative to the AUV.

First, we need to determine the reference point for each landmark. Examining Figure 6.1, the best guess for an elevated object is thought to be at the position of the start of the shadow. For a lowered object, it is thought to be at the end of the shadow. It is important to note that this is not exact, but rather the best guess with the information available. Again recognizing that an elevated object with a height different from zero violates the flat seafloor assumption, we can find the correct landmark position by recalculating the ground range using the obtained minimum and maximum slant ranges. For an elevated landmark, its ground range is

$$r_{g,l} = \sqrt{r_{s_{min}} - (h_t - \hat{h}_l)^2}, \tag{6.6}$$

and for a lowered landmark, it is

$$r_{g,l} = \sqrt{r_{s_{max}} - (h_t + \hat{h}_l)^2}. \tag{6.7}$$

Then, using (4.10), the position of the landmark in the world frame can be found.

Figure 6.6 shows the result of performing landmark detection on the map of the first 100 swaths of the training dataset. The landmark height, the landmark area, and the fill rate of the bounding box are also shown. The black "x" denotes the landmark position.

In addition to the position of the landmark in the world frame, the range and bearing of each detected landmark and the uncertainty of the range and bearing are calculated. Using the range and bearing instead of the world or body position of the landmarks has two reasons. The first one is that the nature of the sonar measurement is a range-bearing measurement. Even though the map generation does not conserve the range-bearing measurement, the landmark detector uses the raw swaths for several of the steps, and we, therefore, want to use range

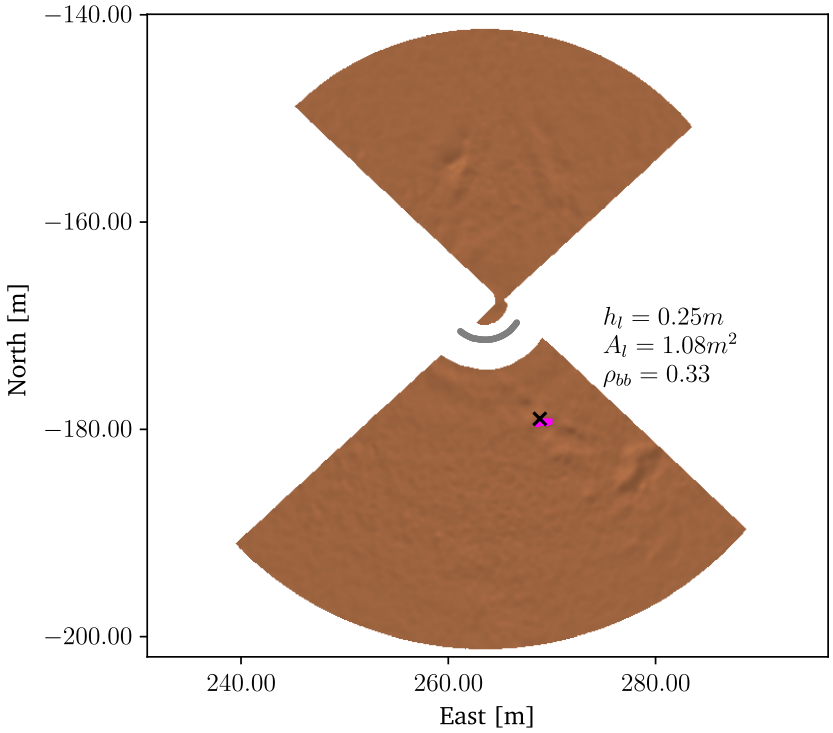


Figure 6.6: The figure shows the resulting landmark after performing landmark detection on the 100 first swaths of the training dataset. Here, the landmark candidate from Figure 6.4 is classified as an elevated landmark, and its height is estimated as $\hat{h}_l = 0.25\,\text{m}$. The height threshold was set to $T_h = 0.2\,\text{m}$, so the landmark was kept. In addition, the figure shows the landmark areal A_l and the fill rate of the bounding box ρ_{bb}. The "x" denotes the landmark position.

and bearing when reasoning about the uncertainty of the range and bearing. The second is that range-bearing measurements are used in the chosen probabilistic data association method in Chapter 7.

The range of a landmark is

$$z_r = \sqrt{(x_l - x_b)^2 + (y_l - y_b)^2}, \tag{6.8}$$

where x_l and x_b are the x-component of the landmark position and the AUV pose, respectively. The same applies to the y-components. The bearing of the landmark, relative to the heading of the AUV, is given by

$$z_b = \text{atan2}(y_l - y_b, x_l - x_b) - \psi, \tag{6.9}$$

where ψ is the AUV heading for the reference pose. Together (6.8) and (6.9) make out the landmark measurement function

$$h(\boldsymbol{x}, \boldsymbol{l}) = \begin{bmatrix} \sqrt{(x_l - x_b)^2 + (y_l - y_b)^2} \\ \text{atan2}(y_l - y_b, x_l - x_b) - \psi \end{bmatrix}, \tag{6.10}$$

where $\boldsymbol{x}$ is the AUV pose, $\boldsymbol{l}$ is the landmark position in the world frame and $\boldsymbol{z} = [z_r, z_b]^T$.

We need to consider the sonar measurements, map generation, and the landmark detector to reason about the uncertainty in the range-bearing measurement. For range-bearing measurements, it is common to use a Gaussian measurement model [19], something this thesis also does.

For the range measurement, we have uncertainty contribution from the sonar measurement, the map generation, and the landmark detector. As the slant resolution of the sonar bins is 0.03 m, the related uncertainty is assumed to be small relative to the other uncertainties and is therefore disregarded. The map generation has a resolution of $\delta_m = 0.1$ m, and we assume a standard deviation equal to the map resolution. The landmark detection has several steps, each contributing to the uncertainty. However, as it is hard to reason about the uncertainty in each step, we want to relate it to the length of the landmark. We assign each range measurement an additional standard deviation of 50 % of the total landmark length. The total standard deviation of the range measurement is then

$$\sigma_r = \sqrt{\delta_m^2 + \left(\frac{r_{g_{max}} - r_{g_{min}}}{2}\right)^2}. \tag{6.11}$$

For the bearing uncertainty, there is also a contribution from the sonar measurement, the map generation, and the landmark detector. However, the main contribution is related to the landmark detector and choosing the reference point of the landmark. We again want to relate the measurement to the map's sizes; in this case, the arc length stemming from the standard deviation of the bearing measurement and the mean range measurement is used. The uncertainty in the bearing

measurement is approximated as the bearing resulting in arclength to five times the map resolution such that the bearing standard deviation becomes

$$\sigma_b = \frac{5\delta_m}{\bar{z}_r}, \tag{6.12}$$

where $\bar{z}_r$ is the mean of the range measurement. This means that landmarks with a shorter range measurement will have a larger bearing uncertainty than landmarks with a longer range measurement. This concludes the landmark detector, and the next section will present the results from performing landmark detection on the training and test dataset.

6.8 Landmark detection on the training and test dataset

This section tests the proposed landmark detector on the training and test datasets. To use the landmark detector in a SLAM pipeline, the landmark detection is done on batches of n_b swaths where each batch corresponds to a timestep t in the SLAM pipeline. A batch of swaths is used mainly because the map generation and landmark detector only operate on batches of swaths. In addition, this also reduces the number of timesteps that have to be inferred.

By using batches of swaths, we run the risk that one landmark can be split in the boundary between two consecutive batches. To help against this, we introduce an overlap of n_o swaths between each batch of swaths. The size of each batch is still n_b swaths, but the first n_o swaths at time t are the last n_o swaths of the swath batch at time $t-1$, where the swaths are sorted according to their measurement time.

When choosing the size of n_b and n_o several factors have to be considered as they are important parameters of the landmark detector. Firstly, we want the overlap to be large enough to ensure that all landmarks are fully displayed in at least one timestep. Secondly, the size of n_b and n_o governs the time between two consecutive timesteps, and hence how large the delay for new pose estimates is. Regarding the delay, we want n_b and n_o to be as small as possible, but this has to be weighed against the extra time needed for inference. In this thesis $n_b = 100$ and $n_o = 50$ are used.

The detected landmarks of the training dataset's 2000 swaths are shown in Figure 6.7, where the parameters in Table 6.1 were used. The parameters were found by repeatedly tuning the landmark detector, examining the detected landmarks, and testing the detected landmarks in the full SLAM pipeline.

Examining Figure 6.7, several low-intensity areas with round shapes appear. These are potential landmarks that are not detected. During the tuning, it became evident that a conservative landmark detector gave the best results for the pipeline, and the potential landmarks that are not detected are, therefore, a choice made during the tuning. In addition, it became evident that the landmark detector had two main weaknesses during the tuning process.

Table 6.1: The resulting parameters after tuning the landmark detector on the training dataset.

Parameter	Value
σ_s	2.0
σ_c	2.0
$T_{i,l}$	0.95
$T_{i,h}$	0.97
K	3
A_{min}	$0.5\,\mathrm{m}^2$
A_{max}	$10.0\,\mathrm{m}^2$
T_{bb}	0.3
σ_f	2.0
T_h	0.2 m

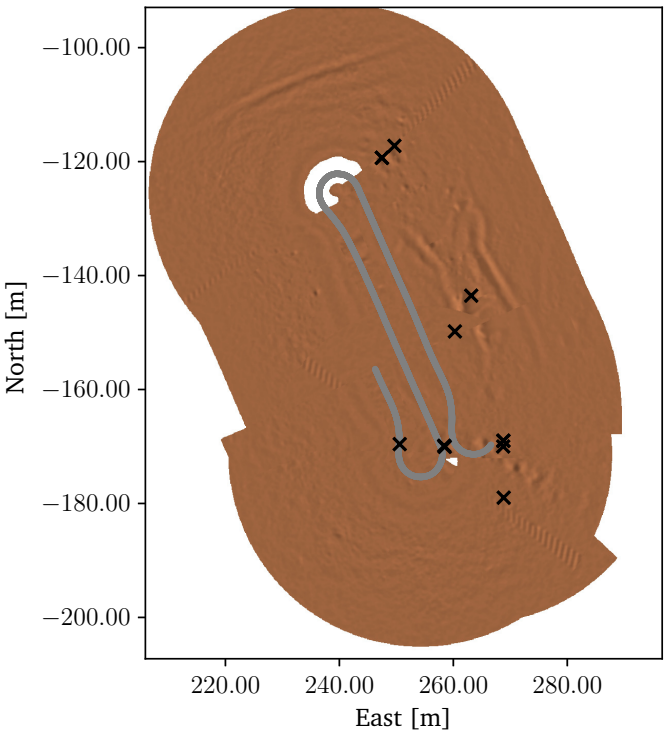


Figure 6.7: Resulting landmarks after running the proposed landmark detector on the 2000 first swaths of the training dataset using the parameters in Table 6.1. The landmarks are shown as "x" on the map, and the path of the AUV is shown in grey.

The first weakness of the landmark detector is that it lacks robustness in detecting the same landmark at different ground ranges. Two properties of the landmark detector are identified as potential reasons for this. The first property is how the size of the landmark shadows changes when the landmarks are observed from different ranges, as shown in Figure 6.1. The second property is how the dynamic range of the map differs at different ranges, as discussed in Section 5.5. The first property is handled by the filtering process and should not be the main reason for this.

The difference in the dynamic range of the map is probably the cause of the lacking re-detection of landmarks from different ground ranges. This is because different dynamic ranges imply that different intensity thresholds should be used in different places on the map. Therefore, tuning the constant intensity thresholds in the landmark detector involves a tradeoff. By increasing the thresholds, landmark candidates can be detected over the entire map, but this may result in an increased number of false positives. On the other hand, reducing the thresholds may help minimize false positives, but it may also make it challenging to detect landmarks in regions with a low dynamic threshold. The latter was chosen during the tuning as it gave the best results when performing SLAM.

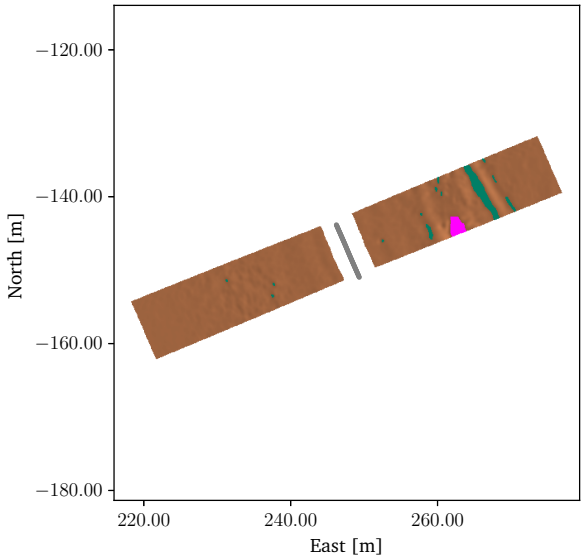
The second weakness is that the landmark detector sometimes picks up only parts of landmarks. This can happen in two different situations. The first situation occurs when a landmark is cut off due to the batching of swaths. This is the case for the pink landmark in Figure 6.8a. If the landmark candidate is filtered out, this is not a problem. However, if the landmark is kept, this can give erroneous landmark detections as both the detected size and middle position are not the landmark's true size and middle position.

The other situation where the landmark detector only partially detects landmarks is probably due to the different dynamic ranges of the map. This is shown in Figure 6.8b, where the large landmark in the middle of the right portion of the map is only partially detected. If the map is examined closely, it can be seen that the landmark has a large tail that is not picked up by the landmark detector.

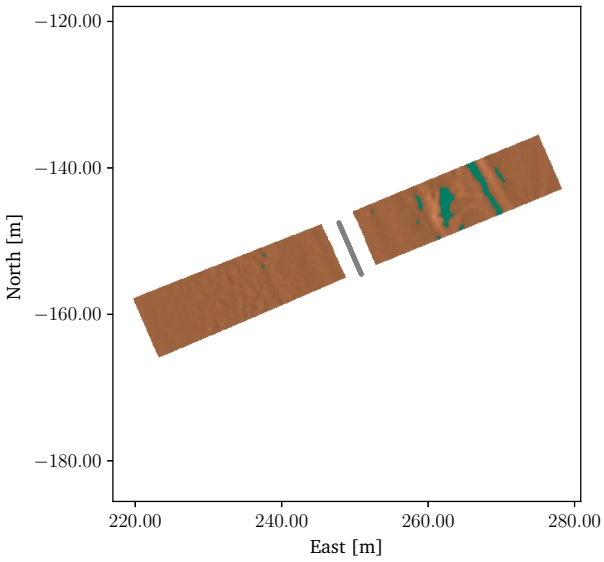
These two weaknesses should be asses and improved in further work as this would increase the performance of the landmark detector and potentially the entire SLAM pipeline. An improvement might be achieved by changing the process of generating landmark candidates in Section 6.3 to a soluting inferring on the geometry of the landmarks instead of their intensity. Machine learning could, for example, be utilized to perform this. Furthermore, a homogenous dynamic range of the map would improve the ability to re-detect and fully detect landmarks. Lastly, an online adaption of the batch size n_b and n_o would be able to solve the weaknesses stemming from performing batch-wise detection.

Figure 6.10 shows the detected landmarks when the proposed landmark detector was tested on the full test dataset. Again, the parameters in Table 6.1 were used. The low number of detected landmarks and the number of non-detected intensities suggest that the landmark detector is conservatively tuned.

This chapter has presented a novel landmark detector for 2D cartesian maps

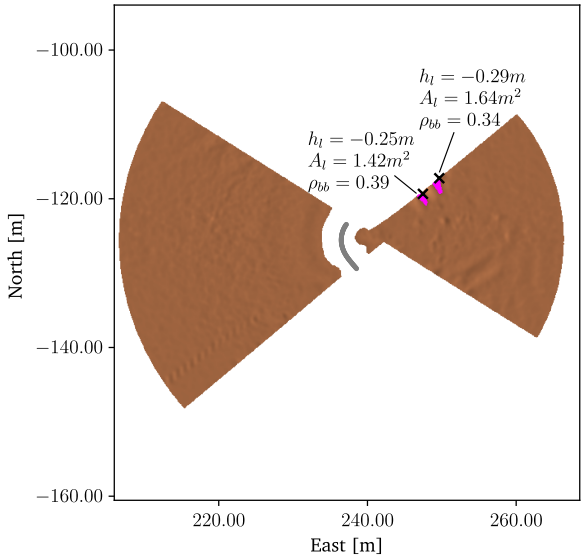


(a) The resulting landmark candidates after geometric filtering of the landmark candidates stemming from the high-intensity threshold. Here the landmark detector is performed of swaths from swath number 1200 to 1300.

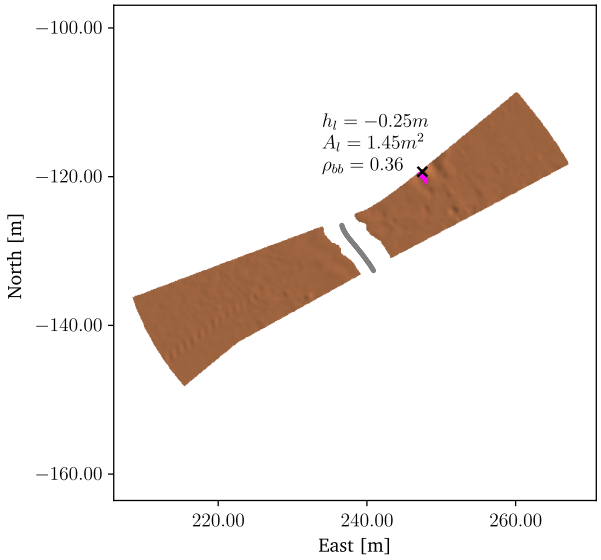


(b) The resulting landmark candidates after geometric filtering of the landmark candidates stemming from the high-intensity threshold. Here the landmark detector is performed of swaths from swath number 1250 to 1350.

Figure 6.8: The figures show the detected landmarks in two consecutive and overlapping batches of swaths. The green landmark candidates are discarded, and the pink one is kept. Not how the same landmark is detected on both plots, resulting in a double detection of the same landmark observation.



(a) The detected landmark in the batch of swaths consisting of swaths from swath number 900 to 1000.



(b) The detected landmark in the batch of swaths consisting of swaths from swath number 950 to 1050.

Figure 6.9: The figures show the result after landmark detection in two consecutive and overlapping batches of swaths. The landmark positions are displayed with an "x." The landmark height h_t, landmark areal A_l, and the percentage of the bounding box covered by the landmark ρ_{bb} are also shown. Note how only one of the landmarks detected in the top map is re-detected in the bottom map.

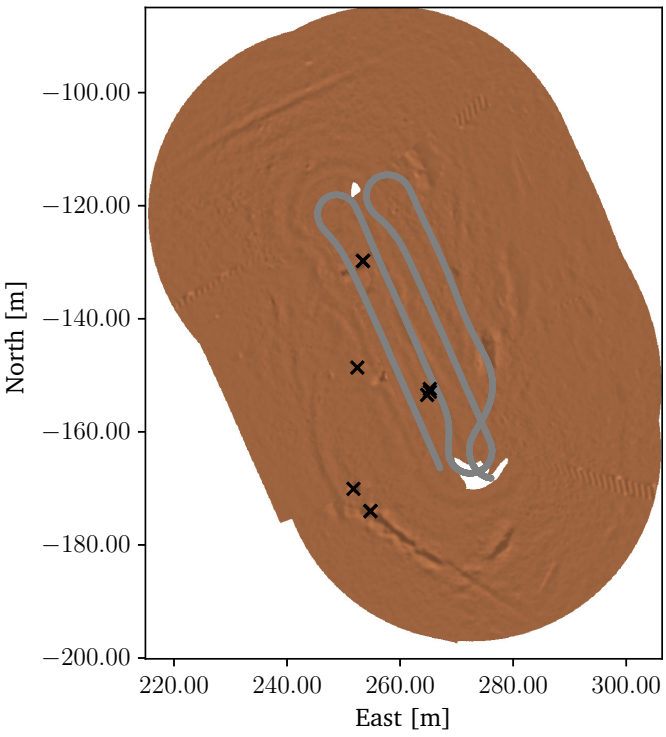


Figure 6.10: Resulting landmarks after running the proposed landmark detector on the entire test dataset using the parameters in Table 6.1. The landmarks are shown as "x" on the map, and the path of the AUV is shown in grey.

generated from side-scan sonar data and tested the landmark detector on the test and training dataset. The results show that it is able to detect landmarks, but two weaknesses were identified that are reducing its performance. The next chapter will present the method for associating the detected landmarks and a method of inference as the final step of the SLAM pipeline.

Chapter 7

Multimodal SLAM with probabilistic data assosiation

The last step of the SLAM pipeline is to perform data association on the detected landmarks and inference on the landmarks and the odometry measurements. This thesis utilizes probabilistic data association and multimodal incremental smoothing and mapping for data association and inference. This chapter will present the relevant background, the methods used, and the results of performing SLAM on underwater side scan sonar data processed by the former steps of the SLAM pipeline.

7.1 Factor graphs

Factor graphs are the backbone of modern SLAM algorithms and are a simple yet powerful graphical model that is the de-facto standard for modeling SLAM [19]. A factor graph, $\mathcal{G} \equiv \mathcal{V}, \mathcal{F}$, is a bipartite graph, where the vertices, $\mathcal{V}$, consists of factors f and variables V . The graph represents the factorization of a function and can be written as

$$f(\boldsymbol{V}) = f_1(\boldsymbol{V_1}) f_2(\boldsymbol{V_2}) ... f_n(\boldsymbol{V_n}), \tag{7.1}$$

where f_i represents a function that takes all its neighboring vertices $\boldsymbol{V_i}$. In the context of inference, we typically let the factor graph represent the unnormalized joint probability over $\boldsymbol{V}$

$$p(\boldsymbol{V}) \propto \prod_{i=1}^{n} f_i(\boldsymbol{V_i}). \tag{7.2}$$

Furthermore, in the context of SLAM, we often let the variables, $\boldsymbol{V_i}$, represent poses, $\boldsymbol{X}$, and landmarks, $\boldsymbol{L}$ and the factor's the probabilistic constraints between them. The factor graph formulation then becomes

$$p(\boldsymbol{X}, \boldsymbol{L} \mid \boldsymbol{Z}) \propto \prod_{i=1}^{n} f_i(\boldsymbol{V_i}),\ \boldsymbol{V_i} \subseteq \boldsymbol{X}, \boldsymbol{L}. \tag{7.3}$$

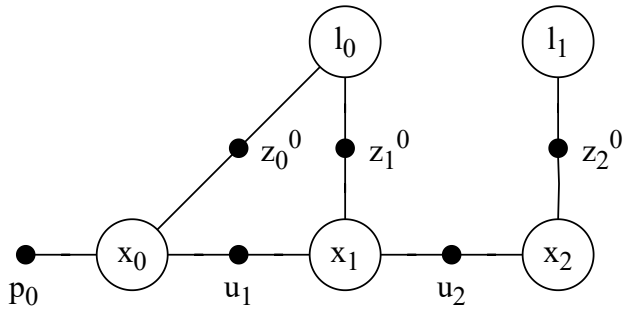


Figure 7.1: A simple factor graph consisting of three poses x_t and two landmarks l_i. p_0 is a prior factor containing prior information about the pose x_0. u_t are odometry factors generated from state estimation and z_t^i landmark measurement factors.

Figure 7.1 shows an example factor graph representing a simple slam problem with two landmarks and three poses in addition to a prior factor, odometry factors, and range-bearing factors for the landmark measurements. It is evident that the factor graph is a powerful graphic tool and that the structures in the SLAM problem are easily visible, at least in the simple example. However, as factor graphs can contain cycles, another graphical model is used for inference.

7.2 Bayes tree

As factor graphs can contain cycles, they are often refactored into a Bayes tree to make inference on them computationally tractable [55, 56]. A Bayes tree is a tree-structured graph consisting of cliques, $\boldsymbol{C_k} = \boldsymbol{F_k}, \boldsymbol{S_k}$, where $\boldsymbol{F}$ and $\boldsymbol{S}$ are subsets of $\boldsymbol{V}$ [19]. The subset $\boldsymbol{F}$ are named the "frontal" variables, and the subset $\boldsymbol{S}$ are named the "separator" variables, and the resulting factorization is

$$p(\boldsymbol{V}) \propto \prod_k p(\boldsymbol{F_k} \mid \boldsymbol{S_k}). \tag{7.4}$$

Figure 7.2 shows an example Bayes tree, where the Bayes tree is refactored from the factor graph in Figure 7.1.

Two algorithms are of great importance for marginalization on Bayes trees: the sum-product and max-product algorithms [19]. Given the observed variables, they are used for inferring the distribution of latent variables or, in other words, the marginal distribution. The algorithms will not be stated in this thesis, and the reader is referred to [19] for a presentation of the algorithms. The important takeaway from the two algorithms is their key differences. The max-product algorithm is used for maximum *a posteriori* (MAP) inference. In the case of a multimodal distribution, for example, a sum of Gaussians, will the max-product algorithm only choose the most dominant mode; hence the resulting marginal is a "max-marginal" distribution. This is a reasonable approximation in the case of

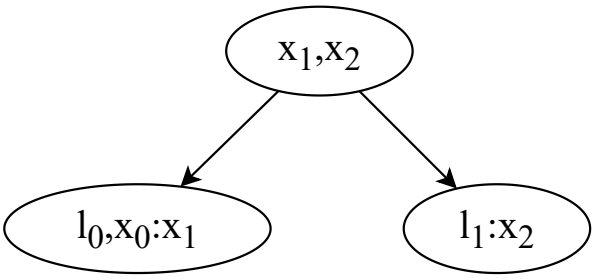


Figure 7.2: A Bayes tree generated from the example factor graph in Figure 7.1. The root clique contains only the frontal variables x_1 and x_2, whereas the leaf cliques contain both frontal and separator variables.

one clearly dominant mode. However, valuable information can get lost if the posterior distribution contains several dominant modes with small differences. Using the sum-product algorithm solves this problem by exactly recovering the marginal distribution, hence being able to keep more than one mode. As we will see later, this can be valuable for probabilistic data association and inference in the context of SLAM.

7.3 Simultaneous localization and mapping

Simultaneous localization and mapping is a procedure of acquiring a map of an unknown environment as a mobile platform explores it and, at the same time, localizes the mobile platform on the same map [57]. SLAM has been one of the key enabling technologies of mobile robot navigation and is so for mobile robot navigation in the underwater environment as well [15]. In the context of landmark-based SLAM, the map is not a 2D cartesian map of intensity, like in Chapter 5, but rather a map of landmarks and poses, together with their probability distributions.

The SLAM procedure generally performs inference in the factor graph in (7.3). The factors used in the factor graph consist of sensor measurements and (potential) priors for the variables. Firstly, we have the odometry factors, u_t, representing the evolution of the pose, or in other terms, the probability distribution $p(\boldsymbol{x_t} \mid \boldsymbol{x_{t-1}}, \boldsymbol{u_t})$. These can be obtained by motion models or sensor measurements, and this thesis calculates the odometry factors using the pose estimate stemming from the state estimation. Secondly, landmark measurement factors are factors that represent landmark sightings and are given by $p(\boldsymbol{l_i} \mid \boldsymbol{z_t^i}, \boldsymbol{x_t})$, where $\boldsymbol{z_t^i}$ is the measurement of landmark i at the time t. The measurement factors can be of different types, but the landmark measurements are treated as range-bearing factors in this thesis. We have here associated measurement $\boldsymbol{z_t^i}$ with landmark $\boldsymbol{l_i}$, and we will in the next section present how the data association is performed. Figure 7.1 shows a factor graph representing a SLAM problem.

Suppose we assume Gaussian measurement models and focus on the MAP solution of the SLAM problem. In that case, we can formulate the inference as a

non-linear least-square optimization problem [19]

$$\hat{\boldsymbol{X}}, \hat{\boldsymbol{L}} = \arg\max_{\boldsymbol{X},\boldsymbol{L}} p(\boldsymbol{X}, \boldsymbol{L} \mid \boldsymbol{Z}), \tag{7.5}$$

where we still represent $p(\boldsymbol{X}, \boldsymbol{L} \mid \boldsymbol{Z})$ as the factor graph in (7.3). Since we are assuming Gaussian measurement models, the factors in the factor graph become

$$f_i(\boldsymbol{V_i}) = \frac{1}{\sqrt{2\pi \det \Sigma_i}} \exp\left\{-\frac{1}{2}||h_i(\boldsymbol{V_i}) - \mu_i||^2_{\Sigma_i}\right\}, \tag{7.6}$$

where μ_i is the mean and Σ_i is the covariance of variable $\boldsymbol{V_i}$, and $||\cdot||^2_\Sigma$ is the squared Mahalanobis distance. Removing the normalization constant, we are left with

$$p(\boldsymbol{X}, \boldsymbol{L} \mid \boldsymbol{Z}) \propto \prod_i \exp\left\{-\frac{1}{2}||h_i(\boldsymbol{V_i}) - \mu_i||^2_{\Sigma_i}\right\}. \tag{7.7}$$

Since the goal is to recover the most probable assignment to poses and landmarks, removing the normalization constant does not affect the result, as it's not dependent on the poses or the landmarks. Using the same reasoning, we can transform the optimization problem in (7.5) by taking the log of the joint probability and inserting (7.7), resulting in

$$\arg\max_{\boldsymbol{X},\boldsymbol{L}} p(\boldsymbol{X}, \boldsymbol{L} \mid \boldsymbol{Z}), = \arg\min_{\boldsymbol{X},\boldsymbol{L}} \sum_i ||h_i(\boldsymbol{V_i}) - \mu_i||^2_{\Sigma_i}. \tag{7.8}$$

The result in (7.8) can be recognized as a nonlinear least-squares problem that can be optimized using methods such as Gauss-Newton and Levenberg-Marquadt. A widely used method SLAM with Gaussian measurement models is iSAM2 [55], which does inference using the Bayes tree and uses the general method presented above.

In the case of a non-Gaussian measurement model or other factors containing non-Gaussian distributions, other solutions must be used. One solution is to use mm-iSAM [56], allowing for a wide range of (potentially) multimodal measurement models and factors. The method uses *approximate belief propagation*, exploiting the Bayes tree structure to perform marginalization. For approximating the non-gaussian and multimodal beliefs in each of the variables, kernel density estimation is used

$$\left[\hat{X}\right] = \sum_{i=1}^{N} w^{(i)} \mathcal{N}\left(x^{(i)}, \Lambda^{(i)}\right), \tag{7.9}$$

where $\left[\hat{X}\right]$ is the estimated probability density over X, $\Lambda^{(i)}$ the bandwidth parameter, $x^{(i)}$ the center of, and $w^{(i)}$ the weighting factor for each of the N kernels.

7.4 Data association

When new landmark measurements arrive, we don't know which landmarks it stems from or if it is a measurement of a new landmark, and the process of determining this is called data association. This is an important part of the SLAM

pipeline, where one wrong association can cause an otherwise correct solution to deviate from the ground-truth solution [19]. Let d_t^k denote the data association of measurement k, at pose $\boldsymbol{x_t}$, where $d_t^k = j$ associates the measurement with landmark $\boldsymbol{l_j}$. Furthermore, let $\boldsymbol{D} \equiv \{\boldsymbol{d_t}\}_{t=1}^{T}$ be the set of all associations. The data association at each time, $\boldsymbol{d_t}$, can be incorporated in the factor graph and will appear as latent variables. The factor graph from (7.3) can now be written

$$p(\boldsymbol{X}, \boldsymbol{L}, \boldsymbol{D} \mid \boldsymbol{Z}) \propto \prod_{i=1}^{n} f_i(\boldsymbol{V_i}),\ \boldsymbol{V_i} \subseteq \boldsymbol{X}, \boldsymbol{L}, \boldsymbol{D}. \tag{7.10}$$

An important distinction must be made between *lazy* data and *proactive* data association. Lazy data association allows for revision of previous decisions, whereas proactive data association makes a data association only based on the prior information available and does so without revision. This thesis uses the latter.

As mentioned above, we don't know if a landmark measurement stems from an existing or new, unobserved landmark. A common and simple method to determine this is measurement gating [19]. The process of measurement gating uses a threshold τ to determine whether to create a new landmark. The method calculates the likelihood of a measurement belonging to a known landmark. This likelihood is calculated for each known landmark, where a new landmark is created if no likelihood falls above the threshold τ.

Maximum-likelihood (ML) data association is one of the most common solutions to the data association problem [19]. It uses initial estimates of the poses $\boldsymbol{X^{(0)}}$ and landmarks $\boldsymbol{L^{(0)}}$ and optimalization to find the ML data association. The ML data association is given by

$$\hat{\boldsymbol{D}} = \arg\max_{\boldsymbol{D}} p(\boldsymbol{D} \mid \boldsymbol{X^{(0)}}, \boldsymbol{L^{(0)}}, \boldsymbol{Z}). \tag{7.11}$$

Methods like the Hungarian algorithm [58] and joint compatibility branch and bound [59] are examples of ML data association methods.

Probabilistic data association (PDA) is an alternative solution that finds the probability of the data associations to be able to marginalize them out of the factor graph when finding the solution to the SLAM problem [19] and is given by

$$\begin{aligned} \hat{\boldsymbol{X}}, \hat{\boldsymbol{L}} &= \arg\max_{\boldsymbol{X},\boldsymbol{L}} \sum_{\boldsymbol{D}} p(\boldsymbol{X}, \boldsymbol{L}, \boldsymbol{D} \mid \boldsymbol{Z}) \\ &= \arg\max_{\boldsymbol{X},\boldsymbol{L}} \sum_{\boldsymbol{D}} p(\boldsymbol{X}, \boldsymbol{L} \mid \boldsymbol{D}, \boldsymbol{Z}) p(\boldsymbol{D} \mid \boldsymbol{Z}) \\ &= \arg\max_{\boldsymbol{X},\boldsymbol{L}} \mathbb{E}_{\boldsymbol{D}}[p(\boldsymbol{X}, \boldsymbol{L} \mid \boldsymbol{D}, \boldsymbol{Z}) \mid \boldsymbol{Z}]. \end{aligned} \tag{7.12}$$

It is important to note that the approximate marginal distribution is almost always multimodal. Specifically, in the context of Gaussian measurement models, the multimodal distribution becomes a sum of Gaussians. Since the traditional least squares method used for SLAM does not handle multimodal distributions,

we are left with two options. The first option is to use a method that can utilize the traditional least squares method, such as the method in [60].

The PDA in [60] performs expectation-maximization iteration on the sum of Gaussians to replace it with a geometric mean, preserving the Gaussian posterior assumption. The second option is to use the mm-iSAM algorithm in [56] to represent the actual multimodal distributions in the data association variables, like the works in [61]. In this thesis, the latter is used.

In [19, 61], a sum-marginalization PDA method for semantic range-bearing landmarks is presented. As the name implies, the method performs sum-marginalization of the data associations variables, resulting in multimodal measurement factors, which are solved using the state-of-the-art non-Gaussian solver mm-iSAM [56]. The method is proactive and uses measurement gating to decide whether to create a new landmark or associate a measurement with the existing landmarks.

The main concern of the PDA in [61] is how to find the multimodal semantic factors for each measurement k that links a pose $\boldsymbol{x_t}$ and all candidate landmarks l_j in a set $\mathcal{H} \subseteq 1, ..., M$ of candidate landmarks

$$f(\boldsymbol{x_t}, \boldsymbol{l_{\mathcal{H}}}) = \sum_{j \in \mathcal{H}} p(\boldsymbol{z_t^k} \mid \boldsymbol{x_t}, \boldsymbol{l_j}) p(d_t^k = j \mid \boldsymbol{Z^-}), \tag{7.13}$$

where $\boldsymbol{Z^-}$ are all measurements prior to the measurements at time t. The measurement model is assumed to be a factorized semantic range-bearing model with $p(\boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{l}) = p(z^r \mid \boldsymbol{x}, \boldsymbol{l}) p(z^b \mid \boldsymbol{x}, \boldsymbol{l}) p(z^s \mid \boldsymbol{l})$. z^r and z^b are the estimated range and bearing, respectively, from the landmark detector, and are both assumed to be Gaussian distributed. z^s are the semantic class prediction from the landmark detector, and $p(z^s \mid \boldsymbol{l})$ corresponds to the confusion matrix of the detector, assumed to be known a priori.

To determine the association probabilities, a Monte Carlo approximation is performed to find the joint-compatibility probability of the data associations [19]. Let $\mathcal{D}_t$ denote the set of all possible associations of measurements at time t to known landmarks. Also, define $\mathbb{D}_t^k(j) \equiv \{\boldsymbol{d_t} \in \mathcal{D}_t | d_t^k = j\}$, the set of all possible sets of data associations at time t in which measurement k is associated to landmark j. Then, assuming a uniform prior for data associations

$$p(d_t^k = j \mid \boldsymbol{Z^-}) \propto \sum_{\boldsymbol{d_t} \in \mathbb{D}_t^t(j)} \prod_{i=1}^{K_t} p(\boldsymbol{z_t^i} \mid \boldsymbol{d_t}, \boldsymbol{Z^-}), \tag{7.14}$$

where K_t is the number of landmark measurements at time t.

The likelihood of each measurement $\boldsymbol{z_t^k}$ given its association d_t^k is found by

marginalization

$$\begin{aligned} p(\boldsymbol{z}_t^k \mid \boldsymbol{d}_t, \boldsymbol{Z}^-) &= p(\boldsymbol{z}_t^k \mid d_t^k = j, \boldsymbol{Z}^-) \\ &= p(z_t^s \mid d_t^k = j, \boldsymbol{Z}^-) p(z_t^r, z_t^b \mid d_t^k = j, \boldsymbol{Z}^-) \\ &= \left[\sum_c p(z_t^s \mid \boldsymbol{l}_j^s = c) p(\boldsymbol{l}_j^s = c \mid \boldsymbol{Z}^-) \right] \times \ldots \\ &\ldots \left[\int \int p(z_t^r, z_t^b \mid d_t^k = j, \boldsymbol{x}_t, \boldsymbol{l}_j) p(\boldsymbol{x}_t, \boldsymbol{l}_j \mid \boldsymbol{Z}^-) d\boldsymbol{x}_t d\boldsymbol{l}_j \right], \end{aligned} \tag{7.15}$$

where c is the landmark class, and the superscript k for the measurement is dropped to make the dependence on the range, bearing, and semantic components explicit.

To simplify the calculation of the integrals, two approximations are performed. First, a Monte Carlo approximation approximates the marginalization over the pose $\boldsymbol{x}_t$. This is done by using a set of pose samples $\boldsymbol{x}_t^{[n]},\ n = 1, \ldots, N$ drawn from $p(\boldsymbol{x} \mid \boldsymbol{Z}^-)$. This results in (7.15) being approximated as

$$\begin{aligned} p(\boldsymbol{z}_t^k \mid \boldsymbol{d}_t, \boldsymbol{Z}^-) \approx &\left[\sum_c p(z_t^s \mid \boldsymbol{l}_j^s = c) p(\boldsymbol{l}_j^s = c \mid \boldsymbol{Z}^-) \right] \times \ldots \\ &\ldots \left[\frac{1}{N} \sum_{n=1}^{N} \int p(z_t^r, z_t^b \mid d_t^k = j, \boldsymbol{x}_t^{[n]}, \boldsymbol{l}_j) p(\boldsymbol{x}_t^{[n]} \mid \boldsymbol{Z}^-) p(\boldsymbol{l}_j \mid \boldsymbol{Z}^-) d\boldsymbol{l}_j \right]. \end{aligned} \tag{7.16}$$

Secondly, the Gaussian measurement model is approximated with an ML estimate, i.e., $p(\boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{l})$ are approximated as $\delta(h(\boldsymbol{x}, \boldsymbol{l}); \boldsymbol{z})$, where $\delta(\cdot)$ is the Dirac delta function equal to one only where $h(\boldsymbol{x}, \boldsymbol{l}) = \boldsymbol{z}$ and zero elsewhere. The integral over the landmark is then simplified to a single term, and (7.16) then becomes

$$\begin{aligned} p(\boldsymbol{z}_t^k \mid \boldsymbol{d}_t, \boldsymbol{Z}^-) \approx &\left[\sum_c p(z_t^s \mid \boldsymbol{l}_j^s = c) p(\boldsymbol{l}_j^s = c \mid \boldsymbol{Z}^-) \right] \times \ldots \\ &\ldots \left[\frac{1}{N} \sum_{n=1}^{N} p(\boldsymbol{x}_t^{[n]} \mid \boldsymbol{Z}^-) p(\boldsymbol{l}_j = \hat{\boldsymbol{l}}_j^{[n]} \mid \boldsymbol{Z}^-) \right], \end{aligned} \tag{7.17}$$

where still $d_t^k = j$ and $\hat{\boldsymbol{l}}_j^{[n]} = \boldsymbol{x}_t^{[n]} + h^{-1}(\boldsymbol{z}_t^k)$. $h^{-1}(\cdot)$ is the inverse measurement function, converting from range bearing to cartesian coordinates, and is used for finding the "implied" landmark position $\hat{\boldsymbol{l}}_j^{[n]}$.

The latter part of (7.17) is used for both measurement gating and to decide the landmark candidates in the set $\mathcal{H}$. In other words, if the likelihood

$$\frac{1}{N} \sum_{n=1}^{N} p(\boldsymbol{x}_t^{[n]} \mid \boldsymbol{Z}^-) p(\boldsymbol{l}_j = \hat{\boldsymbol{l}}_j^{[n]} \mid \boldsymbol{Z}^-) \tag{7.18}$$

is above the measurement gating threshold τ, the landmark is added to the set of landmark candidates $\mathcal{H}$. If, after iterating through all the known landmarks, $\mathcal{H}$ is empty, a new landmark is created.

7.5 Multimodal SLAM with probabilistic data association

This thesis implements multimodal SLAM with PDA, utilizing landmark detections from the proposed landmark detector in Chapter 6, the state estimates in the dataset, a sum-marginalization PDA [19, 61], and mm-iSAM [56] for inference. Since no confusion matrix is found for the proposed landmark detector in Chapter 6, we here use only one landmark class in which all detected landmarks belong. This effectively removes all terms related to the semantic class of the landmark since all related probabilities become equal to one.

The choice of using PDA comes from examining the results from the landmark detector and realizing that since the landmark detector is not able to detect the same landmark observed from different ranges reliably, the data association has to handle missing re-detections. Since the PDA presented in Section 7.4 does not make deterministic associations but can incorporate several possible associations in one multimodal factor, it is chosen for data association. Since the PDA yields multimodal factors, mm-iSAM [56] is used for inference.

The SLAM algorithm process consists of five steps performed iteratively for each timestep. Firstly, the new landmark measurements are found using the landmark detector in Chapter 6. Secondly, a new pose variable and odometry factor are added to the factor graph for the current timestep. Thirdly, measurement gating is performed to decide whether or not to create a new landmark. Fourthly, probabilistic data association is performed if no landmark is to be created and a new measurement factor is added. However, if a new landmark is to be added, the fourth step creates a new landmark variable, and its measurement is added as a factor. Lastly, in step number five, the inference is performed on the factor graph. The following paragraphs will describe the different steps in depth.

The first step is to perform landmark detection on a batch of n_b swaths, utilizing the landmark detector presented in Chapter 6 to generate new landmark measurements. Each new batch of swaths is registered as a new timestep t in the factor graph, and a reference pose is needed to do so. We here choose the pose of the last swath in the batch as the reference pose for the timestep. This way, no landmark measurements concerning the reference pose are performed "forward" in time. Using this reference pose for the timestep, the range-baring measurements and their uncertainty have to be transformed. Transforming the mean of the measurement is trivial using (6.10). However, transforming the uncertainty is not straightforward. Instead of performing any transformation of the uncertainty, it is left unchanged. This is far from optimal, but as it is an open problem, a more accurate solution should be considered in future work.

The second step is to add the new pose variable and odometry factor for each new timestep, i.e., we want to find $p(\boldsymbol{x_t} \mid \boldsymbol{x_{t-1}})$. In the case of pure inertial measurement unit (IMU) measurements, this distribution can be found by preintegration of the IMU measurements [62]. However, as the AUV is equipped with an additional DVL, we instead want to use the estimated pose from the state estimation. First, let $\Delta x_t = x_t - x_{t-1}$ be the change in x-coordinate (and likewise for y).

Since both $x_t \sim \mathcal{N}(\mu_t, \sigma_t^2)$ and $x_{t-1} \sim \mathcal{N}(\mu_{t-1}, \sigma_{t-1}^2)$ are Gaussian distributed, Δx_t is also Gaussian [63], with mean

$$\mu_\Delta = \mu_t - \mu_{t-1}, \tag{7.19}$$

and variance

$$\sigma_\Delta^2 = \sigma_t^2 + \sigma_{t-1}^2 - 2\sigma_{t,t-1}, \tag{7.20}$$

where $\sigma_{t,t-1}$ is the covariance between x_{t-1} and x_t. This covariance is unknown, and in lack of a better option, we assume it to be $\sigma_{t,t-1} = \sigma_{t-1}^2$, such that $\sigma_\Delta^2 = \sigma_t^2 - \sigma_{t-1}^2$. It is important to note that this is, at best, a coarse approximation. Because the data in this thesis is missing the covariance and the variance of the estimate of both the x- and y-coordinates are increasing linearly with time, this still gives a positive and hence a valid variance. For the heading, the variance in the data is constant, and the same type of estimate will not give a valid variance. A coarse overestimate is to let the variance of the odometry heading ψ_Δ be equal to the variance of ψ_t.

The third and fourth step is to perform measurement gating and create a new landmark or associate the measurements with the existing landmarks. Both are performed the same way as in [19] and described in Section 7.4, using only one landmark class. (7.18) is used for the measurement gating. Since we only have one landmark class, (7.17) reduces to

$$p(\boldsymbol{z}_t^k \mid \boldsymbol{d}_t, \boldsymbol{Z}^-) \approx \frac{1}{N} \sum_{n=1}^{N} p(\boldsymbol{x}_t^{[n]} \mid \boldsymbol{Z}^-) p(\boldsymbol{l}_j = \hat{\boldsymbol{l}}_j^{[n]} \mid \boldsymbol{Z}^-), \tag{7.21}$$

the same as (7.18), and is used for calculating the association probabilities.

The last step is to perform inference on the factor graph, and as in [19, 61], the state-of-the-art multimodal inference algorithm mm-iSAM [56] is used.

7.6 SLAM on the training and test dataset

The PDA and SLAM algorithm presented above was tested on the test dataset and parts of the training dataset, where the main goal was to achieve loop closure for at least one of the landmarks. This was achieved for both the training and test dataset.

The results from the first 2000 swaths of the training dataset are shown in Figure 7.3, where the blue dots represent poses, and the red dots the landmarks. The dark blue rings surrounding the poses and landmarks (several are small and therefore hidden by the dots) are the level plots of their probability distribution, where the color bar on the right denotes the value. The gray path represents the original poses from the dataset at each timestep and shows how the SLAM solution differs from the state estimate in the dataset. It is not possible to assess the performance without ground truth, and the most important information from the plot is to see that the solution is finite and that the solution appears probable, given that it deviates some from state estimates in the dataset.

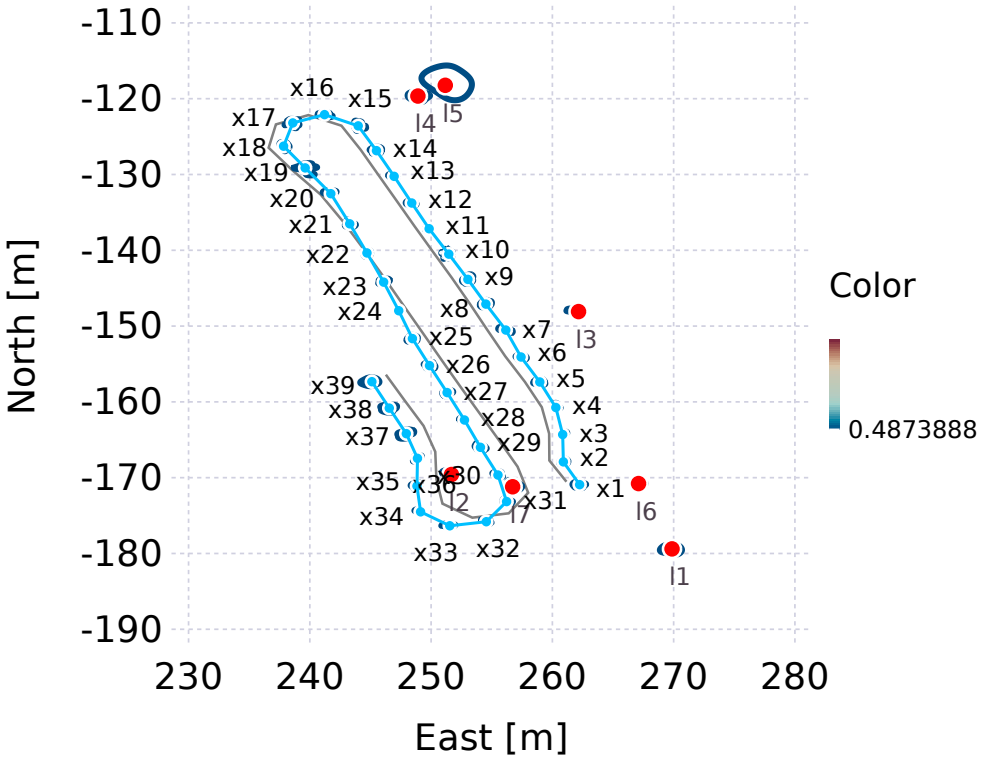


Figure 7.3: The plot shows the result performing PDA and inference on the first 2000 swaths of the training dataset. A measurement gating threshold of $\tau = 1 \cdot 10^{-7}$ was used. The blue dots represent poses, and the red dots the landmarks. The dark blue rings surrounding the poses and landmarks (several are small and therefore hidden by the dots) are the level plots of their probability distribution, where the color bar on the right denotes the value. The gray path represents the original poses from the dataset at each timestep.

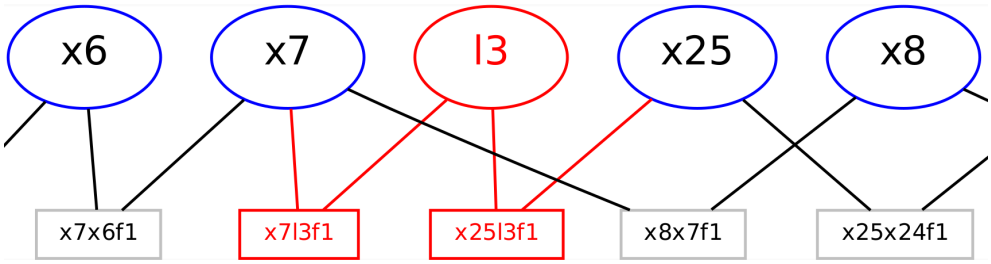


Figure 7.4: Parts of the factor graph from performing PDA and inference on the first 2000 swaths of the training dataset. The ellipses represent variables in the factor graph, and the squares represent the factors. Note how $\boldsymbol{l_2}$ are connected through factors to two poses.

Examining the results in Figure 7.3, a sudden deviation from the original path can be seen around timestep 25. In Figure 7.4, a part of the resulting factor graph is shown, and as we can see, a loop closure is happening at timestep 25. A measurement gating threshold of $\tau = 1 \cdot 10^{-7}$ was used. It was found by examining the detected landmarks from the landmark detector and realizing that it was highly likely that the landmark detected at timestep 7 and the one detected at timestep 25 were the same landmark. Using this information, the associated likelihood calculated by (7.18) was found, and the threshold set slightly lower.

Further examining the factor graph resulting from PDA and inference on the first 2000 swaths of the training dataset, an example of how the PDA can assign multimodal factors can be found. This is shown in Figure 7.5. We can see that both landmark $\boldsymbol{l_4}$ and $\boldsymbol{l_5}$ are detected at timestep 19, but only one of them at timestep 20. The results from landmark detection at timestep 19 and 20 be found in Figure 6.9. This is an example of how the PDA is capable of solving a missing landmark detection from a non-robust landmark detector. This is done by, instead of choosing to associate the new measurement at timestep 20 to one of the existing landmarks, the PDA creates a multimodal factor containing two modes. One mode represents the likelihood of the measurement stemming from $\boldsymbol{l_4}$, and the other represents the likelihood of the measurement stemming from $\boldsymbol{l_5}$.

Figure 7.5 also shows how the double detection discussed in Section 6.8 are being transferred to the factor graph and hence the SLAM solution. This double detection will add information about the pose and should lead to a more certain pose estimate. It is important to note that the extra information is wrongfully added because it stems from the same observation, detected two times by the landmark detector. However, examining the barely visible level curve around $\boldsymbol{x_{20}}$ in Figure 7.3, it can be observed that its uncertainty is larger than for the former and latter poses. This probably stems from the data association not being sure about which landmark to associate the landmark with, and the situation would probably be as expected if there was only one landmark and one measurement to associate. This wrongful adding of information is, however, not thought to be a problem of the SLAM algorithm but rather of the landmark detector.

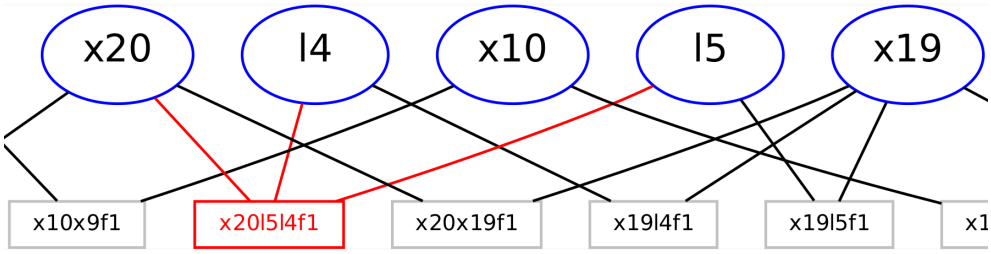


Figure 7.5: Parts of the factor graph from performing PDA and inference on the first 2000 swaths of the training dataset. The ellipses represent variables and the squares factors in the factor graph. Note how the factor in red is a multimodal factor, connecting the pose $\boldsymbol{x_{20}}$ to both landmark $\boldsymbol{l_4}$ and landmark $\boldsymbol{l_5}$.

To test if it was possible to achieve the same results using an unknown dataset, PDA and inference were performed on the entire test dataset. The results are shown in Figure 7.6. Since no ground truth is available, assessing the performance is not possible. However, the solution appears probable, given the state estimates in the dataset. The most interesting is investigating whether a loop closure has been made. By examining the resulting factor graph, a loop closure was found for the last possible timestep, shown in Figure 7.7. It also shows how the double detection of landmarks again is wrongfully added as the factors connecting the landmark to both $\boldsymbol{x_{30}}$ and $\boldsymbol{x_{31}}$. However, as achieving a loop closure was the main goal of the thesis, the results are deemed a success.

This chapter has presented the PDA and the SLAM algorithm used, and, as it is the last step, it concludes the pipeline. The results show that the pipeline is capable of performing loop closures, both in the training and test dataset. Furthermore, the results also show that the lack of robustness in the landmark detector is brought on to the inference, for example, by the double detection of landmarks. The next chapter will asses the full pipeline, discuss its strengths and weaknesses, and propose further work.

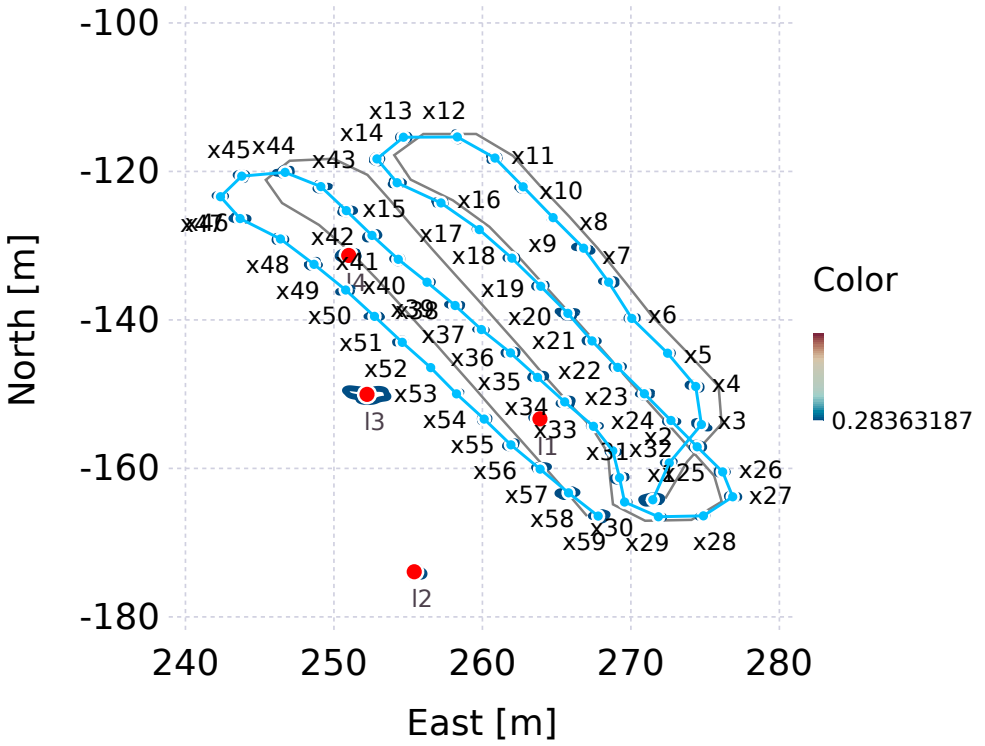


Figure 7.6: The plot shows the result performing PDA and inference on the entire test dataset. A measurement gating threshold of $\tau = 1 \cdot 10^{-7}$ was used. The blue dots represent poses, and the red dots the landmarks. The dark blue rings surrounding the poses and landmarks (several of them are small and therefore hidden by the dots) are the level plots of their probability distribution, where the color bar on the right denotes the value. The gray path represents the original poses from the dataset at each timestep.

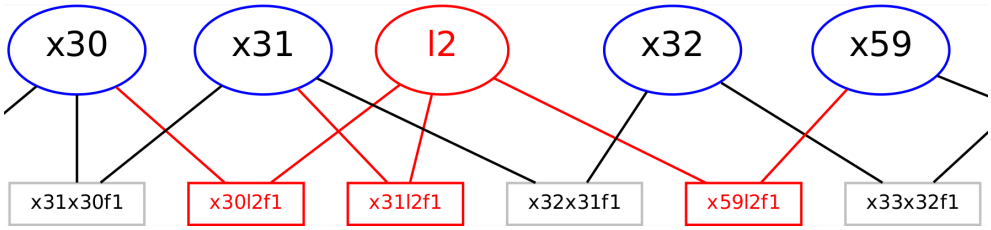


Figure 7.7: Parts of the factor graph from performing PDA and inference on the test dataset. The ellipses represent variables in the factor graph, and the squares represent the factors. Note how $\boldsymbol{l}_2$ are connected through factors to three poses.

Chapter 8

Review of the SLAM pipeline

This chapter will review the SLAM pipeline proposed in this thesis, examining the contributions from the different parts of the pipeline from a broader perspective. It will present how batches of swaths start as raw sonar measurements and end up as two landmark measurements, achieving a loop closure in the SLAM algorithm. The main strengths and weaknesses of the pipeline will also be discussed. The final section will draw from this and present the suggested further work for the pipeline.

8.1 From swaths to loop closure

This section will examine the loop closure in the training dataset, also discussed in Section 7.6, from raw swaths to inference. The loop closure happened between timestep 7 and 25 and concerns the swaths with swath numbers 300 to 400 and 1200 to 1300 in the training dataset. Figure 8.1 shows the intermediate steps of the SLAM pipeline for timestep 7 and Figure 8.2 the intermediate steps for timestep 25. We will below present the intermediate processing steps the batches of swaths undergo, from swath processing to inference and loop closure, discussing the weaknesses and strengths of the different steps as we go.

Swath processing

The first step of the SLAM pipeline is to perform swath processing on the batches of swaths, as presented in Chapter 3. It is important to note that these steps are not performed batch-wise but swath-wise as soon as a new swath arrives.

In Figure 4.3, the results show that the calculated first possible backscatter is not coinciding with the measured first backscatter. This is not a ground truth, as the calculated first possible backscatter is based on the estimated altitude. However, it points in the direction that something is wrong. Since it, unfortunately, was not possible to find the reasons for the mismatch, the errors were accepted, and the methods were used in the SLAM pipeline.

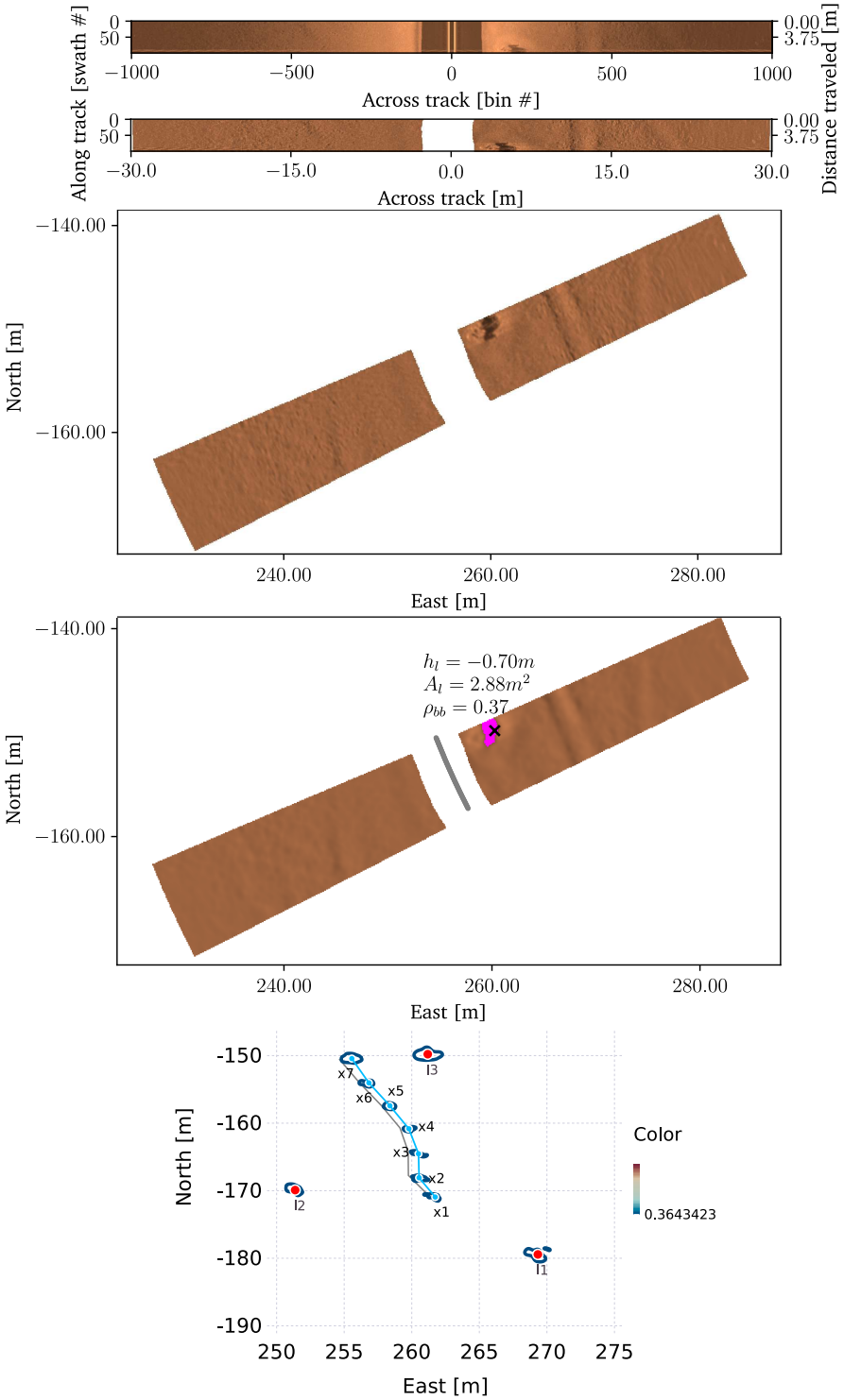


Figure 8.1: The pipeline from raw swaths with swath number 300 to 400 of the training dataset at the top to inference for timestep 7 at the bottom.

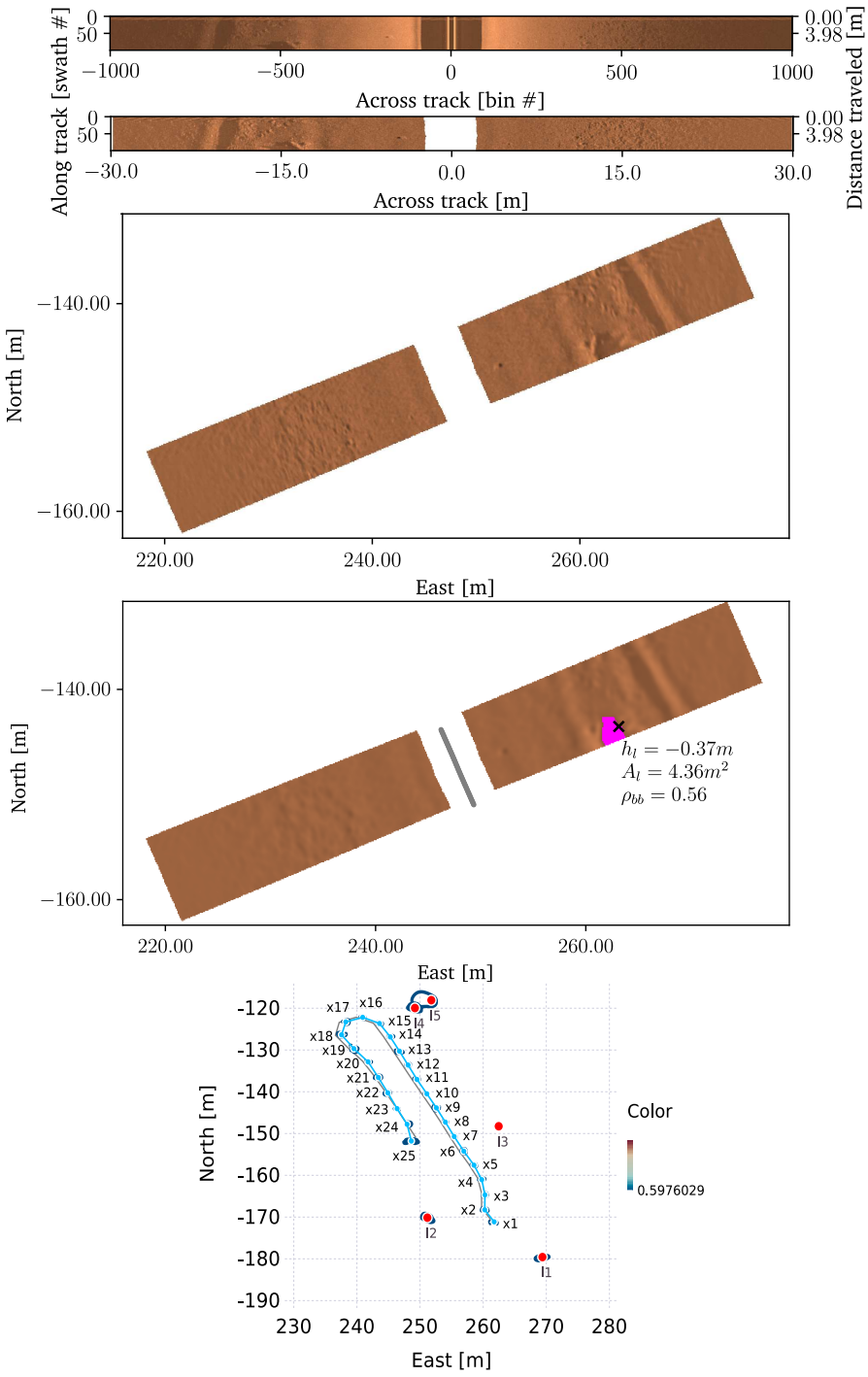


Figure 8.2: The pipeline from raw swaths with swath number 1200 to 1300 of the training dataset at the top to inference for timestep 25 at the bottom.

In the context of the full SLAM pipeline, assessing how the errors found in the swath processing might affect other parts of the pipeline is important. If either the estimated altitude or the slant range of the bins is wrong, the correctness of the map generation is affected because both are used for generating the map. This, again, will affect the performance of the entire SLAM pipeline as the correctness of the landmark detector relies on a correct map. However, as there is no ground truth, it is unfortunately not possible to assess the pipeline's performance without testing in a controlled environment.

Another aspect pointed out in Section 4.3 about the swath processing is how the intensity normalization produces swaths with a dynamic range that changes with the slant range of the bins. This artifact is brought onto the map generation and further to the landmark detector, making the detection using intensity thresholds more difficult.

Map generation

In the second step of the SLAM pipeline, the batches of swaths undergo map generation with the optimized probabilistic map generation algorithm presented in Section 5.4. The algorithm shows good performance in terms of computation time, where a map from a batch of 100 swaths with a map resolution of 0.1 m, like the maps in Figure 8.1 and Figure 8.2, is generated in 67.1 ms.

One aspect to note with the map generation used in the context of the pipeline is that, because of the overlap between batches, it performs the calculations twice for a large portion of the swaths. A proposal to minimize extra computations can be found by examining Algorithm 4 and realizing that the for-loop in line 5 to 18 only needs to be performed once per swath. This portion of the algorithm can then be performed on a smaller batch of swaths, for example, 50 swaths. To generate the final map, results from several batches can be combined by performing lines 19 to 25. This way, the first portion of the algorithm would only be performed once per swath, reducing the total computation time.

Landmark detection

The third step of the SLAM pipeline is to perform landmark detection on the generated maps, as presented in Chapter 6. The first thing to notice in the Figure 8.1 and Figure 8.2 is how the estimated landmark height $\hat{h}_l$, landmark areal A_l and fill rate of the bounding box ρ_{bb} differs, even though it is found to be the same landmark by the PDA. The landmark areal A_l and the fill rate of the bounding box ρ_{bb} should differ, as the landmark is observed by different ranges and, therefore, will produce a different-sized shadow. In addition, it can be observed that both landmarks have cells at the border of the map, and it is reasonable to assume that parts of the landmark are cut off by the batching of swaths for both detections.

The height difference is assumed to stem from the landmarks being cut off. The landmark at timestep 25 is already shown on an earlier stage in the landmark detection in Figure 6.8 and discussed in Section 6.8. It was found that it was

cut off and that the shadow, in reality, had a long tail that was not picked up by the landmark detection in timestep 26. Since the height of the full landmark has not been estimated, the actual difference in the height estimation is unavailable. However, as the extra tail would add to the depth of the landmark, this would be closer than the estimates in Figure 8.1 and Figure 8.2.

The cutoff of the landmarks is again an example of one of the weaknesses of the landmark detector, as discussed in Section 6.8. One simple solution to remove the cutoff of landmarks would be to discard all landmark candidates with a pixel at the map's border. However, this leads to the second important problem: how landmarks are sometimes only partially detected. Discarding the landmark at time step 25 means we would have to rely on the landmark being detected in the next time step. But, as Figure 6.8b shows, only some parts of the landmark are picked up, and the landmark is filtered out by geometric filtering. Unfortunately, this weakness has no obvious and simple solution and should be revised in further work.

Probabilistic data association and inference

The final step of the pipeline is to perform PDA and inference on the detected landmarks, as presented in Section 7.5. As no ground truth is available, it is not possible to assess the accuracy in Figure 8.1 and Figure 8.2.

One important aspect to note is how only the pose at timestep 25 seems to deviate from the trajectory of the dataset. One reason for this could be the course approximations of the probability of the odometry factors. Especially the heading uncertainty is thought to be an overestimate of the true uncertainty and could explain how only the pose at timestep 25 seems to be affected by the loop closure. There was little time left to revise this part of the SLAM algorithm, and the coarse approximation of odometry uncertainty should be a topic for further work.

The range bearing measurement uncertainty is also approximated by not transforming it from the pose found to observe the middle of the landmark to the reference pose, here chosen as the last pose in the batch of swaths. This is an important part of the SLAM algorithm and should, therefore, also be revised in further work.

Computation time and summary

Except for the map generation algorithm, the pipeline has not been implemented or tested for computation time, and it has not been a part of the discussion in this thesis. However, it is important to note that the current implementation cannot process the swaths and perform inference in real time. The steps in the pipeline up to inference is possible to streamline and implement more efficiently. The inference, however, takes a substantial amount of time, and the mm-iSAM algorithm may not be mature enough to function in a real-time application, something also pointed out in [19]. One option would be to change the inference algorithm to the iSAM2 [55], which has a more mature implementation but would remove the possibility of using multimodal factors in the factor graph.

To sum up, this section has examined the loop closure happening in the training dataset and discussed the SLAM pipeline's different strengths and weaknesses. The swath processing is experiencing unexpected results in the calculated and measured first backscatter return that affect its correctness and potentially affect the correctness of the entire pipeline. In addition, the intensity normalization produces swaths with a dynamic range that varies with the slant range of the bins, which is bought into map generation and landmark detection. The map generation shows good results, but overlapping batches lead to performing computation on most swaths twice. Furthermore, the landmark detector is able to detect landmarks but shows two main weaknesses, both affecting the two timesteps. Lastly, the coarse approximations made to calculate the odometry and measurement uncertainty are discussed, together with how the latter is thought to influence how the loop closure affects the trajectory. Regardless, the PDA and inference result in a loop closure, fulfilling the thesis goal.

8.2 Further work

Three main parts are pointed out as further work, as these are considered the most important ones to revise for increasing performance, correctness, and accuracy. In addition, other minor points of improvement of the thesis are discussed as well.

The first main part that should be assessed in further work is the blind-zone removal and discrepancy between the measured and calculated first backscatter. It is important for the correctness of the swath processing, but it could also affect the correctness of the entire pipeline. The results should be verified in a controlled environment, and potential solutions should be searched.

The second part is the landmark detector and the weaknesses found in it. The landmark detector could be further revised with the rest of the pipeline to improve on these, or the detector could be replaced with a different solution, for example utilizing machine learning. For the former, discarding landmarks with pixels at the map's border and performing steps to achieve a homogeneous dynamic range in the swaths, and following the map, would be a natural first step to perform.

The last part is to revise the uncertainty approximation performed in the SLAM algorithm, as these are an integral part of the final solution found by the inference algorithm. For the odometry uncertainty, performing preintegration of the IMU measurements or finding better approximations using the uncertainty from the state estimation could be performed. In addition, adding the semantic class and the height estimate of the landmark in the data association and the SLAM algorithm could also be investigated.

To conclude, the main three points to improve on the work in this thesis are the correctness of the blind zone removal, the robustness of the landmark detector, and the accuracy of calculating the measurement and odometry uncertainty. This concludes the review of the SLAM pipeline.

Chapter 9

Conclusion

This thesis successfully implemented and evaluated an underwater multimodal SLAM pipeline for side scan sonars using real-world data, thereby establishing a groundwork for future advancements in underwater SLAM for side scan sonar.

The thesis made contributions by enhancing swath processing techniques to incorporate roll and pitch correction. A novel approach was developed for generating 2D cartesian probabilistic maps, achieving notable computational efficiency (for example, achieving computational demands of only 67.1 ms for creating a map from 100 swaths at a resolution of 0.1 m). Additionally, a novel landmark detector was proposed, employing intensity thresholding, geometric filtering, and height estimation and filtering techniques. While the landmark detector successfully identified landmarks, it exhibited certain weaknesses (for example, the full extent of landmarks are not always detected). The detected landmarks were associated using probabilistic data association, and the inference was performed using the state-of-the-art mm-iSAM algorithm. The evaluations carried on in the thesis suggest moreover that the underwater multimodal SLAM pipeline is able to perform loop closures in side-scan sonar data as intended.

Nevertheless, the landmark detector's weaknesses constitute a key challenge for the SLAM pipeline and suggest the need for further investigation and improvements in future work. Additionally, the accuracy of blind-zone correction should be verified in a controlled environment due to unexpected results encountered during the evaluation. Lastly, it is recommended to re-evaluate the course approximations made in the SLAM algorithm for odometry and measurement uncertainty to ensure more accurate results.

By addressing these aspects, future research can build upon this thesis's achievements and advance the underwater SLAM field for side scan sonar, ultimately contributing to improved underwater navigation capabilities for AUVs.

Bibliography

[1] B. Reitan Hogstad, 'Side-Scan Sonar Imaging and Error-State Kalman Filter Aiding Unmanned Underwater Vehicle (UUV) to Autonomy,' M.S. thesis, 2022. [Online]. Available: https://ntnuopen.ntnu.no/ntnu-xmlui/handle/11250/3032962.

[2] V. Haraldstad, 'Comparison of two classical landmark detectors for side scan sonar,' Trondheim, 2022.

[3] M. Fava, *How much of the Ocean have we explored to date*, Visited on: 2023-16-05, May 2022. [Online]. Available: https://oceanliteracy.unesco.org/ocean-exploration/.

[4] 'Ocean's Future to 2050,' DNV, Tech. Rep., Visited on: 2023-16-05. [Online]. Available: https://www.dnv.com/oceansfuture.

[5] *Havbasert fiskeoppdrett - SalMar ASA*, Visited on: 2023-16-05. [Online]. Available: https://www.salmar.no/havbasert-fiskeoppdrett-en-ny-aera/.

[6] *Fiskeoppdrett*, Visited on: 2023-16-05. [Online]. Available: https://miljostatus.miljodirektoratet.no/tema/hav-og-kyst/fiskeoppdrett/.

[7] *Hywind Tampen - Equinor*, Visited on: 2023-16-05. [Online]. Available: https://www.equinor.com/energy/hywind-tampen.

[8] R. Bogue, 'Underwater robots: a review of technologies and applications,' *Industrial Robot: An International Journal*, vol. 42, no. 3, pp. 186–191, May 2015, ISSN: 0143-991X. DOI: 10.1108/IR-01-2015-0010. [Online]. Available: https://www.emerald.com/insight/content/doi/10.1108/IR-01-2015-0010/full/html.

[9] J. W. Nicholson and A. J. Healey, 'The Present State of Autonomous Underwater Vehicle (AUV) Applications and Technologies,' *Marine Technology Society Journal*, vol. 42, no. 1, pp. 44–51, Mar. 2008, ISSN: 0025-3324. DOI: 10.4031/002533208786861272. [Online]. Available: https://www.ingentaconnect.com/content/10.4031/002533208786861272.

[10] T. Haugstad, 'Den kan gå minst 20 mil på batteri og jobbe på 6.000 meters dyp i 6 måneder,' *Teknisk ukeblad*, Visited on: 2023-16-05. [Online]. Available: https://www.tu.no/artikler/den-kan-ga-minst-20-mil-pa-batteri-og-jobbe-pa-6-000-meters-dyp-i-6-maneder/473674.

[11] D. M. Rosen, K. J. Doherty, A. Terán Espinoza and J. J. Leonard, 'Advances in Inference and Representation for Simultaneous Localization and Mapping,' *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 4, no. 1, pp. 215–242, May 2021, ISSN: 2573-5144. DOI: 10.1146/annurev-control-072720-082553. [Online]. Available: https://www.annualreviews.org/doi/10.1146/annurev-control-072720-082553.

[12] R. Mur-Artal, J. M. M. Montiel and J. D. Tardos, 'ORB-SLAM: A Versatile and Accurate Monocular SLAM System,' *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, Oct. 2015, ISSN: 1552-3098. DOI: 10.1109/TRO.2015.2463671. [Online]. Available: https://ieeexplore.ieee.org/document/7219438/.

[13] J. Zhang, Y. Jiang and K. Wang, 'A modified FastSLAM for an autonomous mobile robot,' in *2016 IEEE International Conference on Mechatronics and Automation*, IEEE, Aug. 2016, pp. 1755–1759, ISBN: 978-1-5090-2396-7. DOI: 10.1109/ICMA.2016.7558829. [Online]. Available: http://ieeexplore.ieee.org/document/7558829/.

[14] J. Engel, T. Schöps and D. Cremers, 'LSD-SLAM: Large-Scale Direct Monocular SLAM,' in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, PART 2, vol. 8690 LNCS, Springer Verlag, 2014, pp. 834–849. DOI: 10.1007/978-3-319-10605-2{_}54. [Online]. Available: http://link.springer.com/10.1007/978-3-319-10605-2_54.

[15] F. Hidalgo and T. Braunl, 'Review of underwater SLAM techniques,' in *2015 6th International Conference on Automation, Robotics and Applications (ICARA)*, IEEE, Feb. 2015, pp. 306–311, ISBN: 978-1-4799-6466-6. DOI: 10.1109/ICARA.2015.7081165. [Online]. Available: http://ieeexplore.ieee.org/document/7081165/.

[16] L. Paull, G. Huang, M. Seto and J. J. Leonard, 'Communication-constrained multi-AUV cooperative SLAM,' in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2015-June, IEEE, May 2015, pp. 509–516, ISBN: 978-1-4799-6923-4. DOI: 10.1109/ICRA.2015.7139227. [Online]. Available: http://ieeexplore.ieee.org/document/7139227/.

[17] A. Burguera and G. Oliver, 'High-Resolution Underwater Mapping Using Side-Scan Sonar,' *PLOS ONE*, vol. 11, no. 1, J. F. Valentine, Ed., e0146396, Jan. 2016, ISSN: 1932-6203. DOI: 10.1371/journal.pone.0146396. [Online]. Available: https://dx.plos.org/10.1371/journal.pone.0146396.

[18] I. Leblond, S. Tauvry and M. Pinto, 'Sonar image registration for swarm AUVs navigation: Results from SWARMs project,' *Journal of Computational Science*, vol. 36, p. 101 021, Sep. 2019, ISSN: 18777503. DOI: 10.1016/j.jocs.2019.07.008. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1877750318307786.

[19] K. Doherty, 'Robust non-Gaussian semantic simultaneous localization and mapping,' Ph.D. dissertation, Massachusetts Institute of Technology, 2019. [Online]. Available: https://dspace.mit.edu/handle/1721.1/124176.

[20] A. Haude and C. Spancer, *Histoire de L'Académie Royale des Sciences et des Belles Lettres de Berlin avec les Mémoires pour la même année, tirez des registres de cette Académie*. chez Ambroise Haude, 1746. [Online]. Available: https://books.google.no/books?hl=no&lr=&id=zjioSoiunU0C&oi=fnd&pg=PA1&dq=related:g8ZX-n-C9MMJ:scholar.google.com/&ots=tKt1UkI9mI&sig=3HWmaF6d8wAG5xaNHtN6hIXkfEk&redir_esc=y#v=onepage&q&f=false.

[21] L. Bjørnø, T. Neighbors and D. Bradley, *Applied underwater acoustics*. Elsevier, 2017, ISBN: 978-0-12-811240-3.

[22] P. Blondel, *The Handbook of Sidescan Sonar*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, ISBN: 978-3-540-42641-7. DOI: 10.1007/978-3-540-49886-5. [Online]. Available: http://link.springer.com/10.1007/978-3-540-49886-5.

[23] H. Azhari, J. A. Kennedy, N. Weiss and L. Volokh, *From Signals to Image*. Cham: Springer International Publishing, 2020, ISBN: 978-3-030-35325-4. DOI: 10.1007/978-3-030-35326-1. [Online]. Available: http://link.springer.com/10.1007/978-3-030-35326-1.

[24] G. Canepa, A. Munafo, M. Micheli, L. Morlando and S. Murphy, 'Real-time continuous active sonar processing,' in *OCEANS 2015 - Genova*, IEEE, May 2015, pp. 1–6, ISBN: 978-1-4799-8736-8. DOI: 10.1109/OCEANS-Genova.2015.7271658. [Online]. Available: http://ieeexplore.ieee.org/document/7271658/.

[25] D. W. Ricker, *Echo Signal Processing*. Boston, MA: Springer US, 2003, ISBN: 978-1-4613-5016-3. DOI: 10.1007/978-1-4615-0312-5. [Online]. Available: http://link.springer.com/10.1007/978-1-4615-0312-5.

[26] A. Burguera and G. Oliver, 'Intensity correction of Side-Scan Sonar images,' in *Proceedings of the 2014 IEEE Emerging Technology and Factory Automation (ETFA)*, IEEE, Sep. 2014, pp. 1–4, ISBN: 978-1-4799-4845-1. DOI: 10.1109/ETFA.2014.7005092. [Online]. Available: http://ieeexplore.ieee.org/document/7005092/.

[27] M. F. Fallon, M. Kaess, H. Johannsson and J. J. Leonard, 'Efficient AUV navigation fusing acoustic ranging and side-scan sonar,' in *2011 IEEE International Conference on Robotics and Automation*, IEEE, May 2011, pp. 2398–2405, ISBN: 978-1-61284-386-5. DOI: 10.1109/ICRA.2011.5980302. [Online]. Available: http://ieeexplore.ieee.org/document/5980302/.

[28] Y.-S. Shin, Y. Lee, H.-T. Choi and A. Kim, 'Bundle adjustment from sonar images and SLAM application for seafloor mapping,' in *OCEANS 2015 - MTS/IEEE Washington*, IEEE, Oct. 2015, pp. 1–6, ISBN: 978-0-9339-5743-5. DOI: 10.23919/OCEANS.2015.7401963. [Online]. Available: http://ieeexplore.ieee.org/document/7401963/.

[29] *LAUV Fridtjof - NTNU*, Visited on: 2023-16-05. [Online]. Available: https://www.ntnu.edu/aur-lab/lauv-fridtjof.

[30] U. Långström, personal communication, Date: 2023-13-04.

[31] E. Brekke, *Fundamentals of Sensor Fusion Target tracking, navigation and SLAM*, Third edition. Trondheim, 2020, Visited on: 2023-11-05. [Online]. Available: https://folk.ntnu.no/edmundfo/msc2019-2020/sf13chapters.pdf.

[32] J. Braga, A. J. Healey and J. Sousa, 'Navigation Scheme for the LSTS SEACON Vehicles: Theory and Application,' *IFAC Proceedings Volumes*, vol. 45, no. 5, pp. 69–75, Jan. 2012, ISSN: 14746670. DOI: 10.3182/20120410-3-PT-4028.00013. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1474667016305821.

[33] T. D. Barfoot, *State Estimation for Robotics*. Cambridge University Press, Jul. 2017, ISBN: 9781107159396. DOI: 10.1017/9781316671528. [Online]. Available: https://www.cambridge.org/core/product/identifier/9781316671528/type/book.

[34] T. I. Fossen, *Handbook of marine craft hydrodynamics and motion control*, Second Edition. Wiley, 2021, ISBN: 978-1-119-57503-0.

[35] J. Solà, J. Deray and D. Atchuthan, 'A micro Lie theory for state estimation in robotics,' Dec. 2018. DOI: 10.48550/arXiv.1812.01537. [Online]. Available: http://arxiv.org/abs/1812.01537.

[36] S. Dyer and J. Dyer, 'Cubic-spline interpolation. 1,' *IEEE Instrumentation & Measurement Magazine*, vol. 4, no. 1, pp. 44–46, Mar. 2001, ISSN: 10946969. DOI: 10.1109/5289.911175. [Online]. Available: http://ieeexplore.ieee.org/document/911175/.

[37] K. Shoemake, 'Animating rotation with quaternion curves,' in *Proceedings of the 12th annual conference on Computer graphics and interactive techniques - SIGGRAPH '85*, New York, New York, USA: ACM Press, 1985, pp. 245–254, ISBN: 0897911660. DOI: 10.1145/325334.325242. [Online]. Available: http://portal.acm.org/citation.cfm?doid=325334.325242.

[38] M. Al-Rawi, F. Elmgren, M. Frasheri, B. Curuklu, X. Yuan, J.-F. Martinez, J. Bastos, J. Rodriguez and M. Pinto, 'Algorithms for the detection of first bottom returns and objects in the water column in sidescan sonar images,' in *OCEANS 2017 - Aberdeen*, vol. 2017-October, IEEE, Jun. 2017, pp. 1–5, ISBN: 978-1-5090-5278-3. DOI: 10.1109/OCEANSE.2017.8084587. [Online]. Available: http://ieeexplore.ieee.org/document/8084587/.

[39] X. Ye, H. Yang, C. Li, Y. Jia and P. Li, 'A Gray Scale Correction Method for Side-Scan Sonar Images Based on Retinex,' *Remote Sensing*, vol. 11, no. 11, p. 1281, May 2019, ISSN: 2072-4292. DOI: 10.3390/rs11111281. [Online]. Available: https://www.mdpi.com/2072-4292/11/11/1281.

[40] X. Yuan, J.-F. Martínez, M. Eckert and L. López-Santidrián, 'An Improved Otsu Threshold Segmentation Method for Underwater Simultaneous Localization and Mapping-Based Navigation,' *Sensors*, vol. 16, no. 7, p. 1148, Jul. 2016, ISSN: 1424-8220. DOI: 10.3390/s16071148. [Online]. Available: http://www.mdpi.com/1424-8220/16/7/1148.

[41] A. Galdran, A. Isasi, M. Al-Rawi, J. Rodriguez, J. Bastos, F. Elmgren and M. Pinto, 'An efficient non-uniformity correction technique for side-scan sonar imagery,' in *OCEANS 2017 - Aberdeen*, vol. 2017-October, IEEE, Jun. 2017, pp. 1–6, ISBN: 978-1-5090-5278-3. DOI: 10.1109/OCEANSE.2017.8084577. [Online]. Available: http://ieeexplore.ieee.org/document/8084577/.

[42] X. Ye, X. Ge and H. Yang, 'A Gray Scale Correction Method for Side-Scan Sonar Images Based on GAN,' in *Global Oceans 2020: Singapore – U.S. Gulf Coast*, IEEE, Oct. 2020, pp. 1–5, ISBN: 978-1-7281-5446-6. DOI: 10.1109/IEEECONF38699.2020.9389397. [Online]. Available: https://ieeexplore.ieee.org/document/9389397/.

[43] D. Einsidler, M. Dhanak and P.-P. Beaujean, 'A Deep Learning Approach to Target Recognition in Side-Scan Sonar Imagery,' in *OCEANS 2018 MTS/IEEE Charleston*, IEEE, Oct. 2018, pp. 1–4, ISBN: 978-1-5386-4814-8. DOI: 10.1109/OCEANS.2018.8604879. [Online]. Available: https://ieeexplore.ieee.org/document/8604879/.

[44] M. Moszyński, K. Bikonis and Z. Łubniewski, 'Reconstruction of 3D shape from sidescan sonar images using shape from shading technique,' *Hydroacoustics*, no. Vol. 16, pp. 181–188, 2013, ISSN: 1642-1817. [Online]. Available: https://www.infona.pl//resource/bwmeta1.element.baztech-eb5be2ea-dc33-47d8-b806-98e3c178b4c9.

[45] E. Coiras, Y. Petillot and D. M. Lane, 'Multiresolution 3-D reconstruction from side-scan sonar images,' *IEEE Transactions on Image Processing*, vol. 16, no. 2, pp. 382–390, Feb. 2007, ISSN: 10577149. DOI: 10.1109/TIP.2006.888337.

[46] J. L. Bentley, 'K-d trees for semidynamic point sets,' in *Proceedings of the sixth annual symposium on Computational geometry - SCG '90*, New York, New York, USA: ACM Press, 1990, pp. 187–197, ISBN: 0897913620. DOI: 10.1145/98524.98564. [Online]. Available: http://portal.acm.org/citation.cfm?doid=98524.98564.

[47] J. Bezanson, S. Karpinski, V. B. Shah and A. Edelman, 'Julia: A Fast Dynamic Language for Technical Computing,' Sep. 2012. DOI: 10.48550/arXiv.1209.5145. [Online]. Available: http://arxiv.org/abs/1209.5145.

[48] D. G. Lowe, 'Distinctive Image Features from Scale-Invariant Keypoints,' *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, Nov. 2004, ISSN: 0920-5691. DOI: 10.1023/B:VISI.0000029664.99615.94. [Online]. Available: http://link.springer.com/10.1023/B:VISI.0000029664.99615.94.

[49] H. Bay, A. Ess, T. Tuytelaars and L. Van Gool, 'Speeded-Up Robust Features (SURF),' *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, Jun. 2008, ISSN: 10773142. DOI: 10.1016/j.cviu.2007.09.014. [Online]. Available: https://linkinghub.elsevier.com/retrieve/pii/S1077314207001555.

[50] M. Al-Rawi, A. Galdran, F. Elmgren, J. Rodriguez, J. Bastos and M. Pinto, 'Landmark detection from sidescan sonar images,' in *2017 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT)*, vol. 2018-January, IEEE, Oct. 2017, pp. 1–6, ISBN: 978-1-5090-5969-0. DOI: 10.1109/AEECT.2017.8257760. [Online]. Available: http://ieeexplore.ieee.org/document/8257760/.

[51] J. Aulinas, X. Llado, J. Salvi and Y. R. Petillot, 'Feature based slam using side-scan salient objects,' in *OCEANS 2010 MTS/IEEE SEATTLE*, IEEE, Sep. 2010, pp. 1–8, ISBN: 978-1-4244-4332-1. DOI: 10.1109/OCEANS.2010.5664461. [Online]. Available: http://ieeexplore.ieee.org/document/5664461/.

[52] H. Wang, N. Gao, Y. Xiao and Y. Tang, 'Image feature extraction based on improved FCN for UUV side-scan sonar,' *Marine Geophysical Research*, vol. 41, no. 4, p. 18, Dec. 2020, ISSN: 0025-3235. DOI: 10.1007/s11001-020-09417-7. [Online]. Available: https://link.springer.com/10.1007/s11001-020-09417-7.

[53] J. H. Christensen, L. V. Mogensen and O. Ravn, 'Side-Scan Sonar Imaging: Automatic Boulder Identification,' in *OCEANS 2021: San Diego – Porto*, vol. 2021-September, IEEE, Sep. 2021, pp. 1–6, ISBN: 978-0-692-93559-0. DOI: 10.23919/OCEANS44145.2021.9705713. [Online]. Available: https://ieeexplore.ieee.org/document/9705713/.

[54] C. Tomasi and R. Manduchi, 'Bilateral filtering for gray and color images,' in *Sixth International Conference on Computer Vision (IEEE Cat. No.98CH36271)*, Narosa Publishing House, 1998, pp. 839–846, ISBN: 81-7319-221-9. DOI: 10.1109/ICCV.1998.710815. [Online]. Available: http://ieeexplore.ieee.org/document/710815/.

[55] M. Kaess, H. Johannsson, R. Roberts, V. Ila, J. Leonard and F. Dellaert, 'iSAM2: Incremental smoothing and mapping with fluid relinearization and incremental variable reordering,' in *2011 IEEE International Conference on Robotics and Automation*, IEEE, May 2011, pp. 3281–3288, ISBN: 978-1-61284-386-5. DOI: 10.1109/ICRA.2011.5979641. [Online]. Available: http://ieeexplore.ieee.org/document/5979641/.

[56] D. Fourie, J. Leonard and M. Kaess, 'A nonparametric belief solution to the Bayes tree,' in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, vol. 2016-November, IEEE, Oct. 2016, pp. 2189–2196, ISBN: 978-1-5090-3762-9. DOI: 10.1109/IROS.2016.7759343. [Online]. Available: http://ieeexplore.ieee.org/document/7759343/.

[57] C. Stachniss, J. J. Leonard and S. Thrun, 'Simultaneous Localization and Mapping,' in *Springer Handbooks*, Springer Science and Business Media Deutschland GmbH, 2016, pp. 1153–1176. DOI: 10.1007/978-3-319-32552-1{_}46. [Online]. Available: https://link.springer.com/10.1007/978-3-319-32552-1_46.

[58] Y. Latif, C. Cadena and J. Neira, 'Robust loop closing over time for pose graph SLAM,' *The International Journal of Robotics Research*, vol. 32, no. 14, pp. 1611–1626, Dec. 2013, ISSN: 0278-3649. DOI: 10.1177/0278364913498910. [Online]. Available: http://journals.sagepub.com/doi/10.1177/0278364913498910.

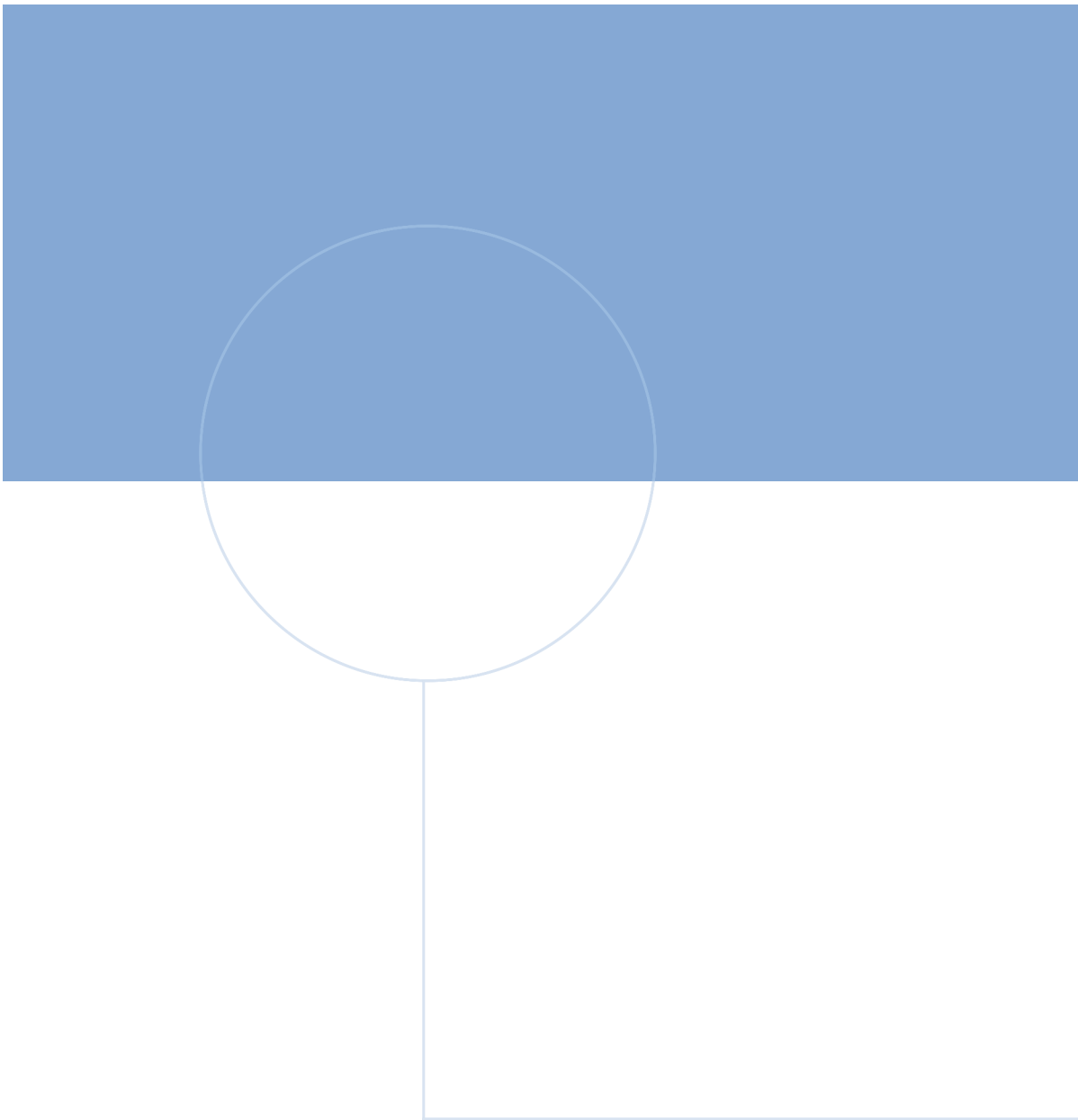
[59] J. Neira and J. Tardos, 'Data association in stochastic mapping using the joint compatibility test,' *IEEE Transactions on Robotics and Automation*, vol. 17, no. 6, pp. 890–897, Dec. 2001, ISSN: 1042296X. DOI: 10.1109/70.976019. [Online]. Available: http://ieeexplore.ieee.org/document/976019/.

[60] S. L. Bowman, N. Atanasov, K. Daniilidis and G. J. Pappas, 'Probabilistic data association for semantic SLAM,' in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, May 2017, pp. 1722–1729, ISBN: 978-1-5090-4633-1. DOI: 10.1109/ICRA.2017.7989203. [Online]. Available: http://ieeexplore.ieee.org/document/7989203/.

[61] K. Doherty, D. Fourie and J. Leonard, 'Multimodal Semantic SLAM with Probabilistic Data Association,' in *2019 International Conference on Robotics and Automation (ICRA)*, vol. 2019-May, IEEE, May 2019, pp. 2419–2425, ISBN: 978-1-5386-6027-0. DOI: 10.1109/ICRA.2019.8794244. [Online]. Available: https://ieeexplore.ieee.org/document/8794244/.

[62] C. Forster, L. Carlone, F. Dellaert and D. Scaramuzza, 'On-Manifold Preintegration for Real-Time Visual–Inertial Odometry,' *IEEE Transactions on Robotics*, vol. 33, no. 1, pp. 1–21, Feb. 2017, ISSN: 1552-3098. DOI: 10.1109/TRO.2016.2597321. [Online]. Available: https://ieeexplore.ieee.org/document/7557075/.

[63] A. C. Rencher and G. B. Schaalje, *Linear models in statistics*. Wiley-Interscience, 2008, p. 672, ISBN: 978-0-471-75498-5. [Online]. Available: https://www.wiley.com/en-ie/Linear+Models+in+Statistics%2C+2nd+Edition-p-9780471754985.



NTNU

Norwegian University of
Science and Technology