Master's thesis

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

Kyrre S. Haugland

Underwater Pose Graph SLAM with DVL-Enhanced Visual Loop Closure for Future Aquaculture

Master's thesis in Cybernetics and Robotics
Supervisor: Annette Stahl
Co-supervisor: Walter Caharija and Bent O.A. Haugaløkken
June 2021

NTNU
Norwegian University of Science and Technology

SINTEF

Kyrre S. Haugland

Underwater Pose Graph SLAM with DVL-Enhanced Visual Loop Closure for Future Aquaculture

Master's thesis in Cybernetics and Robotics
Supervisor: Annette Stahl
Co-supervisor: Walter Caharija and Bent O.A. Haugaløkken
June 2021

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

Kyrre S. Haugland

Underwater Pose Graph SLAM with DVL-Enhanced Visual Loop Closure for Future Aquaculture

This master thesis presents a method for solving the simultaneous localization and mapping (SLAM) problem for a remotely operated vehicle (ROV) doing inside traversal of an aquaculture net pen. The method proposed is a six-degrees-of-freedom pose-graph SLAM algorithm with Doppler velocity log (DVL) enhanced visual loop closures. To the best of the author's knowledge, this problem has never been solved using pose graph SLAM in this environment. The pose-graph method formulates the SLAM problem as a non-linear optimization problem of all the ROV sensor measurements and their respective uncertainties. The non-linear problem can be reformulated as a least-squares problem and solved with the Levenberg Marquardt Algorithm by using the incremental smoothing and mapping 2 (iSAM2) framework.

This work has created a novel data association algorithm for solving the SLAM loop closure- problem inside the fish cage using a mono camera. The algorithm is based on the ROV net inspection procedure, where the ROV is pointing towards the net while traversing it. Therefore, potential image loop closure candidates are filtered on the similarity of depth- and heading measurements when the images were taken. To also account for the fact that the fish cage is a sparse featured environment, a global saliency measure was adapted to aid this filtering process further, avoiding matching low featured scenes. Finally, after filtering image candidates based on these three measures, the cosine similarity of term frequency–inverse document frequency (TF-IDF) histograms of image visual words was used to sort out the best loop closure candidates.

Assuming that the net pen is a planar structure, homography and homography decomposition can be used to obtain the relative translation and rotation between two images used for loop closure. As the homography decomposition solution is ambiguous and only provides translation up to scale, the DVL measurements have been used to estimate the plane structure in front of the ROV and solve the ambiguity and scaling problem.

The system was tested on real-world datasets provided by SINTEF Ocean, and the pose-graph showed promising results as long as the planar net-pen assumption was fulfilled. The robustness of the solution with respect to the planer net-pen assumption must be addressed in future work.

Sammendrag

Kyrre S. Haugland

Underwater Pose Graph SLAM with DVL-Enhanced Visual Loop Closure for Future Aquaculture

Denne masteroppgaven presenterer en metode for å løse lokaliserings- og kartleggingsproblemet (SLAM) for en fjernstyrt undervannsfarkost (ROV) i en merde. Dette problemet er løst ved å bruke en poseringsgraf-SLAM-algoritme med seks frihetsgrader og forbedret visuell sløyfelukking ved bruk av DVL. Etter forfatterens fatning så har dette problemet aldri blitt løst før ved bruk av en poseringsgraf SLAM algoritme. Poseringsgraf metoden formulerer SLAM problemet som et ikke-lineært optimaliseringsproblem som prøver å ta hensyn til alle sensormålingene og deres relaterte usikkerheter. Det ikke-lineære optimaliseringsproblemet kan bli reformulert som et minste kvadraters problem som kan bli løst med Levenberg Marquardt algoritmen via iSAM2 rammeverket.

I dette arbeidet har det blitt laget en ny dataassosieringsalgoritme for å løse sløyfelukkingsproblemet i SLAM ved å bruke et monokamera. Algoritmen er basert på hvordan ROV-inspiseringer blir utført i merden. Under inspeksjon siktes ROV-en mot merden når den traverserer den. Potensielle sløyfelukkingskandidater blir derfor filtrert basert på dybde og retningen ROV-en peket i da bilde ble tatt. Ettersom merden er et miljø hvor det finnes få karakteristiske trekk, så ble et mål, kalt for global saliency, brukt til å måle hvor unikt et bilde framsto. Etter sløyfelukkingskandidatene er filtrert på *global saliency*, retning og dybde, så har bildenes TF-IDF histogrammer blitt sammenlignet ved bruk av cosinuslikhet, for å finne de beste kandidatene.

Ved å anta at fiskenettet er plant, så kan man bruke en projektiv transformasjon til å finne den relative transformasjonen mellom to bilder tatt fra ulike vinkler. Selv om denne projektive transformasjonsdekomponeringen vil ha to løsninger med ukjent bildedybde, så kan man finne den riktige løsningen, samt bildedybden, ved å bruke DVL målingene til å estimere et plan foran ROV-en.

Systemet ble testet på data produsert av SINTEF Ocean sin ROV og algoritmen ga lovende resultater så lenge planantagelsen var reel. Hvor robust denne antagelsen er må bli studert mer i fremtiden.

Acknowledgements

This thesis represents the final requirements of my MSc program in Cybernetic and Robotics at the Norwegian University of Science and Technology. The thesis has been carried out as a collaboration between SINTEF Ocean and NTNU, as part of SFI Exposed (RCN pr. num. 237790), where I have worked under the supervision of associate professor Annette Stahl and co-supervisors Walter Charija and Bent. O.A Haugaløkken. I want to thank Annette for her theoretical expertise and guidance, and Bent for continually coming with proposals and great inputs. I also want to give a special thanks to Walter Charija for always being uplifting throughout the whole process and for good conversations when ski touring. SINTEF Digital has also been involved in this work, and I would like to thank the team for contributing to ideas and guidelines. Lastly, I would like to thank all the fantastic people I have met during my studies who have made my stay here an unforgettable experience.

Kyrre S. Haugland

Contents

List of Figures

List of Abbreviations

2D	Two-**D**imensional
3D	Three-**D**imensional
AUV	**A**utonoumous **U**nderwater **V**ehicle
BOW	**B**ag **O**f **W**ords
CLAHS	**C**ontrast-**L**imited **A**daptive **H**istogram **S**pecification
DOF	**D**egrees **O**f **F**reedom
DR	**D**ead **R**eckoning
DVL	**D**oppler **V**elocity **L**og
EKF	**E**xtended **K**alman **F**ilter
GPS	**G**lobal **P**ositioning **S**ystem
GT	**G**round **T**ruth
IDF	**I**nverse **D**ocument **F**requency
iSAM	**I**ncremental **S**moothing **A**nd **M**apping
LC	**L**oop **C**losure
LOG	**L**aplacian **O**f **G**aussian
LOS	**L**ine-**O**f-**S**ight
LS	**L**east **S**quares
MAP	**M**aximum **A** **P**osteriori
NED	**N**orth-**E**ast-**D**own
NF	**N**et-**F**ollowing
ORB	**O**riented FAST and **R**otated **B**RIEF
PCCS	**P**ose-**C**onstrainted **C**orrespondance **S**earch
PDF	**P**robabilistic **D**ensity **F**unction
PIRF	**p**ositional-**I**nvariant **R**obust **F**eature
RANSAC	**Ran**dom **Sa**mple **C**onsensus
ROV	**R**emotely **O**perated **V**ehicle
SIFT	**S**cale -**I**nvarient **F**eature **T**ransform
SLAM	**S**imultaneous **L**ocalization **A**nd **M**apping
SOTA	**S**tate **O**f **T**he **A**rt
SO(n)	**S**pecial **O**rthogonal Group
SURF	**S**peeded **U**p **R**obust **F**eatures
SVD	**S**ingular **V**alue **D**ecomposition
TF-IDF	**T**erm **F**requency - **I**nverse **D**ocument **F**requency
USBL	**U**ltra-**S**hort **B**ase**L**ine
VO	**V**isual **O**dometry

Chapter 1

Introduction

1.1 Background & Motivation

It is estimated that by the year 2030 the global food consumption will have increased by 70% compared to 2016 [41]. Therefore, a question raised is how to support such a demand in a world affected by climate changes. Today, only 2% of human food consumption stems from the ocean [41], yet the food from the ocean is considered one of the most sustainable alternatives. This is because most marine animals feed on plankton and algae, which is considered the lowest trophic level of the ecosystem. Hence, fish farms are a convenient solution for handling these demands. However, there are many challenges related to aquaculture, where one of them is fish escaping. Escaping fish is a financial inconvenience for the fish farmers and an environmental problem as the cross-breeding between wild fish and farm fish results in a less genetically diverse species.

FIGURE 1.1: SINTEF ACE Rataren, a full-scale laboratory facility designed to develop and test new aquaculture technologies. Courtesy of [2]

It has been reported that two-thirds of registered fish escaping from Norwegian aquaculture stem from tears in the net [46]. Therefore, these holes need to be repaired, which is solved today by combining a remotely operated vehicle (ROV) to detect the holes and use divers to repair them. Net repair is a risky procedure for the diver as their heavy equipment can easily get entangled in the net, which can have lethal consequences. The current state-of-the-art (SOTA) solution for locating holes is to use an ROV equipped with a forward-looking camera, depth sensor, and compass. A combination of the ROV pilot's experience and the measurements from these sensors are then used to give a rough estimate of where the hole might be [40]. This type of estimation cannot produce an accurate position estimate. Therefore, it is desirable to find a new way of obtaining a more precise and reliable estimate that can better ensure the safety of the divers.

The aquaculture environment is a challenging scenery for obtaining position estimates. One reason is that global positioning system (GPS) signals are not available here as the electromagnetic signals are strongly attenuated underwater [32]. Therefore other sensors need to be used to acquire a position estimate; however, many of them are not as effective in the underwater environment compared to land. Optical sensors as cameras and lasers are limited as turbid waters cause attenuation and light

scattering [4]. In addition, acoustic sensors are known to have a high signal-to-noise ratio [32]. Other issues are the high concentration of fish in the fish cage environment that disrupt measurements and the fact that the fish cage flexible structure. As it is flexible, it can be deformed by the drag forces caused by ocean currents, and it has been reported that currents over 0.6m/s can reduce the fish cage volume by 30% [29].

The current SOTA solution used for position tracking in an environment where there are no GPS signals is the simultaneous localization and mapping (SLAM) algorithm. The SLAM framework tries to build a map of the environment while using this map to deduce its position [11]. This is solved by using sensor fusion of sensor measurement to reduce the individual uncertainties related to each sensor measurement. A critical part separating the SLAM algorithm from other position estimators is loop closure (LC), which allows for position correction if the robot re-detects a previously observed scene [11]. Today there are several ways of solving the SLAM problem, where among others are the EKF-SLAM and visual SLAM. However, implementing these solutions directly into the fish cage environment is challenging. EKF-SLAM is especially vulnerable to incorrect landmark association due to incrementally fusing its new measurements to obtain a current pose estimate [38]. Visual SLAM relies on visual features to obtain reasonable visual odometry estimates. Visual odometry used in visual SLAM would also most likely fail in the fish cage environment as it relies on pair-wise image matching, which is difficult in a scene with similar features causing ambiguity while also having moving fish. However, cameras are one of the best sensors for detecting previously seen scenes and are therefore an excellent sensor for obtaining loop closures (LC).

1.2 Problem Statement

The problem statement is to create a functional SLAM framework for the fish cage environment. As some sensor measurements in the underwater environment are not as reliable as on land, it is desired to create a framework that fuses all sensors to achieve the most reliable six-degrees-of-freedom (DOF) position estimate. The loop closure problem is a big part of SLAM. The thesis also seeks to find a feasible way of detecting LCs in the dynamic and feature sparse aquaculture environment using a mono camera. It is desired that this framework can be used for real-time operations in the future, but this thesis has not been focusing on achieving a computationally optimal code.

1.3 Data Acquisition Platform

1.3.1 The Argus Mini Remotely Operated Vehicle

The ROV used for the experiments conducted in this master thesis is the Argus Mini ROV, manufactured by Argus Remote Systems AS. The Argus Mini is illustrated in figure 1.2. This ROV is of the observation class and built for intervention and inspection purposes.

Instrumentation

The vessel is mounted with six sensors: a SONY FCB-EV7100 Full HD, 60Hz camera, a fluxgate compass, a depth sensor, a gyro, a tilt sensor for pitch and roll, and a

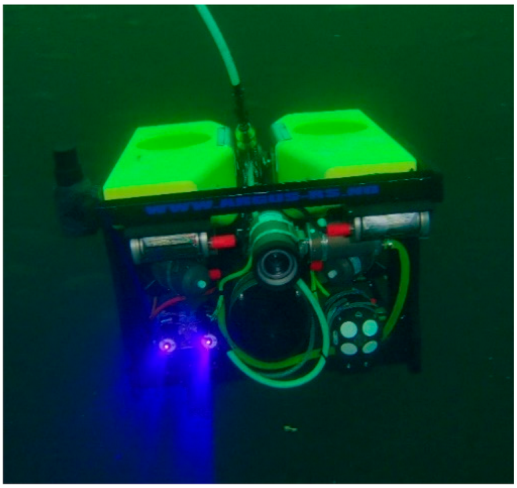


FIGURE 1.2: Illustration of the Argus Mini ROV with mounted lasers. Courtesy of [4]

Nortek DVL 1000 velocity sensor. As the Argus Mini is used to inspect the net pen, the DVL is mounted horizontally. Orienting the DVL towards the net and having a high acoustic frequency configuration allows for an acoustic rebound from the fish cage [40]. This way, the unconventional setup provides velocity readings.

Another part of the ROV instrumentation is the Sonardyne Ultra-Short Baseline (USBL). This system consists of a Micro-Ranger Transceiver attached to a surface vessel outside the fish cage and a nano-transponder mounted onto the ROV that stays on the inside of the net. The USBL transponder/transceiver system communicates over a low frequency, enabling signal transmission across the net [40]. This way, the USBL system provides absolute position estimates of the ROV.

The Argus Mini has a dimension of $L \times W \times H = 0.9m \times 0.65m \times 0.6m$, and is equipped with six ARS800 thrusters. Four of these are oriented in the horizontal plane, while the others are pointing vertically. This way, the Argus Mini is fully actuated in the horizontal plane while having 4 degrees of freedom (DOF); surge, sway, heave, and yaw. In addition, the ROV is passively stabilized by gravity in pitch and roll.

Control System

Based on SINTEF's in-house control systems, the Argus Mini has three operational modes: manual (assisted with auto-heading, AH, and auto-depth, AD), dynamic positioning and net-following (NF) mode [4]. In net-following mode, the ROV is controlled such that it traverses the net-pen at a given depth. The NF mode is only available if the ROV is pointing towards the net and gets good DVL feedback from the net.

Another feature of the ROV control system is that an extended Kalman filter (EKF) has been implemented to acquire a better positional estimate of the ROV. This filter uses the USBL measurements, the DVL, depth sensor, compass, and gyro to obtain a more reliable positional estimate. As there is no ground truth in the datasets, this will be used as a quasi-ground truth in the experiments conducted in this thesis.

1.3.2 The Operational Environment

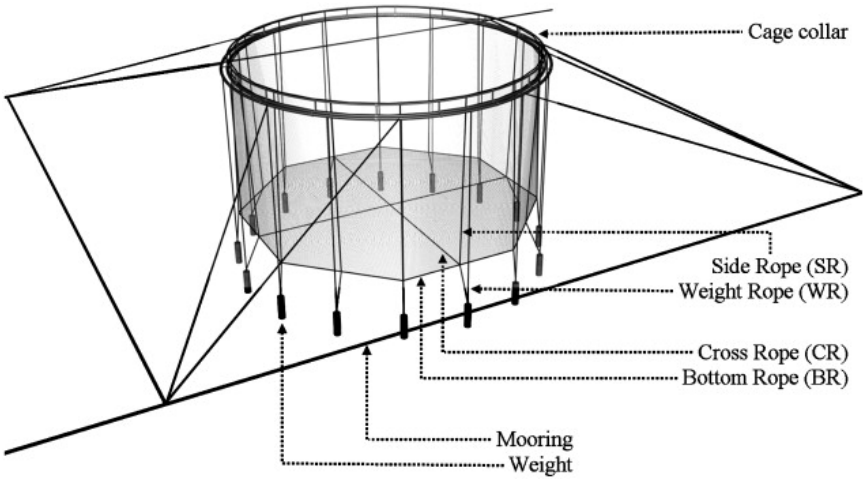


FIGURE 1.3: Traditional circular fish net. Courtesy of [37]

This thesis addresses in cage operations within sea-based aquaculture. An aquaculture net-pen is shown in figure 1.3 and this is similar to the ones used at SINTEF's fish farms, where datasets for this thesis were acquired. The fish cages used in this experiment had a diameter of 50m and 15m deep, and the bottom had a conical shape. There are several mooring points along the net, and these weights cause sharp edges to be formed. As these anchor points help to hold the net in place, the fish cage can be viewed as a polygon-shaped box with planar walls. The net is constructed of nylon threads, and as nylon is a flexible material, it is affected by the drag forces of the ocean currents. A net-pen. such as the one depicted in fig 1.3, contains on average 180000 to 200000 salmon individuals, for a total biomass at slaughter of up to 1000 tons, assuming 5kg weight per individual at production.

1.4 Main Contributions

Main Contribution 1: Implementation of a six-degrees-of-freedom pose-graph SLAM in the challenging fish cage environment. This framework was selected due to its robust nature of formulating the SLAM problem as a non-linear optimization problem, allowing for simple addition of new sensor measurement to the system while also not being as fragile towards bad data association as the EKF-SLAM.

Main Contribution 2: A novel data association method for solving the loop closure problem in this environment. The data association algorithm is based on the current net inspection procedure used by SINTEF, where the ROV traverses the net while pointing towards it. The data association algorithm created is a filtering method that proposes image loop closure candidates based on the similarity between heading and depth measurements recorded when the images were taken. To avoid matching between low feature images, it uses the global saliency measure proposed in [27] and finds the best LC-candidate by evaluating the cosine similarity of image TF-IDF histogram.

Main Contribution 3: Using DVL measurements to scale and find the correct homography decomposition solution from image loop closure.

1.5 Outline

This thesis is written in 6 chapters, where the remanding chapters are outlined as follows:

- Chapter 2 explains the general theoretical concepts which are essential for the construction of the pose SLAM algorithm.
- Chapter 3 represents literature that was used as inspiration in this work.
- Chapter 4 gives a detail description of the method and implementation of the SLAM algorithm, describing its front- and back.
- Chapter 5 shows and discusses the results of running the algorithm on three different datasets from net inspection using the Argus mini ROV. It also discusses some of the drawbacks of the bag of words algorithm used in this work.
- Chapter 6 provides the conclusion and discusses future work

Chapter 2

Theory

The following sections will explain general theory for understanding the method and implementation used in this thesis. The following topics will be addressed in the given order: Reference Frames & Transformations, Camera Measurements, Probability Theory, Non-Linear Optimization and iSAM2. Note that section 2.1, and subsections 2.3.1 and 2.3.1 are based on previous work "Acoustic DVL-SLAM for Future Autonomous Aquaculture", the project that lead to this thesis and that was delivered in December 2020.

2.1 Reference Frames & Transformations

The mini Argus ROV and all other vehicles have sensors that are used to obtain information about either the velocity of the vehicle or obtain knowledge about the environment, e.g., if one has a lidar that provides range data of its surroundings. As these sensors will be placed in different positions on the vehicle, the measurements will be related to the relative orientation and position (*pose*) of the specific sensor. To simplify the process of relating these measurements to the world the vehicle is traveling in, one uses different *reference frames* and converts between them by knowing the relative position and orientation of one frame to the other. According to [30], "A reference frame, or simply frame, can be pictured as a rigid lattice attached to an observer, relative to which that observer quantifies the motion of objects.". By this definition, reference frames can change pose relative to each other, despite being attached to an observer. This way, if one has a reference frame attached to the body of a vehicle, the body frame will be fixed to the body while the vehicle is moving in the world.

NED frame

NED (North East Down) frame is a coordinate frame fixed to the surface of the Earth, where the origin is defined relative to the Earth's reference ellipsoid (WGS-84) [17]. The xy-plane forms the Earth's tangent plane, where the x-axis points towards the north while the y-axis points eastward. The z-direction points downward and is normal to the Earth's surface (tangent plane), forming a right-handed coordinate system.

Body-frame

The body frame is a reference frame that is fixed to the body of a vehicle, and is therefore a moving reference frame. As the body-frame is the reference frame used to describe the linear and angular velocity of the vehicle, the body frame origin is located

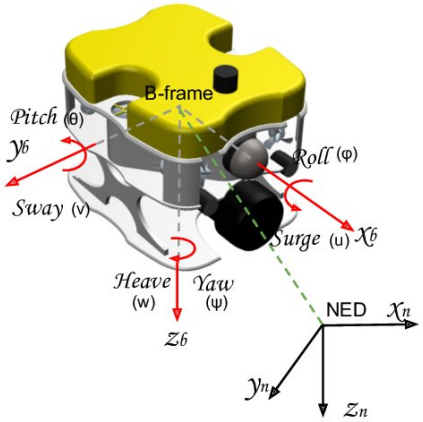
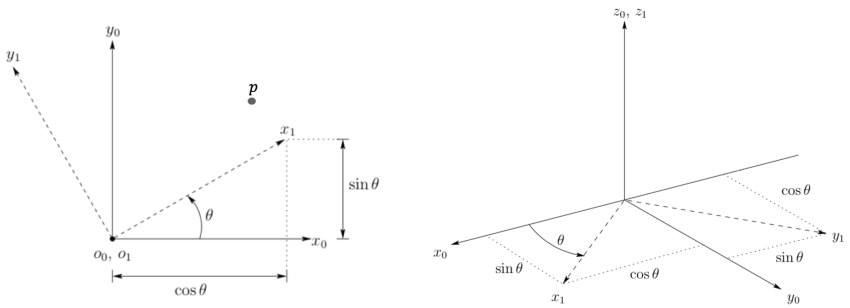


FIGURE 2.1: NED-frame and body-frame illustration. Note in this drawing the NED-frame has the subscript n, however in this article NED-frame is denoted w for world-frame. Courtesy of[48]

at the vehicle's COG (center of gravity) as this is where the forces and moments are acting [17]. The axes of the body frame are chosen to coincide with the *principal axes of inertia*, and are defined as follows; x_b is longitudinal axis directed from aft to fore, y_b is an axis directed to starboard and z_b is normal axis directed from top to bottom [17]. In figure 2.1 an illustration of the body frame and the NED-frame is shown.

Transformation Between Reference Frames

To perform a transformation between one frame to another is done by applying a combination of rotation, aligning the orthonormal axis of both frames, and translation. The rotation of a frame is achieved by using a rotation matrix, while translation is vector addition. By using a homogeneous transformation matrix, one can transform from one frame to another in one calculation. What a rotation matrix and a homogeneous transformation matrix are will be further explained in the following subsections.



(A) 2D- rotation (B) 3D- rotation

FIGURE 2.2: Rotation from coordinate frame (x_0, y_0) to frame (x_1, y_1). Courtesy of [42]

Rotation matrix

Figure 2.2a shows an illustration of two frames oriented at an angle θ with respect to each other. In this figure there is a point $\mathbf{p}$. As the frames are oriented differently the vector representation of the point will also be dissimilar in each frame. If the point is only observed from frame $\mathcal{F}_1$, $\mathbf{p}^1$, one can describe the point in $\mathcal{F}_0$ by using a rotation matrix $\mathbf{R}_1^0(\theta)$ that maps from $\mathcal{F}_1$ to $\mathcal{F}_0$. The frame transformation $\mathbf{R}_1^0(\theta)$ can be described as follows:

$$\mathbf{R}_1^0 = [x_1^0 | y_1^0] \tag{2.1}$$

where x_1^0 and y_1^0 maps the x-axis and y-axis from $\mathcal{F}_1$ to $\mathcal{F}_0$ respectively[42]. From figure 2.2a one obtain:

$$x_1^0 = \begin{bmatrix} \cos\theta \\ \sin\theta \end{bmatrix}, \quad y_1^0 = \begin{bmatrix} -\sin\theta \\ \cos\theta \end{bmatrix}$$

Inserting this into equation 2.1 gives:

$$\mathbf{R}_1^0 = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{2.2}$$

The rotation matrix from equation 2.2 belong to the *Special Orthogonal group of order* 2, $SO(2)$. Rotational matrices, $\mathbf{R}$, belonging to SO(n) obtains certain properties that can be summarized as follows [42]:

- The columns and rows are mutually orthogonal
- The columns and rows are unit vectors
- $\det \mathbf{R} = 1$
- $\mathbf{R}^{-1} = \mathbf{R}^T$
- $\mathbf{R} \in SO(n)$ and $\mathbf{R}^T \in SO(n)$

As seen from the forth property, the inverse rotation, mapping from $\mathcal{F}_0$ to $\mathcal{F}_1$, can be found by $\mathbf{R}_0^1 = (\mathbf{R}_1^0)^T$.

As the rotation in figure 2.2a occurs on a plane, the rotation can easily be transferred to 3D as shown in figure 2.2b. Since the z-axis before and after the rotation aligns, the $SO(3)$ becomes:

$$\mathbf{R}_1^0 = \mathbf{R}_{z,\theta} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.3}$$

As for the 2D-rotation, it can be shown that the $SO(3)$ holds the properties listed previously. The rotation in equation 2.3 is known as a *basic rotation matrix*. The other basic rotation matrices belonging to $SO(3)$ are rotations about the x-, and y-axis can be calculated to be equal to:

$$\mathbf{R}_{x,\theta} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}, \quad \mathbf{R}_{y,\theta} = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

Rotation Composition

Composition of rotations to achieve more complex orientations can be obtained by multiplying together basic rotations. If one has three frames $\mathcal{F}_0$, $\mathcal{F}_1$ and $\mathcal{F}_2$, where $\mathcal{F}_0$ and $\mathcal{F}_1$ are related by the rotation $\mathbf{R}_1^0$ and $\mathcal{F}_1$, $\mathcal{F}_2$ are related by the rotation $\mathbf{R}_2^1$, then the relationship between $\mathcal{F}_0$ and $\mathcal{F}_2$ can be found to be:

$$\mathbf{R}_2^0 = \mathbf{R}_1^0 \mathbf{R}_2^1 \tag{2.4}$$

Equation 2.4 is the composition law for rotational transformation and it can be used to transform a point $\mathbf{p}^2$ to $\mathbf{p}^0$ by first transforming it to $\mathbf{p}^1$ [42]. With this composition the first rotation, $\mathbf{R}_2^1$, is therefore relative to the $\mathcal{F}_2$ frame, while the second rotation, $\mathbf{R}_1^0$ is relative to $\mathcal{F}_1$. One therefore says that the rotation occurs relative to the *current frame*.

It is also possible doing rotation composition about a *fixed frame*, e.g. the NED-frame. Such rotation composition can be derived by using the current frame composition shown in equation 2.4. If one has a rotation $\mathbf{R}_1^0$ relating $\mathcal{F}_0$ and $\mathcal{F}_1$, while also having another rotation $\mathbf{R}$ that is relative to $\mathcal{F}_0$, then the rotation $\mathbf{R}$ can be expressed in the *current* frame as $(\mathbf{R}_1^0)^{-1}\mathbf{R}\mathbf{R}_1^0$ [42]. The *fixed frame* rotation composition therefore becomes:

$$\mathbf{R}_2^0 = \mathbf{R}_1^0[(\mathbf{R}_1^0)^{-1}\mathbf{R}\mathbf{R}_1^0] = \mathbf{R}\mathbf{R}_1^0 \tag{2.5}$$

Hence, when using fixed frame rotation composition, one is therefore *pre multiplying* the rotations, while for current frame rotation composition, one does *post multiplication*.

Roll, Pitch, Yaw

An intuitive representation of a general SO(3) rotation matrix is the *roll, pitch, yaw-representation*. This representation uses a fixed-axes rotation, where roll (ϕ), pitch (θ), yaw (ψ) denotes the Euler angles used for the basic rotations about the x-, y- and z-axis respectively. Based on the results from equation 2.5 one can express any $SO(3)$ rotation by doing successive rotations about the x,y and z-axis:

$$\begin{aligned}
\mathbf{R}_{x,y,z} &= \mathbf{R}_{z,\phi}\mathbf{R}_{y,\theta}\mathbf{R}_{x,\psi} \\
&= \begin{bmatrix} c_\psi & -s_\psi & 0 \\ s_\psi & c_\psi & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} c_\theta & 0 & s_\theta \\ 0 & 1 & 0 \\ -s_\theta & 0 & c_\theta \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & c_\phi & -s_\phi \\ 0 & s_\phi & c_\phi \end{bmatrix} \\
&= \begin{bmatrix} c_\phi c_\theta & -s_\phi c_\psi + c_\phi s_\theta s_\psi & -s_\phi s_\psi + c_\phi s_\theta c_\psi \\ s_\phi c_\theta & c_\phi c_\psi + s_\phi s_\theta s_\psi & c_\phi s_\psi + s_\phi s_\theta c_\psi \\ -s_\theta & c_\theta s_\psi & c_\theta c_\psi \end{bmatrix}
\end{aligned} \tag{2.6}$$

Here c_α and s_α denotes $\cos\alpha$ and $\sin\alpha$ respectively. One drawback of using the Euler angle representation is that there is a singularity when the θ has an angle of $\pm 90°$; causing the yaw angle to be undefined [3]. This is also known as a gimbal lock, but in the context of this work this will not be an issue as the ROV is passively stabilized by gravity in pitch and roll [4].

Rigid Motion

A combination of positional translation and rotation is called *3D rigid body* transformation [18]. This could either be used as a procedure to transfer between different

reference frames, or to transfer a frame to a new location and orientation in space, as done in dead reckoning. An example of a rigid body motion is illustrated in figure 2.3. This shows how one can transform points observed in the body frame $\mathcal{F}_b$ to the world frame $\mathcal{F}_w$ by first applying the rotation $\mathbf{R}_b^w$ in order to align the points with $\mathcal{F}_w$ and then adding the translation vector of $\mathcal{F}_b$ with respect to $\mathcal{F}_w$, $\mathbf{t}_{bw}^w$. A vector described as $\mathbf{v}_{bc}^a$, where $\{a\}$, $\{b\}$ and $\{c\}$ are coordinate systems, denotes the vector of the point o_b with respect to coordinate system $\{c\}$ expressed in $\{a\}$ [17]. Putting the rigid body transformation into mathematical terms one gets:

$$\mathbf{p}^w = \mathbf{R}_b^w \mathbf{p}^b + \mathbf{t}_{wb}^w \tag{2.7}$$

The transformation can also be expressed as a single matrix multiplication as follows:

$$\tilde{\mathbf{p}}^w = \mathbf{T}_b^w \tilde{\mathbf{p}}^b \tag{2.8}$$

where:

$$\mathbf{T}_b^w = \begin{bmatrix} \mathbf{R}_b^w & \mathbf{t}_{wb}^w \\ \mathbf{0} & 1 \end{bmatrix}, \qquad \tilde{\mathbf{p}}^b = \begin{bmatrix} \mathbf{p}^b \\ 1 \end{bmatrix}, \qquad \tilde{\mathbf{p}}^w = \begin{bmatrix} \mathbf{p}^w \\ 1 \end{bmatrix}$$

utilizing a homogeneous transformation matrix and coordinate vectors. Inserting the rotation expression from equation 2.6 into transformation matrix $\mathbf{T}_b^w$ of equation 2.8 results in:

$$\mathbf{T}_b^w = \begin{bmatrix} \mathbf{R}_b^w & \mathbf{t}_{wb}^w \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} c_\phi c_\theta & -s_\phi c_\psi + c_\phi s_\theta s_\psi & -s_\phi s_\psi + c_\phi s_\theta c_\psi & t_{bw,x}^w \\ s_\phi c_\theta & c_\phi c_\psi + s_\phi s_\theta s_\psi & c_\phi s_\psi + s_\phi s_\theta c_\psi & t_{bw,y}^w \\ -s_\theta & c_\theta s_\psi & c_\theta c_\psi & t_{bw,z}^w \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.9}$$

where the Euler angles are based on the relative orientation of the principal axes of $\mathcal{F}_b$ with respect to $\mathcal{F}_w$. The inverse transformation going from world to body can be found to be equal to [42]:

$$\mathbf{T}_w^b = \begin{bmatrix} (\mathbf{R}_b^w)^T & -(\mathbf{R}_b^w)^T \mathbf{t}_{wb}^w \\ \mathbf{0} & 1 \end{bmatrix}$$

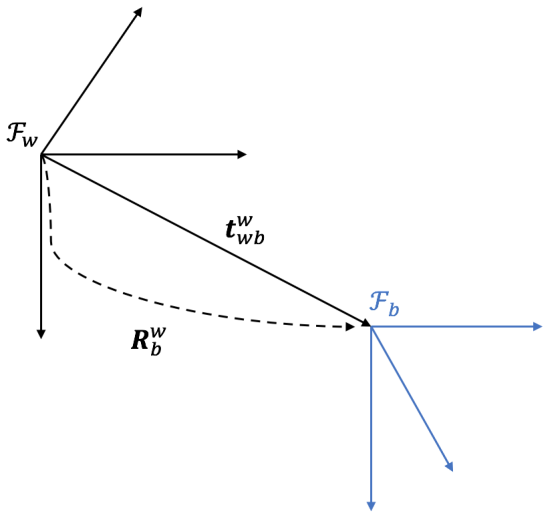


FIGURE 2.3: Rigid body motion from world frame to body frame.

2.2 Camera Measurements

This section will explain important theoretical concepts used for obtaining image features and knowing relative translation and rotation between two camera frames. The camera projection model will be explained first as this is essential for knowing how images capture a scene.

2.2.1 Camera Projection Model

The geometric camera model projects 3D points, $\mathbf{x}^c$, observed in the camera frame $\mathcal{F}_c$, onto pixels in the image as shown in equation 2.10

$$\mathbf{u} = \begin{bmatrix} u \\ v \end{bmatrix} = \pi(\mathbf{x}^c) \tag{2.10}$$

where π is the mapping function (camera model). The most common camera model is the *perspective camera model*. This model is illustrated in figure 2.4a where a 3D-point, $\mathbf{x}^c$, undergoes a central projection through the origin of $\mathcal{F}_c$ and mapped on the x-y-plane at z=-f. Here f is the focal length, which is an intrinsic camera property. When using the perspective camera model, the 3D-points is flipped; therefore a more suitable representation is to use the *frontal projection model*, shown in figure 2.4b, which maps the point onto the plane at $z = f$. To obtain the frontal projection model it is convenient to first project x^c onto a normalized image plane where $z = 1$. This is achieved by first using homogeneous perspective projection, Π_0, to convert from homogeneous coordinates $\tilde{\mathbf{x}}^c$ to Cartesian camera frame coordinates:

$$\tilde{\mathbf{x}}_n = \Pi_0 \tilde{\mathbf{x}}^c = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tilde{\mathbf{x}}^c = \mathbf{x}^c \tag{2.11}$$

And then obtaining normalized image coordinates by normalizing the points based on depth, z^c:

$$\check{\mathbf{x}}_n = \begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix} = \begin{bmatrix} x^c/z^c \\ y^c/z^c \\ 1 \end{bmatrix} = \frac{1}{z^c}\mathbf{x}^c \tag{2.12}$$

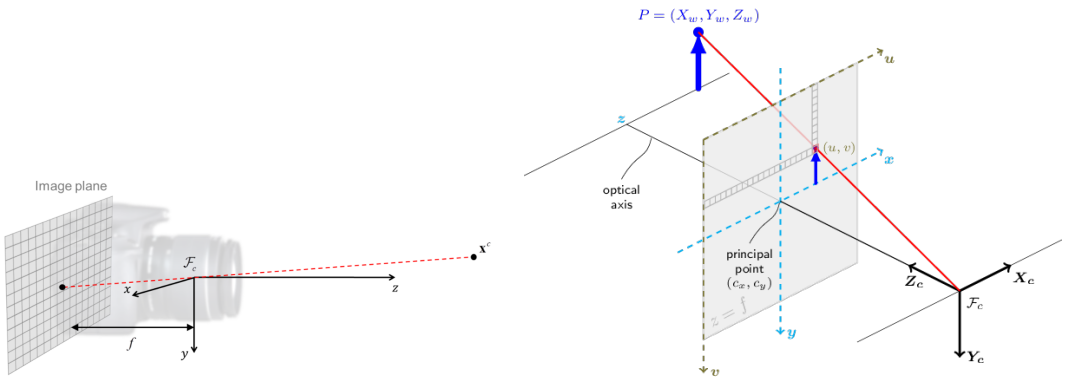
Based on the results from 2.12 one can convert the normalized image coordinates to pixel coordinates using the following equation:

$$\mathbf{u} = \pi_p(\mathbf{x}_c; \mathbf{K}) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \mathbf{K} \frac{1}{z^c}\mathbf{x}^c = \begin{bmatrix} f_u \frac{x^c}{z^c} + c_u \\ f_v \frac{y^c}{z^c} + v_u \end{bmatrix} \tag{2.13}$$

Here the first matrix converts from homogeneous pixel coordinates to 2D Euclidean space, while $\mathbf{K}$ is an upper triangular matrix called the *intrinsic camera matrix*. $\mathbf{K}$ is an affine transformation matrix which contains idealized camera-specific properties to map from normalized image coordinates to pixels. K is constructed as follows:

$$\mathbf{K} = \begin{bmatrix} f_u & s_\theta & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

where the parameters are defined as [18]:



(A) Perspective camera model (B) Frontal projection model

FIGURE 2.4: Geometric camera models. Courtesy of [18] and [7]

- f_u - Defines a pixel's horizontal unit length. This can also be defined as fs_u where f is the focal length, while s_u is a scaling factor in v-direction.
- f_v - Defines a pixel's vertical unit length. This can also be defined as fs_v where f is the focal length, while s_v is a scaling factor in v-direction.
- c_x - Defines the principal point in u-coordinate of the image frame.
- c_y - Defines the principal point in v-coordinate of the image frame.
- s_θ - Defines a skew factor that is proportional to $\cot\theta$, where θ is the angle between the u- and v-axis.

Note that in this work the idealized **K** matrix has been constructed such that s_θ is set to zero.

2.2.2 Scale- Invariant Feature Transform

To obtain a loop closure in SLAM using a camera, one needs to obtain features that can be tracked from one image to another. There are several established feature detectors, where one of them is the Scale- Invariant Feature Transform (SIFT). The SIFT algorithm for extracting features is divided into four consecutive steps:

- Scale-space extrema detection
- Keypoint localization
- Orientation Assignment
- Keypoint descriptor

Scale-Space Extrema Detection

One important property feature detectors need to have is that they are invariant to translation, rotation and scale. This is a necessity when matching features from two images as these images could be captured at different locations and orientations in space. The SIFT algorithm finds scale invariant image *keypoints* (pixel positions) by locating local maximas over different Gaussian scale spaces by using an approximation of the Laplacian of Gaussian (LoG). It approximates LoG with the Difference of Gaussian (DoG) to be more computationally efficient. The DoG calculation is visualized in figure 2.5. When DoG is calculated, the maximas of a keypoint $[u, v]^T$ can be found by comparing the DoG at different scales.

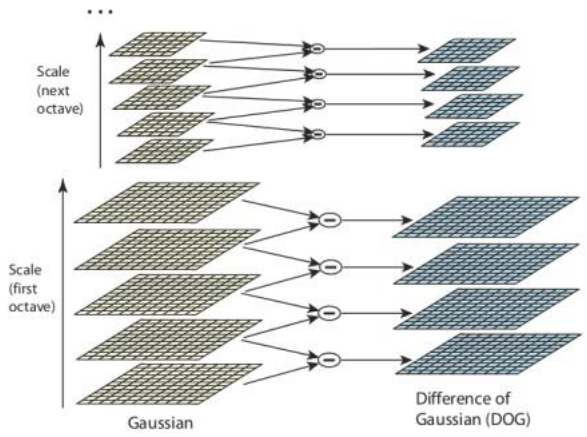


FIGURE 2.5: Difference of Gaussian calculations over different scale spaces. Courtesy of [21]

Keypoint localization

After scale-space extermas are detected, these points are refined by applying a *contrast threshold* and a *edge threshold*. The constrast threshold is used to remove low-contrast maximas, while the edge threshold is used to remove edge keypoints. The reason why one would like to remove edges is due to the fact that DoG has a higher response to edges [21].

Orientation Assignment

Orientation is assigned to each keypoint afterward to enable rotational invariance. This assigned orientation has both a magnitude and a direction which are found from a scale-dependent neighborhood. This neighborhood is then divided into 36 bins over 360 degrees. Pixels falling under the same degree category have their magnitude summed. The orientation of the keypoint is then based on the maximum magnitude, but if other bin-directions are 80% of the maximum, then these are also used to describe the keypoint's orientation.

Keypoint Descriptor

The keypoint descriptor is the property describing the characteristics of a detected feature. The SIFT descriptor is generated by considering a 16×16 neighbourhood around the keypoint, which is then divided into 16 $4x4$-subregions. Then a down scaled version of the keypoint orientation assignment is performed, where an eight bin orientation histogram is created for each subregion. As one has eight bins in each of the 16 subregions, this gives 128 bins in total, which is used as a vector to describe a keypoint. Hence, the SIFT feature can be described with the following vector;

$$SIFT_{feature} = [p, s, r, f]^T \tag{2.14}$$

where p is the pixel position of the keypoint, s is the scale, r is the orientation and f is the feature descriptor.

2.2.3 Homography & Homography Decomposition

When a planar scene is observed from two camera angles and the intrinsic camera matrix is known, the camera displacement can be found by decomposing the *homography*

matrix relating the two views [35]. As the net pens of the cage can be approximated to be planar, this is therefore relevant for determining camera displacement of relocated scenes. In figure 2.6 a planar scene is observed by two different camera views. In this figure there is a plane, π; described by its normal vector $\mathbf{n}$, which consist of n 3D-points, $\mathcal{P}_i = (x_i^w, y_i^w, z_i^w)$. The plane is observed by a source frame, $\mathcal{F}_s$ and a desired frame $\mathcal{F}_d$, where the 3D-points of the plane are projected to the normalized homogeneous image coordinates $\mathbf{m}_i^* = \tilde{\mathbf{m}}_i^s = [x_i^s, y_i^s, 1]^T$ and $\mathbf{m}_i = \tilde{\mathbf{m}}_i^d = [x_i^d, y_i^d, 1]^T$. The pixel coordinates of both camera frames, $\mathbf{u}^d$ and $\mathbf{u}^s$, can be found by using equation 2.13. The depth/distance away from the plane is termed d^* and d for desired and current frame respectively.

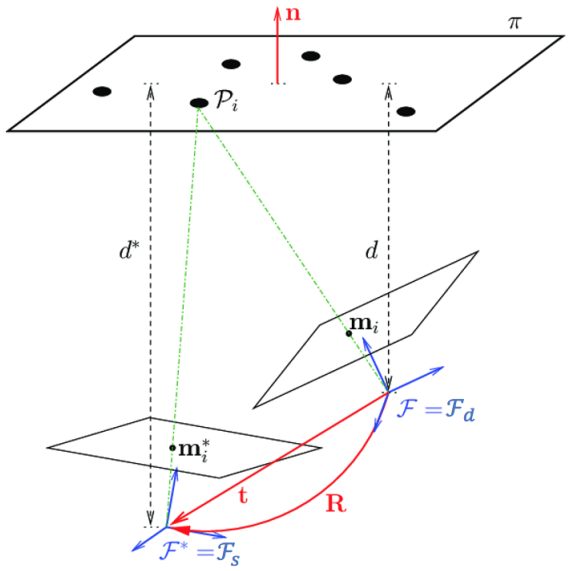


FIGURE 2.6: Desired- and source frame ($\mathcal{F}_d$ and $\mathcal{F}_s$) observing the same planar scene, $\mathcal{P}_i$. Courtesy of [35]

According to *theorem 2.10* in [1] a homography (also known as a *projectivity*) is defined as in definition 1 where $\mathbb{P}^2$ is a projective plane.

DEFINITION 1:

A mapping h : $\mathbb{P}^2 \to \mathbb{P}^2$ is a projectivity if and only if there exists a non-singular 3×3 matrix $\mathbf{H}$ such that for any point in $\mathbb{P}^2$ represented by a vector $\mathbf{x}$ it is true that $\mathbf{h}(\mathbf{x}) = \mathbf{H}\mathbf{x}$

From this definition there exists a *Euclidean homography matrix* that maps from $\mathcal{F}_s$ to $\mathcal{F}_d$ according to:

$$\alpha_h \tilde{\mathbf{m}}^d = \mathbf{H}\tilde{\mathbf{m}}^s \tag{2.15}$$

where α_h is a scaling factor. The projective transformation achieved by the homography matrix can be found to be equal to [35]:

$$\mathbf{H} = \mathbf{R} + \mathbf{t}\mathbf{n}^T \tag{2.16}$$

where $\mathbf{R} = \mathbf{R}_s^d$, $\mathbf{t} = \mathbf{t}_{ds}^d / d^*$ and $\mathbf{n} = \mathbf{n}^d$. Note that the translational component $\mathbf{t}$ is only found up to scale. Similarly, the *projective homography matrix* $\mathbf{G}$ mapping

pixels seen from one frame to another can be found to be:

$$\alpha_g \tilde{\mathbf{u}}^d = \mathbf{G}\tilde{\mathbf{u}}^s \tag{2.17}$$

where $\mathbf{G}$ is a 3×3 matrix equal to $\mathbf{G} = \gamma\mathbf{K}(\mathbf{R} + \mathbf{t}\mathbf{n}^T)$, where γ is a scaling factor and $\mathbf{K}$ is the intrinsic camera matrix. Both $\mathbf{H}$ and $\mathbf{G}$ can be calculated using *the normalized Direct-Linear Transformation (DLT) algorithm* or *the Gold Standard algorithm* (used by the openCV library [22]) which is explained in detail in *algorithm 4.2* and *algorithm 4.3* in [1].

Homography Decomposition

The homography matrix $\mathbf{H}$ can be decomposed into its respective components shown in equation 2.16. A common method for homography decomposition is Zhang method based on singular value decomposition (SVD) [49]. From SVD one can describe any rectangular matrix $\mathbf{A} \in \mathbb{R}^{n \times m}$ as in equation 2.18.

$$\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{V}^T \tag{2.18}$$

Here, $\mathbf{U} = [\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3]$ and $\mathbf{V} = [\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3]$ are orthogonal matrices, while $\mathbf{\Lambda} = diag(\lambda_1, \lambda_2, \lambda_3)$ is a diagonal matrix. Since $\mathbf{U}$ and $\mathbf{V}$ are orthogonal matrices the property $\mathbf{Q}^T\mathbf{Q} = \mathbf{Q}\mathbf{Q}^T = \mathbf{I}$ holds. Hence, the squared matrix of the homography can be expressed as:

$$\mathbf{H}^T\mathbf{H} = \mathbf{V}\mathbf{\Lambda}^2\mathbf{V}^T \tag{2.19}$$

where $\mathbf{\Lambda}$ contains the eigenvalues of $\mathbf{H}$ while $\mathbf{V}$ contains the eigenvectors [35]. By formulating the homography matrix in equation 2.16 as:

$$\mathbf{H} = \mathbf{R}(\mathbf{I} + \mathbf{R}^T\mathbf{t}\mathbf{n}^T) = \mathbf{R}(\mathbf{I} + \mathbf{t}^*\mathbf{n}^T) \tag{2.20}$$

where $\mathbf{t}^*$ is defined as the normalized translation vector in $\mathcal{F}_s$. Inserting equation 2.20 into 2.19 and using the properties of the $SO(3)$ group one gets:

$$\begin{aligned}\mathbf{H}^T\mathbf{H} &= (\mathbf{I} + \mathbf{t}^*\mathbf{n}^T)^T \underbrace{\mathbf{R}^T\mathbf{R}}_{\mathbf{I}}(\mathbf{I} + \mathbf{t}^*\mathbf{n}^T) \\ &= \mathbf{I} + \mathbf{n}(\mathbf{t}^*)^T + \mathbf{t}^*\mathbf{n}^T + k^2\mathbf{n}\mathbf{n}^T\end{aligned} \tag{2.21}$$

where $k^2 = (\mathbf{t}^*)\mathbf{t}^* > 0$. Based on equation 2.21 and the definition of eigenvalues and eigenvectors $(\mathbf{H}^T\mathbf{H} - \lambda\mathbf{I})\mathbf{v} = 0$) it is evident that one eigenvalue/eigenvector solution is $\lambda = 1$ and $\mathbf{v} = \mathbf{t}^* \times \mathbf{n}$ (this solution is defined as λ_2 and $\mathbf{v}_2$ in [49]). This is because this configuration results into $(\mathbf{H}^T\mathbf{H} - \lambda\mathbf{I}) = \mathbf{n}(\mathbf{t}^*)^T + \mathbf{t}^*\mathbf{n}^T + k^2\mathbf{n}\mathbf{n}^T$, which is a 3×3 matrix within the vector space of $\mathbf{t}^*$ and $\mathbf{n}$, while the vector $\mathbf{t}^* \times \mathbf{n}$ is orthogonal to this vector space. Based on this solution Zhang derived the following relations [49]:

$$\lambda_1 \geq \lambda_2 = 1 \geq \lambda_3$$
$$\|\mathbf{t}^*\| = \lambda_1 - \lambda_3, \qquad \mathbf{n}^T\mathbf{t}^* = \lambda_1\lambda_3 - 1$$

in addition to:

$$\mathbf{v}_1 \propto \quad \mathbf{v}_1^{'} = \zeta_1 \mathbf{t}^* + \mathbf{n} \tag{2.22a}$$

$$\mathbf{v}_2 \propto \quad \mathbf{v}_2^{'} = \mathbf{t}^* \times \mathbf{n} \tag{2.22b}$$

$$\mathbf{v}_3 \propto \quad \mathbf{v}_3^{'} = \zeta_3 \mathbf{t}^* + \mathbf{n} \tag{2.22c}$$

where $\mathbf{v}_i$ are the unit eigenvectors and $\zeta_{1,3}$ are scalar functions. Both $\mathbf{v}_i$ and $\zeta_{1,3}$ are derived from the eigenvalues of $\mathbf{H}^T\mathbf{H}$ according to [35]:

$$\zeta_{1,3} = \frac{1}{\lambda_1 \lambda_3}\left(-1 \pm \sqrt{1 + 4\frac{\lambda_1 \lambda_3}{(\lambda_1 - \lambda_3)^2}}\right) \tag{2.23}$$

and

$$\left\|\mathbf{v}_i^{'}\right\|^2 = \zeta_i^2(\lambda_1 - \lambda_3)^2 + 2\zeta_i(\lambda_1\lambda_3 - 1) + 1 \tag{2.24}$$

combining the results from 2.22a, 2.22c, 2.23 and 2.24 one obtain 4 solutions of $\mathbf{t}^*$, $\mathbf{n}$ and $\mathbf{R}$, where the first two are:

$$\mathbf{t}^* = \pm\frac{\mathbf{v}_1^{'} - \mathbf{v}_3^{'}}{\zeta_1 - \zeta_3}, \qquad \mathbf{n} = \pm\frac{\zeta_1\mathbf{v}_3^{'} - \mathbf{v}_1^{'}\zeta_3}{\zeta_1 - \zeta_3}$$

and the second two are:

$$\mathbf{t}^* = \pm\frac{\mathbf{v}_1^{'} + \mathbf{v}_3^{'}}{\zeta_1 - \zeta_3}, \qquad \mathbf{n} = \pm\frac{\zeta_1\mathbf{v}_3^{'} + \mathbf{v}_1^{'}\zeta_3}{\zeta_1 - \zeta_3}$$

The rotation matrix $\mathbf{R}$ can be derived by manipulating the expression of 2.20 and inserting a solution of $\mathbf{t}^*$ and $\mathbf{n}$.

A similar method that provides an analytical solution to the homography decomposition is proposed in [35]. This provides the same four solutions as Zhang's method and is used by the openCV library used in this thesis. Due to this method being more computational complex, Zhang's method has been explained here for conceptual purposes.

2.3 Relevant Probability Theory

In robotics, a common complication is the *inference problem*. Robots need to draw knowledge from their environment based on incoming sensor data as well as prior information. One of these inference problems is the SLAM problem, where one tries to estimate the location of a robot and map the environment it operates in. As the sensor data are uncertain measurements, one needs a probabilistic framework to draw better inferences about the world. One of these statistical frameworks used for solving the Bayesian inference problem is *factor graphs*. Important probabilistic concepts will now be explained, leading up to the explanation of factor graphs.

2.3.1 Multivariate Gaussian Distribution

In equation 2.25 one can see the equation of a multivariate Gaussian distribution of a random variable, $\mathbf{x}$. Here the variable $\boldsymbol{\mu}$ represents the expected value of the distribution, while $\mathbf{P}$ represents its covariance. The Gaussian distribution is one of the

most important types of *probability density functions* (PDF), and maps the probability density/ relative likelihood for any random variable $\mathbf{x}$. The Gaussian distribution is mainly described by its quadratic exponent, but since the probabilistic sum of all possible values of $\mathbf{x}$ needs to sum to 1, one needs to include the normalization constant $\frac{1}{(2\pi)^{\frac{n}{2}}|\mathbf{P}|^{\frac{1}{2}}}$ in equation 2.25.

$$\mathcal{N}(\mathbf{x};\boldsymbol{\mu},\mathbf{P}) = \frac{1}{(2\pi)^{\frac{n}{2}}|\mathbf{P}|^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\mathbf{P}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right) \tag{2.25}$$

Gaussian Linearity Theorem

Considering the quadratic exponent of the multivariate Gaussian, one can derive the expression of a Gaussian random variable that goes through a linear transformation. For example, suppose that one has the Gaussian distribution described in equation 2.25. One can then derive the distribution of the random variable $\mathbf{y} = \mathbf{F}\mathbf{x}$. For the general proof, one assumes that $\mathbf{F}$ is a positive definite matrix, meaning it is invertible.

$$\begin{aligned}
\mathcal{N}(\mathbf{x};\boldsymbol{\mu},\mathbf{P}) &\propto \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\mathbf{P}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right) \\
\mathcal{N}(\mathbf{y};...) &\propto \exp\left(-\frac{1}{2}(\mathbf{F}^{-1}\mathbf{y}-\boldsymbol{\mu})^T\mathbf{P}^{-1}(\mathbf{F}^{-1}\mathbf{y}-\boldsymbol{\mu})\right) \\
&= \exp\left(-\frac{1}{2}(\mathbf{F}^{-1}\mathbf{y}-\boldsymbol{\mu})^T\mathbf{F}^T(\mathbf{F}^T)^{-1}\mathbf{P}^{-1}(\mathbf{F}^{-1})\mathbf{F}(\mathbf{F}^{-1}\mathbf{y}-\boldsymbol{\mu})\right) \\
&= \exp\left(-\frac{1}{2}(\mathbf{y}-\mathbf{F}\boldsymbol{\mu})^T(\mathbf{F}^T)^{-1}\mathbf{P}^{-1}(\mathbf{F}^{-1})(\mathbf{y}-\mathbf{F}\boldsymbol{\mu})\right) \\
&= \exp\left(-\frac{1}{2}(\mathbf{y}-\mathbf{F}\boldsymbol{\mu})^T(\mathbf{F}\mathbf{P}\mathbf{F}^T)^{-1}(\mathbf{y}-\mathbf{F}\boldsymbol{\mu})\right)
\end{aligned} \tag{2.26}$$

From the equation 2.26 one arrive at the following theorem [5]:

Theorem 2.3.1 (Linearity) *If having a RV $\boldsymbol{x}$ given by the Gaussian distribution $\mathcal{N}(\boldsymbol{x};\boldsymbol{\mu},\boldsymbol{P})$, the distribution of a RV $\boldsymbol{y} = \boldsymbol{F}\boldsymbol{x}$ is then equal to $\mathcal{N}(\boldsymbol{y};\boldsymbol{F}\boldsymbol{\mu},\boldsymbol{F}\boldsymbol{P}\boldsymbol{F}^T)$*

2.3.2 The Maximum a Posteriori Estimator

In the SLAM problem one tries to estimate the unknown true *states* $\mathcal{X}$ from sensor measurements $\mathcal{Z}$. The state vector $\mathcal{X}$ is defined as $\mathcal{X} = [X, L]$ in the full SLAM problem, where X is a pose vector containing n poses: $X = [\mathbf{x}_0, \mathbf{x}_1, ..., \mathbf{x}_{n-1}, \mathbf{x}_n]^T$, and L is a landmark vector containing m landmarks: $L = [\mathbf{l}_1, \mathbf{l}_2, ..., \mathbf{l}_{m-1}, \mathbf{l}_m]^T$. In the pose SLAM problem however; $\mathcal{X} = X$. A sensor measurement is termed $\mathbf{z}_i$ and can come from various sensors such as lidar, dvl, tilt sensors, etc. All measurements received from the environment are corrupted by noise. A common way of modeling a measurement $\mathbf{z}_i$ is according to the *measurement model* shown in equation 2.27 where $h_i(\mathbf{x}_i) = \hat{\mathbf{z}}_i$ is the *measurement prediction function* of the true state $\mathbf{x}_i$ [44] and $\mathbf{w}_i$ is noise, assumed to have the characteristics of a zero-mean Gaussian noise model; $\mathbf{w}_k \sim \mathcal{N}(\mathbf{0}, \Sigma^w)$. Here Σ^w is the measurement covariance related to the sensor.

$$\mathbf{z}_i = h_i(\mathbf{x}_i) + \mathbf{w}_i \tag{2.27}$$

Based on the measurement function in equation 2.27 one can generate an error function for measurement i: $e_i(\mathbf{x}_i) = h_i(\mathbf{x}_i) - \mathbf{z}_i$. One can then select an objective

function that minimizes the squared error sum of all n errors, $e_i(\mathbf{x}_i)$. Here, the square function of $e_i(\mathbf{x}_i)$ is equal to the squared Mahalanobis distance [34]. Hence, one obtain the following non-linear least squares (LS) objective function:

$$\mathcal{X}^* = \arg\min_{\mathcal{X}} f(\mathcal{X}) = \arg\min_{\mathcal{X}} \sum_{i}^{n} \|h_i(\mathbf{x}_i) - \mathbf{z}_i\|^2_{\Sigma_i} \tag{2.28}$$

The solution of equation 2.28 proves to be the *maximum a posteriori* (MAP) estimate [9]. The MAP estimator tries to maximize $p(\mathcal{X}|\mathcal{Z})$, the *posteriori density function*, with respect to $\mathcal{X}$ given measurements $\mathcal{Z}$ as shown in equation 2.29. Equation 2.29 can be rewritten to 2.30 by utilizing Bayes' rule.

$$\mathcal{X}^{MAP} = \arg\max_{\mathcal{X}} p(\mathcal{X}|\mathcal{Z}) \tag{2.29}$$

$$= \arg\max_{\mathcal{X}} \frac{p(\mathcal{Z}|\mathcal{X})p(\mathcal{X})}{p(\mathcal{Z})} \tag{2.30}$$

In equation 2.30 the *marginal* $p(Z)$ is a normalization factor, and is therefore irrelevant the posteriori maximization. Hence, one can instead use a proportional expression of 2.30 for finding the MAP according to equation 2.31 [9]

$$\mathcal{X}^{MAP} = \arg\max_{\mathcal{X}} l(\mathcal{X}; \mathcal{Z})p(\mathcal{X}) \tag{2.31}$$

where $l(\mathcal{X}; \mathcal{Z})$ is the *likelihood function* of $\mathcal{X}$ given $\mathcal{Z}$ which is proportional to $p(\mathcal{Z}|\mathcal{X})$; $l(\mathcal{X}; \mathcal{Z}) \propto p(\mathcal{Z}|\mathcal{X})$

2.3.3 Factor Graph for Inference

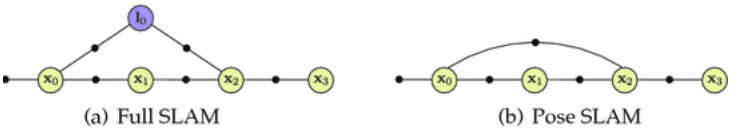


FIGURE 2.7: Illustration of the factor graph of full SLAM problem in (a), and (b) illustrated the factor graph of the pose SLAM problem. Courtesy of [16]

The SLAM problem can be represented graphically by utilizing a *factor graph*. A factor graph is a probabilistic model and can be drawn as in figure 2.7 (a) and (b) for both the full SLAM problem and the *pose* SLAM problem respectively. A factor graph is a bipartite graph consisting of *nodes* and *factors* [16]. Nodes indicates variables one are interested in estimating (e.g. poses and landmarks), while factors are probabilistic constraints between nodes obtained from various noisy measurements. As the factors, $\mathbf{\Psi}_i(\mathbf{x}_i, \mathbf{l}_i)$, are the only component that constraints the pose and landmarks, one can find the *MAP* solution of the non-linear optimization problem by minimizing:

$$\underbrace{X^*, L^*}_{\mathcal{X}^*} = \arg\min_{X,L} \sum_{i} \mathbf{\Psi}_i(\mathbf{x}_i, \mathbf{l}_i) = \arg\max_{X,L} p(X, L|\mathcal{U}, \mathcal{Z}) \tag{2.32}$$

Here $\mathcal{U}$ is a vector of all odometry measurements. As we are more interested in the pose SLAM in this work, equation 2.32 can be rewritten accordingly:

$$\mathcal{X}^* = \arg\min_{\mathcal{X}} \sum_i \Psi_i(\mathbf{x}_i) = \arg\max_{\mathcal{X}} p(\mathcal{X}|\mathcal{U}, \mathcal{Z}) \tag{2.33}$$

where $\mathcal{X} = X$. In pose SLAM landmarks are not a part of the state vector $\mathcal{X}$, but environmental features are used to obtain relative pose measurements between poses as in *visual odometry* (VO).

Assuming that all measurements can be modeled by a multivariate Gaussian distribution (see equation 2.25) and are independent, one can find the MAP of the joint probability distribution of all measurements by rewriting equation 2.33 to:

$$\begin{aligned}\mathcal{X}^* &= \arg\min_{\mathcal{X}} -\log p(\mathcal{X}|\mathcal{U}, \mathcal{Z}) \\ &= \arg\min_{\mathcal{X}} \left[\sum_i \|\mathbf{x}_i - \mathbf{f}_i(\mathbf{x}_{i-1}, \mathbf{u}_{i-1})\|^2_{\Sigma_i^v} + \sum_j \left\|\mathbf{z}_j - \mathbf{h}_j(\mathbf{x}_{i_j})\right\|^2_{\Sigma_j^w} \right]\end{aligned} \tag{2.34}$$

where $\mathbf{f}_i$ is the motion model and Σ_i^v and Σ_j^w are the covariance of motion- and measurement model respectively. Notice how the formulation in 2.34 enables effortless addition of new sensor measurements to the pose graph. Another important point to note is that the error functions for both motion- and measurement model are described by the Mahalanobis distance. However, the Mahalanobis distance can be reformulated to become a euclidean norm expression by doing the following manipulation:

$$\|\mathbf{e}(\mathbf{x})\|^2_{\Omega_i} = \mathbf{e}(\mathbf{x})^T \Omega^{-1} \mathbf{e}(\mathbf{x}) = (\Omega^{-1/2}\mathbf{e}(\mathbf{x}))^T (\Omega^{-1/2}\mathbf{e}(\mathbf{x})) = \left\|\Omega^{-1/2}\mathbf{e}(\mathbf{x})\right\|^2$$

The next section will clarify why it is convenient to have the problem formulated in this way.

2.4 Non-Linear Optimization

SLAM systems are often separated into a *front-end* system and a *back-end* system. The front-end system is responsible constructing the optimization problem (equation 2.34), while the back-end handles the optimization. When having an objective function $\mathbf{f}(\mathbf{x}) = \arg\min_{\mathbf{x}} \|\mathbf{e}(\mathbf{x})\|^2$ where $\mathbf{e}(\mathbf{x})$ is a non-linear function, this requires non-linear solvers for obtaining the optimal solution. Two common non-linear solvers are the Gauss-Newton and the Levenberg-Marquardt algorithm. These are both iterative solvers, which requires an initial estimate to converge towards a *local* minimum. Both algorithm follow the general framework given below [18]:

1. Select initial estimate; $\hat{\mathbf{x}}^0$
2. Linearize $\mathbf{e}(\mathbf{x})$ about the current estimate $\hat{\mathbf{x}}^t$
3. Find increment by solving the linearized problem
4. Update estimate with found increment

Here step 2-4 are repeated until the solution converges or one are left with a satisfying error $\mathbf{e}(\mathbf{x})$. The linearization step of a non-linear function $\mathbf{e}(\mathbf{x})$ can be achieved by using a first order Taylor expansion about a current estimate $\hat{\mathbf{x}}^t$ as follows:

$$\mathbf{e}(\mathbf{x}) = \mathbf{e}(\hat{\mathbf{x}}^t + \Delta\mathbf{x}) \approx \mathbf{e}(\hat{\mathbf{x}}^t) + \mathbf{J}^{\mathbf{e}}_{\hat{\mathbf{x}}^t} \Delta\mathbf{x} \tag{2.35}$$

where $\mathbf{J}^{\mathbf{e}}_{\hat{\mathbf{x}}^t}$ is the Jacobian matrix of $\mathbf{e}(\mathbf{x})$ at the $\hat{\mathbf{x}}^t$, and $\Delta\mathbf{x} = \hat{\mathbf{x}} - \hat{\mathbf{x}}^t$. From the expression in equation 2.35 one can approximate the non-linear LS problem to:

$$\begin{aligned}\mathbf{f}(\mathbf{x}) &= \arg\min_{\mathbf{x}} \|\mathbf{e}(\mathbf{x})\|^2 \\ &\approx \arg\min_{\mathbf{x}} \left\|\mathbf{J}^{\mathbf{e}}_{\hat{\mathbf{x}}^t}\Delta\mathbf{x} + \mathbf{e}(\hat{\mathbf{x}}^t)\right\|^2 = \arg\min_{\mathbf{x}} \left\|\mathbf{J}^{\mathbf{e}}_{\hat{\mathbf{x}}^t}\Delta\mathbf{x} - (-\mathbf{e}(\hat{\mathbf{x}}^t))\right\|^2 \\ &= \arg\min_{\mathbf{x}} \|\mathbf{A}\Delta\mathbf{x} - \mathbf{b}\|^2 \end{aligned} \tag{2.36}$$

The solution to 2.36 can be found by taking the derivative of the expression and setting it to zero. One would the obtain the expression;

$$\Delta\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b} \tag{2.37}$$

where the expression $(\mathbf{A}^T\mathbf{A})$ can be found to be an approximation of the Hessian of $\mathbf{f}(\mathbf{x})$ at current estimate $\hat{\mathbf{x}}^t$ [18]. Equation 2.37 can be solved by *QR* factorization or Cholesky factorization. Based on the solution of the linearized problem 2.37 one can update the current estimate for the non-linear problem and repeat the procedure around the new state estimate.

2.4.1 The Gauss- Newton Algorithm

This work has utilized the Levenberg-Marquardt optimizer for finding the MAP solution to the non-linear LS problem obtained from the incremented factor graph. However, this method is based on the Gauss-Newton method, and one therefore needs to understand this algorithm first. The Gauss-Newton procedure is shown in algorithm 1. As for line 2-4 these are the same steps explained in the previous section, while line 5-7 are the stop criteria.

Algorithm 1 Gauss-Newton Algorithm [18]

Require: Objective function $\mathbf{f}(\mathbf{x})$ and an initial estimate $\hat{\mathbf{x}}^0$
1: **for** $i = 0$ to t^{max} **do**
2: $\mathbf{A}$, $\mathbf{b} \leftarrow$ Linearize $\mathbf{f}(\mathbf{x})$ about $\hat{\mathbf{x}}^t$
3: $\Delta\mathbf{x} \leftarrow$ By solving the equation $\Delta\mathbf{x} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$
4: $\hat{\mathbf{x}}^{t+1} \leftarrow \hat{\mathbf{x}}^t + \Delta\mathbf{x}$
5: **if** $\mathbf{f}(\hat{\mathbf{x}}^{t+1}) < \mathbf{f}(\mathbf{x})_{th}$ or $\hat{\mathbf{x}}^{t+1} \approx \hat{\mathbf{x}}^t$ **then**
6: $\hat{\mathbf{x}} \leftarrow \hat{\mathbf{x}}^{t+1}$ ▷ Estimated solution to the non-linear LS problem
7: **return** $\hat{\mathbf{x}}$
8: **end if**
9: **end for**

2.4.2 The Levenberg-Marquardt Algorithm

As stated previously, the expression $(\mathbf{A}^T\mathbf{A})$, which is used in the Gauss-Newton method, is an approximation of the Hessian matrix of $\mathbf{f}(\mathbf{x})$ at current estimate $\hat{\mathbf{x}}^t$. Because the Gauss-Newton method is using this approximation it is not guaranteed to converge towards a solution [18]. Hence, a way of handling this is to use *trust regions* that determines how confident one is to trust the approximated Hessian. Such trust regions are used in the Levenberg-Marquardt algorithm, and the modification to the

solution of $\Delta\mathbf{x}$ is done accordingly:

$$\Delta\mathbf{x} = (\mathbf{A}^T\mathbf{A} + \lambda diag(\mathbf{A}^T\mathbf{A}))^{-1}\mathbf{A}^T\mathbf{b} \tag{2.38}$$

Based on the addition of the increment $\Delta\mathbf{x}$ is contributing to a convergence towards the solution or not, the trust region is either increased or decreased by the parameter λ. The Levenberg-Marquardt algorithm is shown in algorithm 2.

Algorithm 2 Levenberg-Marquardt Algorithm [18]

Require: Objective function $\mathbf{f}(\mathbf{x})$ and an initial estimate $\hat{\mathbf{x}}^0$

$\lambda \leftarrow 10^{-4}$
for $i = 0$ to t^{max} **do**
  $\mathbf{A}$, $\mathbf{b} \leftarrow$ Linearize $\mathbf{f}(\mathbf{x})$ about $\hat{\mathbf{x}}^t$
  $\Delta\mathbf{x} \leftarrow$ By solving the equation $\Delta\mathbf{x} = (\mathbf{A}^T\mathbf{A} + \lambda diag(\mathbf{A}^T\mathbf{A}))^{-1}\mathbf{A}^T\mathbf{b}$
  if $\mathbf{f}(\hat{\mathbf{x}}^{t+1}) < \mathbf{f}(\hat{\mathbf{x}}^t)$ **then**
    $\hat{\mathbf{x}}^{t+1} \leftarrow \hat{\mathbf{x}}^t + \Delta\mathbf{x}$ ▷ Accept update
    $\lambda \leftarrow \lambda/10$ ▷ Increase trust region
  else
    $\hat{\mathbf{x}}^{t+1} \leftarrow \hat{\mathbf{x}}^t$ ▷ Decline update
    $\lambda \leftarrow \lambda * 10$ ▷ Decrease trust region
  end if
  if $\mathbf{f}(\hat{\mathbf{x}^{t+1}}) < \mathbf{f}(\mathbf{x})_{th}$ or $\hat{\mathbf{x}}^{t+1} \approx \hat{\mathbf{x}}^t$ **then**
    $\hat{\mathbf{x}} \leftarrow \hat{\mathbf{x}}^{t+1}$ ▷ Estimated solution to the non-linear LS problem
    return $\hat{\mathbf{x}}$
  end if
end for

2.5 Incremental Smoothing and Mapping 2

As explained in section 2.3.3 and 2.4 a factor graph can be optimized by solving the non-linear problem shown in equation 2.32. The MAP solution is found by continues linearization until convergence. However, the SLAM problem is an *incremental* inference problem, as new measurements arrive in a temporal sequence. A naive approach for solving this problem would be to solve the entire LS-problem for each new measurement, but this would require too much computational power in a real-time system. The iSAM2 (incremental smoothing and mapping 2), predecessor of iSAM, is a SOTA solution to this issue. The iSAM2 algorithm are able to update MAP estimate by utilizing the previous MAP estimate; allowing for a computationally faster calculation. This incremental update is made possible by structuring measurements into *cliques* in a Bayes Net, structuring this Bayes Net into a Bayes tree, and doing incremental updates of the MAP solution through fluid relinearization. The iSAM2 procedure is summarized in algorithm 3

Bayes Tree

The Bayes tree, seen in figure 2.8 (c), is a graphical model used as a framework in the iSAM2 model to handle incremental MAP updates efficiently. This is based on rewriting of a Bayes Net (a transformation of a factor graph), seen in figure 2.8 (b), which is possible due to the *chordal* property of the Bayes Net. The chordal property

Algorithm 3 iSAM2 Algorithm for adding new measurements and updating MAP estimate [44]

Require: Bayes tree $\mathcal{T}$, factors $\mathcal{F}$, linearization point from previous estimate Θ, previous update Δ, new factors $\mathcal{F}'$, new initial estimate Θ'

1: Add new factor to factor graph $\mathcal{F} = \mathcal{F} \cup \mathcal{F}'$
2: Add initial estimate to linearization point $\Theta = \Theta \cup \Theta'$
3: Fluid relinearization of all affected cliques $\mathcal{J}$
4: Update Bayes tree $\mathcal{T}$
5: Find MAP incremental update Δ
6: Increment previous MAP estimate $\Theta \oplus \Delta$

is any undirected cycle of length greater than three [9]. Such an undirected cycle is apparent in figure 2.8 (b). The chordal property allow for identification of groups of interconnected nodes, *cliques*. Cliques are the foundation of the Bayes tree as its tree nodes are represented by the conditional probability distribution to the variables eliminated in cliques [24]. The joint probability density of the Bayes tree, $p(X)$, can be found by taking the product of all the conditional density $p(F_k|S_k)$ provided by the cliques. Here S_k is the separator of the clique and its parents node, while the frontal variables F_k are the remaining variables [9]. This can be described mathematically as follows:

$$p(X) = \prod_k p(F_k|S_k) \tag{2.39}$$

Updating the Bayes Tree

When a new measurement arrives, one need to add the respective factor into the Bayes Tree. An example of such a factor is a factor depending on two variables $\Psi_(x_i, x_j)$. When adding such a factor only the paths from the root to these cliques needs to be changed, as these are the only probabilities affected by the addition of the factor. An example of adding a factor $\Psi_(x_3, x_1)$ to the Bayes tree is shown graphically in figure 2.9.

Fluid Relinearization

As an addition of a new measurement to a clique will *only* affect the conditional probabilities of cliques between the updated clique and the root, there needs to be done a relinearization of these respective nodes affected by the new measurement. All the respective nodes from the Bayes Tree needs to then be removed, relinearized (if needed) and put back into the Bayes Tree. This relinearized is done by the fluid relinearization algorithm shown in algorithm 4. A variable is only relinearized if its current state update vector Δ is larger than some threshold β as stated in line 1.

Algorithm 4 Fluid relinearization [24]

Require: Current linearization point Θ and current Δ

1: Mark variables in Δ above some threshold : $J = \{\Delta_j \in \Delta | \Delta_j \geq \beta\}$
2: Update marked values $\Theta_J = \Theta_J \oplus \Delta_J$
3: Mark all cliques M that involve the marked variables Θ_j and all their ancestors
4: **return** M and Θ_J

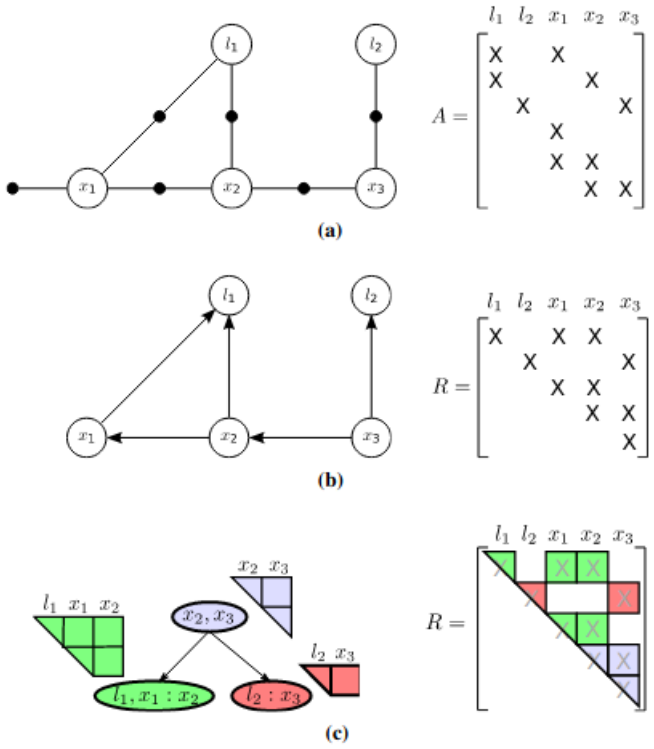


FIGURE 2.8: The factor graph and the associated Jacobian matrix A for a small SLAM example, where a robot located at successive poses x1, x2, and x3 makes observations on landmarks l1 and l2. In addition there is an absolute measurement on the pose x1. (b) The chordal Bayes net and the associated square root information matrix R resulting from eliminating the factor graph using the elimination ordering l1, l2, x1, x2, x3. The last variable to be eliminated, here x3, is called the root. (c) The Bayes tree and the associ- ated square root information matrix R describing the clique structure in the chordal Bayes net. A Bayes tree is similar to a junction tree, but is better at capturing the formal equiv- alence between sparse linear algebra and inference in graphical models. The association of cliques and their conditional densities with rows in the R factor is indicated by color. Courtesy of [24]

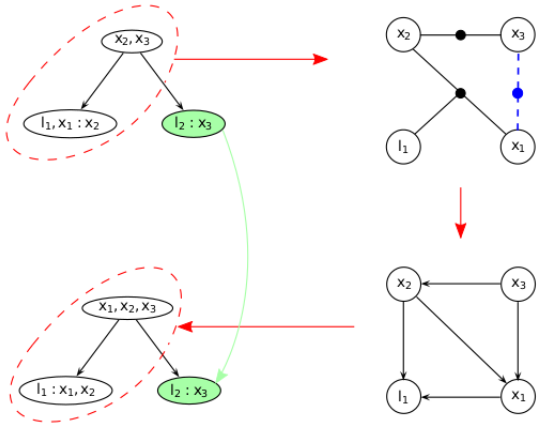


FIGURE 2.9: Updating a Bayes tree with a new factor, based on the example in figure 2.8. The affected part of the Bayes tree is highlighted for the case of adding a new factor between x1 and x3 . Note that the right branch (green) is not affected by the change. (top right) The factor graph generated from the affected part of the Bayes tree with the new factor (dashed blue) inserted. (bottom right) The chordal Bayes net resulting from eliminating the factor graph. (bottom left) The Bayes tree created from the chordal Bayes net, with the unmodified right "orphan" sub-tree from the original Bayes tree added back in. Courtesy of [9]

Chapter 3

Literature

This chapter will review relevant field work and studies that has inspired the method proposed for solving the SLAM problem in the aquaculture net pen environment. When writing this thesis, there has been studied several other papers. However, as many of these solutions use expensive imaging sonars or propose a not transferable solution to the aquaculture environment, these have not been mentioned in this chapter.

3.1 Experimental Evaluation of Hydroacoustic Instruments for ROV Navigation Along Aquaculture Net Pens

Motivation: *This article has been included in the literature as it is a critical study showing that both the DVL and USBL can be used inside a fish cage. In addition, this article also presents an average positional error estimate of the USBL measurements, which has been used for determining the USBL measurement uncertainty in this thesis.*

In this article, Rundtop and Frank do an experimental study of a DVL and USBLused on an ROV inside a fish cage. The experiments were conducted on a salmon farm under calm weather conditions where the fish cage had an upper diameter of 15m and was 25m deep. Nylon thread was the fabric used for the mesh grid, and the mesh had a twine diameter of 2-3mm. The equipment used was a Teledyne Workhorse Navigator DVL, which was mounted on the frontal side of ROV and pointed towards the net, and a Scout plus USBL from Sonardyne where the transceiver is connected to a surface vessel outside the fish cage while the transponder is attached to the ROV. The DVL was operated at a frequency of 1200kHz, a frequency high enough to cause reflection of the DVL beams from the net pen [40], allowing for velocity and range measurements. The USBL, on the other hand, operated at a low frequency between 35 and 55kHz. The low operating frequency has the opposite effect and allows communication between the transmitter and transponder located at each side of the net.

The research aimed to determine if the acoustic equipment was reliable to use in a fish cage in its peak production stage. A single net pen can hold up to 200 000 fish at peak production, corresponding to 800 000tons [40]. This amount of fish can cause challenges in the acoustic world, as the air-filled swim-bladders of the fish can cause acoustic scattering [15]. When the experiments were conducted, the cage contained approximately 170 000 fish, and halocline occurrences, which cause acoustic refraction, were unusual in the area.

Despite the fact that USBL and DVL were subjected to a challenging environment, the results from the experiments were positive. Open water studies were conducted for the USBL and recorded a positional standard deviation to be 1.1m, 0.7m, and 0.3m for the respective NED coordinates. USBL experiments were also conducted when the fish cage was filled with fish. These results are shown in figure 3.1. In this study, the transponder was placed at equally spaced position along the circumference of the net pen, ranging from 0 to 180 degrees. This experiment was conducted for three different depths: 2m, 8m, and 15m. From figure 3.1 it is evident that accuracy decreases as the transponder is placed at 54° or more. This caused by a combination of longer travel time in addition to fish bladder scattering the acoustic signal. Another observation regarding this study was that there were generally longer intervals between samples when the transponder was set at more than 54°; ranging from 1s to 2s.

Similar positive results was seen for the DVL. The DVL proved to be effective at tracking a relative distances of 1.5m to 3.5m between the ROV and the net with low noise. However, the DVL was not effective when fish obstructed the travel path of the DVL beam. The DVL was also used to estimate net relative yaw angle and velocity of the ROV, and these estimates were effected in the similarly.

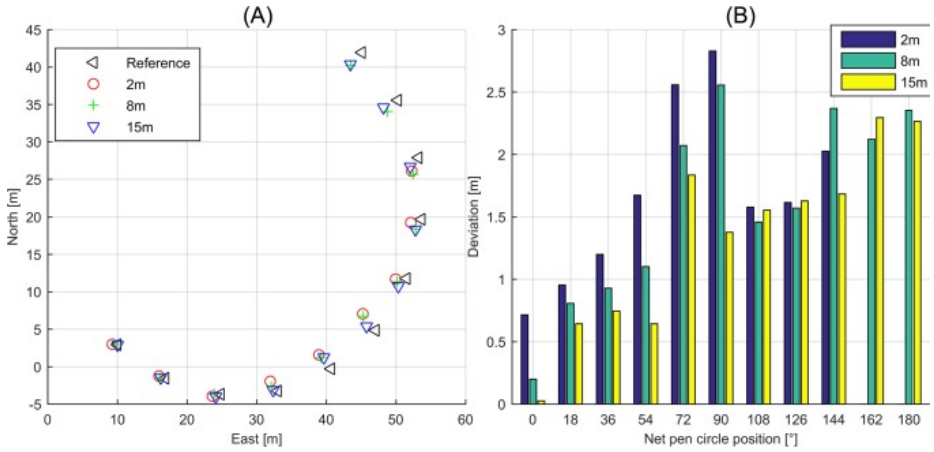


FIGURE 3.1: The figure to the left shows the average position estimated by the USBL in the horizontal plane after 3 minutes of sampling. The right figure illustrates the horizontal deviation from the transponder reference position. Courtesy of [40]

3.2 Autonomous ROV Inspections of Aquaculture Net Pens Using DVL

Motivation: *Findings in this paper are summarized as it proposes a method for approximating the net pen as a plane. The approximation method has been used in the SLAM algorithm proposed in this thesis for scaling and determining the correct homography decomposition when doing loop closures.*

In this paper, Amundsen and his team are developing a line-of-sight (LOS) guiding method for an ROV to enable autonomous traversing of the net pen using only the DVL. The Argus Mini ROV was also used in this work, where the DVL was mounted similarly to experiments conducted in this thesis. Instrumentation and control systems are explained in detail in section 1.3.1 and 1.3.1. The solution represented was based on four steps [2]:

- Estimate the geometry of a local region in front of the ROV. This was done by plane approximation based on DVL readings, using the same method seen in [10].
- Estimate the distance and yaw angle of the ROV relative to the approximated plane.
- Use the estimated relative yaw angle to control the ROV such that it is facing the plane.
- Apply LOS guidance and velocity control to allow for traversing.

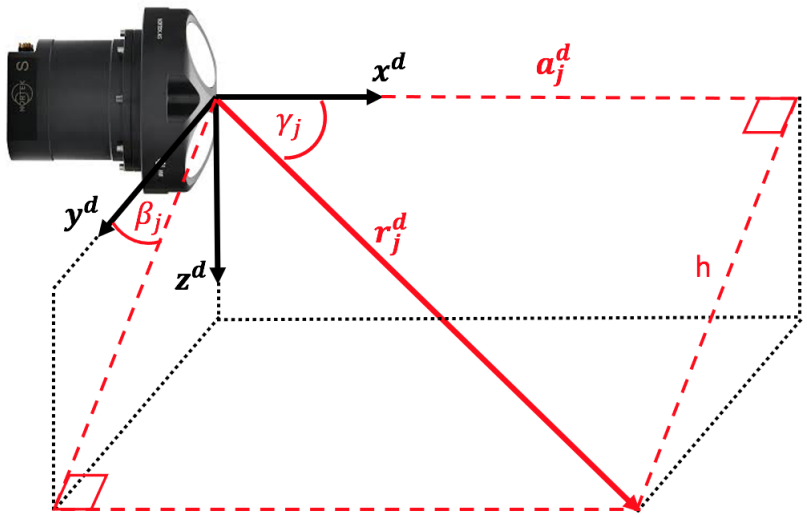


FIGURE 3.2: Illustrating the jth beam vector with the DVL pointing downwards. Inspired by [10]

This article review will only explain the first three steps as the last is not relevant for this thesis. Before proceeding with explaining the plane approximation method, it is important to know how the DVL sensor readings are converted to range measurements. By looking at figure 3.2, one can see how the j-th beam vector is projected. This beam vector can be calculated according to equation 3.1

$$r_j^d = \begin{bmatrix} x_j^d \\ y_j^d \\ z_j^d \end{bmatrix} = \begin{bmatrix} a_j^d \\ h\cos(\beta_j) \\ h\sin(\beta_j) \end{bmatrix} = a_j^d \begin{bmatrix} 1 \\ \tan(\gamma_j)\cos(\beta_j) \\ \tan(\gamma_j)\sin(\beta_j) \end{bmatrix} \tag{3.1}$$

Here a_j^d is the DVL reading of beam j, and β_j and γ_j are angles related to the static beam orientation with respect to the y^d- and x^d-axis, respectively. The four DVL beam range measurements can be transformed from DVL-frame, d, to the body-frame, b, by doing the calculation shown in equation 3.2. Further, the transformation from body-frame to NED-frame n is shown in equation 3.3.

$$\mathbf{r}_j^b = \mathbf{R}_d^b(\mathbf{\Theta}_{bd})\mathbf{r}_j^d + \mathbf{r}_{bd}^b \tag{3.2}$$

$$\mathbf{r}_j^n = \mathbf{R}_b^n(\mathbf{\Theta}_{nb})\mathbf{r}_j^b + \mathbf{x}_{nb}^n \tag{3.3}$$

Here $\mathbf{R}_d^b$ is the rotation matrix from DVL-frame to body-frame, and Θ_{bd} is the relative orientation of the DVL-frame with respect to the body-frame (equal to the identity matrix for the Argus Mini ROV). $\mathbf{R}_b^n$ is the rotation matrix from body-frame to NED-frame, and Θ_{nb} is the heading of the ROV with respect to the NED-frame, also referred to as yaw, ψ.

The plane equation is shown in 3.4.

$$f(x, y, z) = -x + by + cz + d = 0 \tag{3.4}$$

As this consist of three unknown variables, b, c and d, one only needs three valid beam readings to approximate a plane. However, if four beams are available, one can use a least square regression to find the best fitting plane related to the measurements. The objective function to minimize is shown in equation 3.5.

$$\sum_{j=1}^{4}[a_j^d - (by_j^d + cz_j^d + d)]^2 \tag{3.5}$$

Writing equation 3.5 on matrix form, one gets

$$\underbrace{\begin{bmatrix} y_1^d & z_1^d & 1 \\ y_2^d & z_2^d & 1 \\ y_3^d & z_3^d & 1 \\ y_4^d & z_4^d & 1 \end{bmatrix} \begin{bmatrix} b \\ c \\ d \end{bmatrix}}_{\mathbf{Ax}} = \underbrace{\begin{bmatrix} a_1^d \\ a_2^d \\ a_3^d \\ a_4^d \end{bmatrix}}_{\mathbf{b}} \tag{3.6}$$

The solution to 3.6 is found by solving the normal system $\mathbf{A}^T\mathbf{Ax} = \mathbf{A}^T\mathbf{b}$ [12]. It is also worth noting that this regression method can help filter out the noise related to the DVL [2].

Calculation of Desired Heading

For the ROV to be pointed directly towards the approximated plane, the heading of the ROV needs to be aligned with the normal vector to the plane, f (see equation

3.4). This normal vector in the DVL-frame, $\{d\}$, can be written as follows [2]:

$$\mathbf{n}^d = \begin{bmatrix} -1 \\ b \\ c \end{bmatrix} \tag{3.7}$$

The normal vector in equation 3.7 can be converted to the NED-frame n by doing the following rotational transformation:

$$\mathbf{n}^n = \mathbf{R}_b^n(\psi)(\mathbf{R}_b^d(\Theta_{db}))^T\mathbf{n}^d \tag{3.8}$$

The vector $\mathbf{n}^n$ is a 3D-vector, and needs to be projected onto the 2D-North-East-plane in order to acquire the desired heading. This projection is done in the following manner according to [2]

$$\mathbf{n}^n_{projection} = \begin{bmatrix} x^n_{projection} \\ y^n_{projection} \\ 0 \end{bmatrix} = -(\mathbf{z}^n \times \mathbf{n}^n \times \mathbf{z}^n) \tag{3.9}$$

where $\mathbf{z}^n = [0, 0, 1]^T$. The desired heading which allows the ROV to be pointing towards the net pen can then be calculated using basic trigonometry:

$$\psi_d = \arctan 2(y^n_{projection}, x^n_{projection}) \tag{3.10}$$

ROV Distance Relative to the Approximated Plane

The normal vector of the plane with respect to the DVL-frame is shown in equation 3.7 and can be transformed to its respective unit vector by doing the following manipulation:

$$\mathbf{n}^d_{unit} = \frac{\mathbf{n}^d}{||\mathbf{n}^d||_2} = \frac{1}{\sqrt{1+b^2+c^2}} \begin{bmatrix} -1 \\ b \\ c \end{bmatrix}$$

As the shortest distance from a plane to a point $\mathbf{p}^*$ will be perpendicular to the plane, one can use the property of the dot product to find the distance to a plane. Since one is interested in determining the distance from the body to the plane one will start of describing a vector going from a random point on the plane $\mathbf{p}^d_0$ to the body coordinate, which can be defined as follows:

$$\mathbf{v}^d = \mathbf{r}^d_{db} + \mathbf{p}^d_0 = \begin{bmatrix} x^d_{db} + x^d_0 \\ y^d_{db} + y^d_0 \\ z^d_{db} + z^d_0 \end{bmatrix}$$

Note that $\mathbf{v}^d$ is described in the DVL-frame. By taking the dot product between $\mathbf{v}^d$ and $\mathbf{n}^d_{unit}$ one will project $\mathbf{v}^d$ onto $\mathbf{n}^d_{unit}$. Hence, by the absolute value of this dot product will correspond to the distance between the body and the plane:

$$d_{b/net} = |(\mathbf{v}^d)^T\mathbf{n}^d_{unit}| = \frac{|-x^d_{bd} + by^d_{bd} + cz^d_{bd} - d|}{\sqrt{1+b^2+c^2}}$$

3.3 Real-Time Visual SLAM for Autonomous Underwater Hull Inspection Using Visual Saliency

Motivation: *This article is added into the literature chapter as this article is the main inspiration for this thesis. The article shows how measurements can be added to a pose graph and solved by the iSAM framework. It also provides a measure for determining intra- and inter-image saliency, which this thesis uses for its data association filtering algorithm.*

In this work, Ayoung and Eustice created a visual SLAM algorithm for ship hull inspection purposes. The proposed method was proven to handle limited field of view imagery while also handling feature-sparse regions. This was accomplished by using the iSAM-framwork to create a six-DOF pose-graph and only consider imagery considered to be visually salient. In addition, an online bag-of-words algorithm was created to determine the inter and intra saliency of an image.

A visual representation of the pose-graph used in this work can be seen in figure 3.3. Here each pose node, $\mathbf{x}_i$, contains a stored image taken at time i. The absolute measurements constraining the nodes are gathered from roll/pitch and depth measurements, odometry was captured by a DVL, while the camera factor was obtained from a five DOF pair-wise camera measurements.

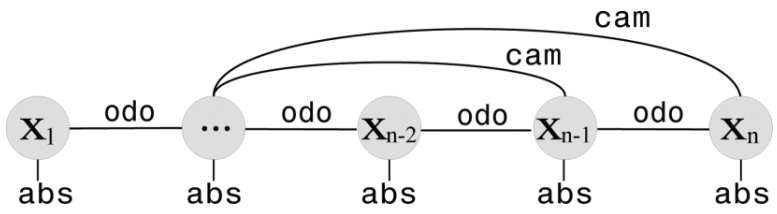


FIGURE 3.3: Shows the pose-graph structure used in Ayoung- and Eustice's work. Odo and abs referes to *odometry* and *absolute* measurements for roll/pitch and depth, respectively. Cam is camera constraints. Courtesy of [27]

Camera Constraints

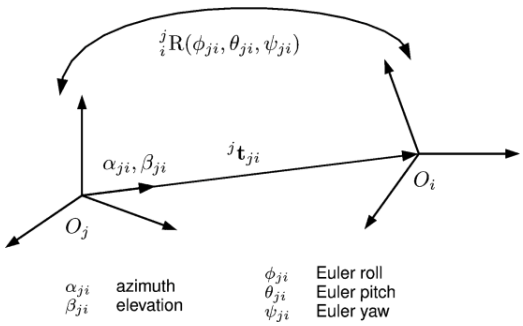


FIGURE 3.4: Shows the pose-graph structure used in Ayoung- and Eustice's work. Odo and abs referes to *odometry* and *absolute* measurements for roll/pitch and depth, respectively. Cam is camera constraints. Courtesy of [27]

The mono camera had a fixed bearing pointing nadir towards the ship's hull surface. Due to this constraint, one only needs a five-DOF measurement model for the camera. This five-DOF model is obtained from image pairing, and models the relative baseline direction of motion azimuth, α_{ij}, elevation angle, β_{ij} and the relative Euler angles between keyframes; ϕ_{ij}, θ_{ij} and ψ_{ij} [27]. The model is illustrated in figure 3.4 and mathematically in 3.11.

$$\mathbf{h}_{cam}(\mathbf{x}_i, \mathbf{x}_j) = [\alpha_{ij}, \beta_{ij}, \phi ij, \theta ij, \psi_{ij}] \tag{3.11}$$

The image features and camera constraint are obtained by using a camera-client processing pipeline which is described as follows [27]:

(a) The image is radially undistorted and features are enhanced by using contrast-limited adaptive histogram specification (CLAHS).

(b) SURF is used to extract image features which is processed using GPU power for real-time performance.

(c) A pose-constrained correspondence search (PCCS) together with RANSAC is used as a geometric model selection framework [26].

(d) The inliers found by the previous step are then feed into a two-view bundle adjustment algorithm to acquire a 5 DOF camera measurement as well as its corresponding covariance [19].

(e) The 5-DOF camera measurement is added as a factor constraint to the iSAM framework.

Figure 3.5 shows three possible cases of the camera-client processing pipeline. The first case is related to poor features, but there is a strong SLAM prior. Then the PCCS provides a minimal search region for feature matching that allows for successful feature matching despite the scene being feature-poor. The middle case shows the worst-case scenario where one has not good features nor a good prior, causing the image matching to fail. The final case considers a feature-rich image with a weak SLAM prior. Due to the feature rich scene, one is also able to obtain a camera constraint like in the first case.

Bag-of-Words

To distinguish between salient and non-salient frames, the paper proposed two types of salient measures; local saliency (intraimage) and global saliency (interimage) [27], where both are computed using an online BOW algorithm. The online BOW algorithm used 128-dimension SURF descriptors, which were extracted from a pre-blurred image (in addition to applying the CLAHS algorithm). Pre-blurring was applied to enable better generalization by forcing the SURF-algorithm to search for large-scale features and remove particle noise. The BOW-algorithm used assumed no prior appearance knowledge about the environment, and the vocabulary was therefore set to be empty at initialized. During an inspection, the vocabulary size was augmented as new words were discovered. SURF- features were assigned to respective words by evaluating the Euclidean inner product to existing words in the vocabulary, and new vocabulary words were assigned if the direction cosine of a discovered word to existing vocabulary was more significant than 0.4.

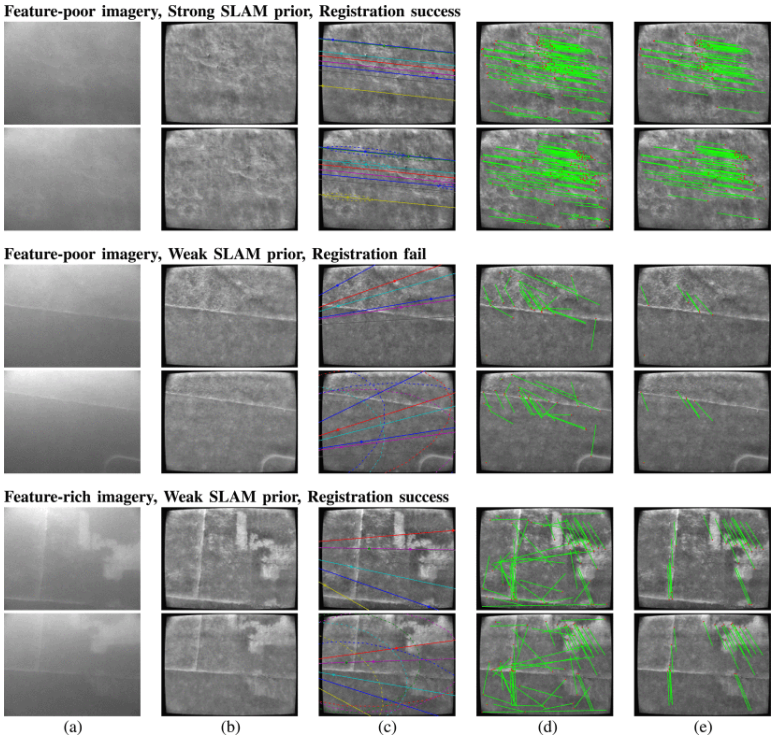


FIGURE 3.5: This image shows three cases of the camera-client processing pipeline. Here the columns (a) to (e) are as follows: (a) the raw image input, (b) CLAHS enhanced image, (c) PCCS search regions, (d) Putative matches and (e) inliers found by RANSAC. Courtesy of [14]

Local Saliency

The word diversity within a keyframe i, $\mathcal{K}_i$, was found by calculating the entropy of its BOW histogram which is calculated according to equation 3.12:

$$H_i = -\sum_{k}^{W(t)} p(w_k)\log_2(p(w_k)) \tag{3.12}$$

Here, $p(w_k)$ is the *term frequency* (TF) of a word w_k within image i, which can be described mathematically as $\mathcal{W}(t) = \{w_k\}_{k=1}^{W(t)}$, where $\mathcal{W}(t)$ is the size of the online vocabulary at time t. The entropy calculation is normalized, and the paper refer to this as the local saliency of the image. This normalization is done accordingly:

$$S_{L_i} = \frac{H_i}{\log_2 \mathcal{W}(t)}$$

Here $\log_2 \mathcal{W}(t)$ is the maximum entropy possible, which is achieved by a uniform word distribution in an image across the entire vocabulary[27]. S_{L_i} gives a value between zero and one, determining how feature-rich an image is considered.

Global Saliency

The global saliency was as an interimage measure for determining uniqueness. The purpose of the global saliency is to detect unique pictures that can be used for large-scale loop-closure. The way to determine the uniqueness of features contained within

a keyframe is done through the use of *inverse document frequency* (IDF). The IDF algorithm will give a higher weight to less frequent features, making them easier to detect. This paper used a sum of IDF within a keyframe to determine its score. This was done accordingly:

$$\mathcal{R}_i(t) = \sum_{k \in \mathcal{W}_i} \log_2 \frac{N(t)}{n_{w_k}(t)} \tag{3.13}$$

where $\mathcal{W}_i \subseteq \mathcal{W}(t)$ is the subset of words found in image i, $n_{w_k}(t)$ is the number of images containing the word w_k and $N(t)$ is the number of images containing all words in the vocabulary. In a similar way as for local saliency, one also calculates a normalized global saliency score $S_{G_i} \in [0, 1]$:

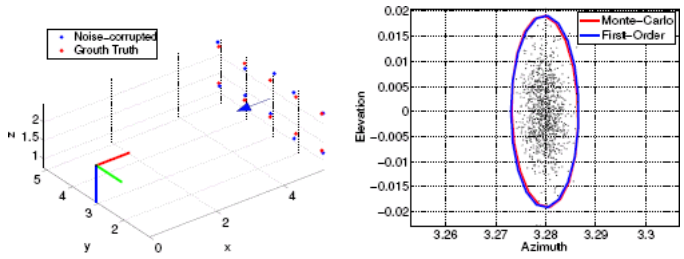
$$S_{G_i} = \frac{\mathcal{R}_i(t)}{\mathcal{R}_{max}} \tag{3.14}$$

where $\mathcal{R}_{max}$ is the maximum summed IDF score encountered thus far. As the rarity of words changes over time, the $\mathcal{R}_i$ score needs to be updated regularly.

3.4 Real-time SLAM with Piecewise-planar Surface Models and Sparse 3D Point Clouds

Motivation: *This article has not been used in this work, but it is added to the literature study as this article proposes a method for evaluating the plane uncertainty related to DVL approximated plane in front of the ROV. This uncertainty evaluation can be relevant for further development of the proposed pose-graph solution in the aquaculture net pen environment.*

In this paper, Ozog and Eustice implemented a real-time SLAM solution where a DVL sensor was used as a mapping and pose-correction device for ship hull inspections. The DVL extracted planes from the scene, and these were used as factors in a factor graph, using the iSAM framework [23]. The solution provided was sufficient for correcting odometry inaccuracies, and enhancing the performance of mono camera measurements.



(a) Least-squares plane fit (b) First-order covariance

FIGURE 3.6: Shows the least square fit of a plane approximation using PCA on DVL measurements (a). In (b), the propagation of uncertainty of the DVL point cloud to elevation and azimuth. Courtesy of [39]

The SLAM algorithm they constructed in this paper estimates pose with six-DOF; $\mathbf{x}_i = [x_i, y_i, z_i, \phi_i, \theta_i, \psi_i]$. The plane approximation of the $k-th$ observed plane seen in index frame $\mathbf{x}_i$ is expressed as: $\boldsymbol{\pi}_k^i = [a_k^i, e_k^i, d_k^i]$. The components a_k^i, e_k^i and d_k^i are the azimuth- and elevation of the surface normal, and the orthogonal distance of the vehicle to the plane, respectively [39]. The unit surface normal of the plane, $\mathbf{n}_k^i$, is found from a least-square fit using a sliding window of N consecutive DVL measurements with respect to the vehicle pose $\mathbf{x}_i$, $\mathbf{p}_k^i \in \mathbb{R}^{3\times N}$. The LS estimate of $\mathbf{n}_k^i$ is achieved by using principal component analysis (PCA) on the DVL points, $\mathbf{p}_k^i$, and this LS-fitting is illustrated in figure 3.6 (a). The surface normal can be modeled as a function of azimuth and elevation as seen in equation 3.15.

$$\mathbf{n}_k^i = \begin{bmatrix} n_{k_x}^i \\ n_{k_y}^i \\ n_{k_z}^i \end{bmatrix} = h(\begin{bmatrix} a_k^i \\ e_k^i \end{bmatrix}) = \begin{bmatrix} \cos(e_k^i)\cos(a_k^i) \\ \cos(e_k^i)\sin(a_k^i) \\ \sin(e_k^i) \end{bmatrix} \tag{3.15}$$

By inverting the Cartesian surface normal in equation 3.15 one achieves the azimuth and elevation:

$$\begin{bmatrix} a_k^i \\ e_k^i \end{bmatrix} = h^{-1}(\begin{bmatrix} n_{k_x}^i \\ n_{k_y}^i \\ n_{k_z}^i \end{bmatrix}) = \begin{bmatrix} \text{atan2}(n_{k_y}^i, n_{k_x}^i) \\ \text{atan2}(n_{k_z}^i, \sqrt{{n_{k_x}^i}^2 + {n_{k_y}^i}^2}) \end{bmatrix} \tag{3.16}$$

The standoff distance parameter, d_k^i, is found by taking the dot product of the centroid of $\mathbf{p}_k^i$ and $\mathbf{n}_k^i$. By using three-DOF parametrization of the plane vector, $\boldsymbol{\pi}_k^i$, one avoids

issues of over-parametrization of nodes in the iSAM framework.

The Plane Factor

A single plane factor connected to a pose, $\mathbf{x}_i$, is modeled as a unitary factor:

$$\Psi(\boldsymbol{\pi}_k^i; \mathbf{z}_{\boldsymbol{\pi}_k^i}, \Sigma_{\boldsymbol{\pi}_k^i}) = \left\| \mathbf{z}_{\boldsymbol{\pi}_k^i} - \boldsymbol{\pi}_k^i \right\|^2_{\Sigma_{\boldsymbol{\pi}_k^i}}$$

where $\mathbf{z}_{\boldsymbol{\pi}_k^i}$ is the measurement of the plane, while $\Sigma_{\boldsymbol{\pi}_k^i}$ is the covariance estimate of the plane measurement. $\Sigma_{\boldsymbol{\pi}_k^i}$ is estimated by assuming that the 3D-points stored $\mathbf{p}_k^i$ is corrupted by Gaussian noise. As these points are mapped to the plane, $\boldsymbol{\pi}_k^i$, by some non-linear function $f()$, one can propagate the uncertainty of the points by linearizing $f()$ around the observed measurements, $\mathbf{p}_k^i$. This can be described mathematically as:

$$\boldsymbol{\pi}_k^i = f(\mathbf{p}_k^i) \quad \mathbf{p}_k^i \sim \mathcal{N}(\boldsymbol{\mu}_{\mathbf{p}_k^i}, \Sigma_{\mathbf{p}_k^i})$$
$$\Sigma_{\boldsymbol{\pi}_k^i} \approx \mathbf{F} \Sigma_{\mathbf{p}_k^i} \mathbf{F}^T$$

where $\mathbf{F}$ is the jacobian matrix of $f()$. In figure 3.6 (b) one can see an illustration of this first-order covariance approximation.

Chapter 4

Method & Implementation

In this chapter, the method and implementation of the pose SLAM algorithm for obtaining the ROV path in the net pen will be explained. The method uses the iSAM2 framework for achieving this and has been inspired by the articles [10] and [27] and the fieldwork in [40]. The loop closure problem is solved using a proposed filtering method to achieve correct data association. The DVL has been used together with the mono-camera measurements to obtain the loop closure factor. An important assumption for this method is that the vertical walls of the fish cage are assumed to be planar.

Main Assumption

The vertical fish cage walls are assumed to be planar.

The pose SLAM algorithm implemented in this thesis can be divided into two parts: the front-end and the back-end. This architecture is illustrated in figure 4.2. The front-end is responsible for data abstraction, feature extraction and the data association of measurements, and the back-end is responsible for the inference of the abstracted data produced by the front-end [6]. The construction of both of these components will now be explained for pose SLAM, starting with the back-end. For convenience, the back-end section will also include the factor graph structure. After the front-end and back-end, the selected noise parameters will be listed and explained.

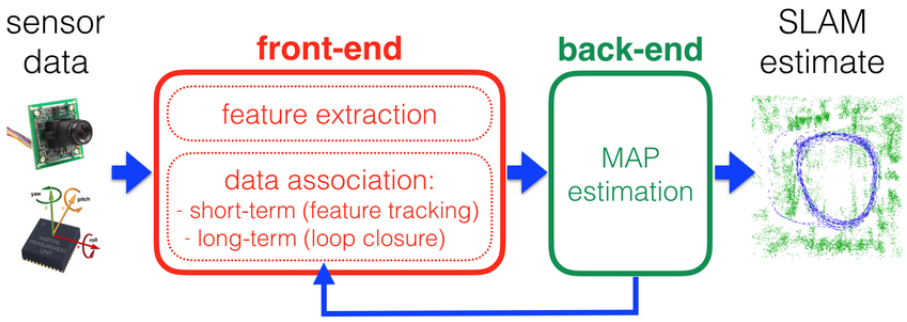


Figure 4.1: The SLAM front-end and back-end. Courtesy of [6]

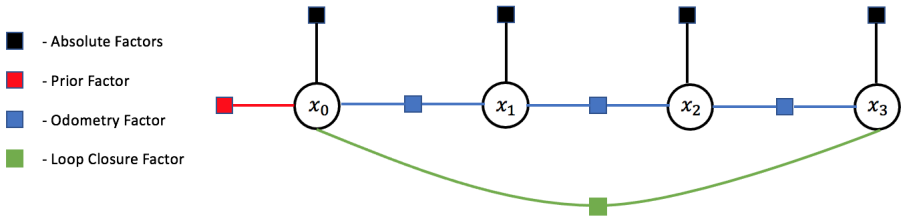


FIGURE 4.2: The pose graph structure used in this work. Here the absolute factors are measurements received from compass, depth sensor, tilt sensor and USBL. The prior are based on estimate from an extended Kalman filter, DVL speed estimates are used as a odometry factor and loop closure are obtained from image matching

4.1 The Simultaneous Localization and Mapping Back-End

The method used in this work can be summarized by the pose graph shown in figure 4.2. The reason why only optimizing for poses instead of solving the full SLAM problem is due to the environment the ROV operates in. Not only is this an environment containing low amounts of feature points to be mapped (as the corners in the grid cannot be used due to ambiguity problems between frames [27]), the environment is also affected by the currents. Although the fish cage has several mooring points which should hold the structure in place, the net can be deformed by the drag-forces of these currents as the net is a flexible structure [31], violating the SLAM assumption that landmarks are fixed in space. The non-linear pose SLAM problem has been solved using the iSAM2 framework explained in section 2.5, where the GTSAM library has been used to construct the factor graph and optimization. As the iSAM2 framework tries to find the MAP solution of the pose graph, this enables effortless addition of measurements obtained from various sensors. The optimization algorithm used in GTSAM is the Levenberg-Marquardt Algorithm 2. As this requires an initial estimate to converge towards a solution, a four-DOF pose estimate provided by an extended Kalman filter has been used. The EKF is an in-house algorithm of SINTEF that assumes stability in pitch and roll and fuses the measurements of the DVL, USBL, compass, depth-sensor and the ROV gyro measurement (measuring yaw rate).

The factor graph structure used in this work is visualized in figure 4.2. The graph consists of four factors: absolute factor, prior factor, odometry factor, and loop closure factor. All of the factors are modeled by Gaussian distribution with a diagonal covariance matrix. The absolute factors are factors that only related to a single pose. The absolute factors consist of the measurements from the pressure sensor (providing depth measurements), the compass reading, tilt sensors in pitch and roll, and the USBL sensor reading in the NE-plane of the NED-frame. As the USBL transponder is not placed in the origin of the body-frame, one needs to transform this measurement. How this is done is shown in 4.3. The direct USBL measurement is $\mathbf{t}^w_{wu}$, while one is interested in using $\mathbf{t}^w_{wb}$ for the USBL factor. Here, b refers to body-frame, w refers to NED-frame/world-frame, and u refers to USBL-frame. $\mathbf{t}^w_{wb}$ can be calculated by vector subtraction:

$$\mathbf{t}^w_{wb} = \mathbf{t}^w_{wu} - \mathbf{t}^w_{bu} \tag{4.1}$$

where $\mathbf{t}_{bu}^{w}$ is found by transforming the know vector $\mathbf{t}_{bu}^{b}$ into the world coordinate system:

$$\mathbf{t}_{bu}^{w} = \mathbf{R}_{b}^{w}(\psi^{w})\mathbf{t}_{bu}^{b} = \begin{bmatrix} c_{\psi^w} & -s_{\psi^w} & 0 \\ s_{\psi^w} & s_{\psi} & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{t}_{bu}^{b} \tag{4.2}$$

This calculation neglects the pitch and roll angles, and assumes that the ROV is heading is the same as the direct heading measurement from the compass. The prior factor used for the initial pose estimate of the ROV stems from the four-DOF EKF. The odometry factor is a unitary factor obtained from the DVL speed estimate and yaw rate from the gyro, and it provides a connection between two consecutive poses $\mathbf{x}_{n-1}$ and $\mathbf{x}_{n}$. The relative transformation between one body frame to the next is modeled by a homogenous transformation matrix described in section 2.1. One issue with the DVL velocity measurement is that these are not always available. When DVL measurements are not available, as the SINTEF in-house outlier detection algorithm filters out bad measurements, the speed estimates of the four-DOF EKF is used as a substitute. The final factor is the loop closure factor, and this factor is obtained from the re-detection of a previously identified scene by the camera. This factor and the data association algorithm for achieving it will be explained in the following section.

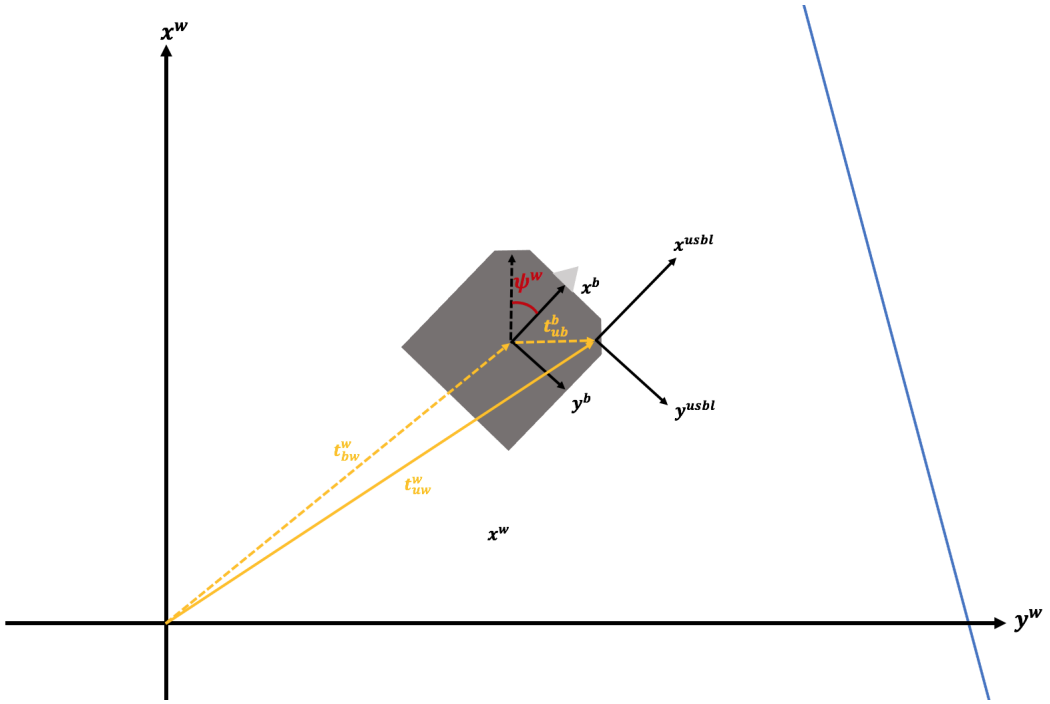


FIGURE 4.3: USBL measurement transformation from USBL-frame to body-frame

4.2 The Simultaneous Localization and Mapping Front-End

4.2.1 Feature Extraction

Cameras are one of the most used sensors for obtaining loop closures in SOTA SLAM systems. However, obtaining loop closures in the fish cage environment using a camera is not an easy task for several reasons. The main reason for this is because the

fish cage is constructed by a standardized grid net, making every scene look similar in texture. An example of how different scenes in the fish cage may look like is illustrated in figure 4.4. These pictures also illustrate why VO would be hard to implement in this environment, as there would be an ambiguity in matching similar corner features between frames.

However, due to high nutrient concentration in the net pen, this causes the blossom of algae on the net [20]. As these algae grow in random patterns, the patterns created will be unique to a specific location and can therefore be used as features to orient around. An example of such algae growth is shown in image 4.5a. Here, one can observe the algae on the net; however, these features are not that evident from a computer perspective due to low contrast. As described in [27], as a way of increasing the contrast while also dealing with the nonuniform lightning, a common issue in underwater imagery [13], one can use the contrast limited adaptive histogram specification (CLAHS), which is a more general version of the CLAHE algorithm. The effect of applying this algorithm is shown in image 4.5b, and it is apparent that algae features are more distinct compared to the original image. The CLAHS algorithm inputs a grayscale, blurred image and divides this image into equal-area regions. The image is pre-blurred is to reduce noise while also forcing the feature detector to focus on larger-scale features. For each region, *histogram specification* is applied, which tries to redistribute pixel intensity values from the original image to fit the desired distribution using a gray-level intensity transformation function $p_{new} = T(p_{old})$. Afterward, a *clip limit* is used to set a maximum value to each individual bin (intensity value) in the transformed histogram, where in some cases, the clipped intensities would be uniformly distributed to the remaining bins [13].

Empirical studies were conducted to determine the desired distribution and parameters of the CLAHS algorithm. These experiments showed that one obtained a higher contrast image in the fish cage environment by remapping the image histogram towards a Rayleigh distribution with a scale parameter $\alpha = 0.4$. The clip limit was set to 6, but their intensity redistribution was not included. The size of each region was selected to be 15×15.

128-dimensional SIFT features were used to extract image characteristics. The reasoning for selecting this feature detector was based on the fact that SIFT features detect blobs on the image as it uses DoG at different scale-spaces [21]. These blobs will be a more general feature compared to a pixel-based feature detector as BRIEF used in ORB. SIFT is therefore not as subjected to noise and would therefore be better to use when wanting to track algae, as these can be somewhat affected by the ocean currents. 128-dimensional descriptor, as opposed to 64, was selected to ensure the possibility of detecting larger algae patches. Despite the camera having a frequency of 60Hz, extracting SIFT features from each image is not ideal as it is a computationally heavy operation, and there would be little to no additional information. Therefore there was only sampled a keyframe every second.

Dynamic Environment Management

Another complication related to the fish cage environment is the presence of fish and objects outside of the net. Both of these issues are shown in figure 4.6, and both are a part of the same underlying problem: they are moving objects in the scene. Moving

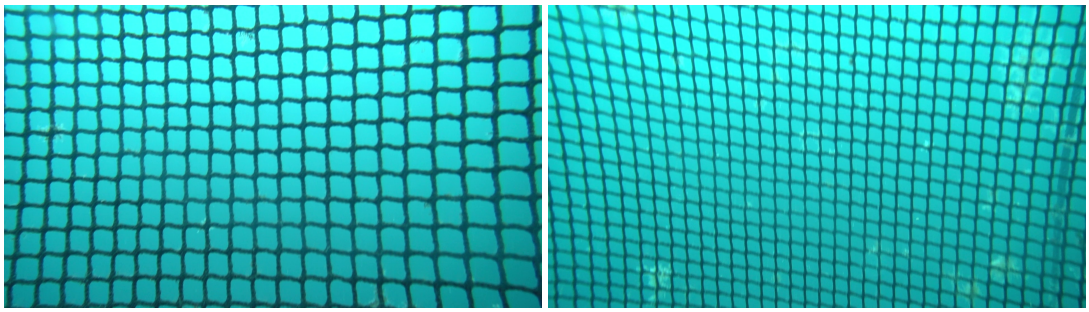
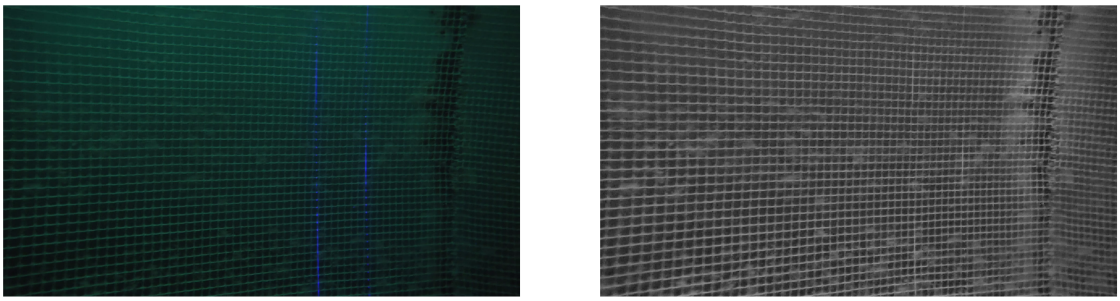


FIGURE 4.4: The extreme case showing the difficulty of determining where a scene is from, making it hard to do loop closure or VO due to the ambiguity related to frame matching.



(A) Original image (B) Applying the CLAHS algorithm on image (A)

FIGURE 4.5: Algae growth on the net pen

objects are problematic as these are not features one wants to track in the environment as they cannot be used for loop closure. The fact that fishes move around in the scene is evident; however, objects outside the net appear to move as the net is transparent. When moving from one frame to another, the object's position relative to the net would have changed due to the ROV movement.

The concept called positional-invariant robust feature (PIRF) described in [25] has been used to avoid sampling features from moving objects in the scene. Instead of sampling features directly and then identifying the features' respective words in a BOW dictionary, one is instead only using the matched descriptors in a frame $\mathcal{F}_i$ to its previous frame $\mathcal{F}_{i-1}$ to describe the features of frame $\mathcal{F}_i$. Note that in the original paper, the PIRF- feature descriptor is an average of the matched descriptors, while in this work, the matched descriptor of frame $\mathcal{F}_i$ has been used. The feature matching procedure used to obtain the PIRF features is based on Lowe's ratio test, which compares the Euclidean distance of the closest neighbor of a possible match to that of the second-closest neighbor [33]. If one has a feature f_a described in frame $\mathcal{F}_i$ and the best match and the second best match in frame $\mathcal{F}_i$ is f_b^1 and f_b^2 respectively, then the ratio test will be as follows:

$$\frac{d(f_a, f_b^1)}{d(f_a, f_b^2)} < L_{th} \tag{4.3}$$

where L_{th} is a ratio threshold defining how much better the best match needs to be compared to the second-best match for it to be considered a match. Setting the L_{th} to a high value would correspond to being more conservative in the feature

matching, avoiding ambiguous matches (a real concern in the fish cage as explained in the previous section). In this work, L_{th} was set to a value of 0.7, which appeared to work sufficiently, as bad matches are further filtered out in the homography procedure when doing loop closure. More on this later.

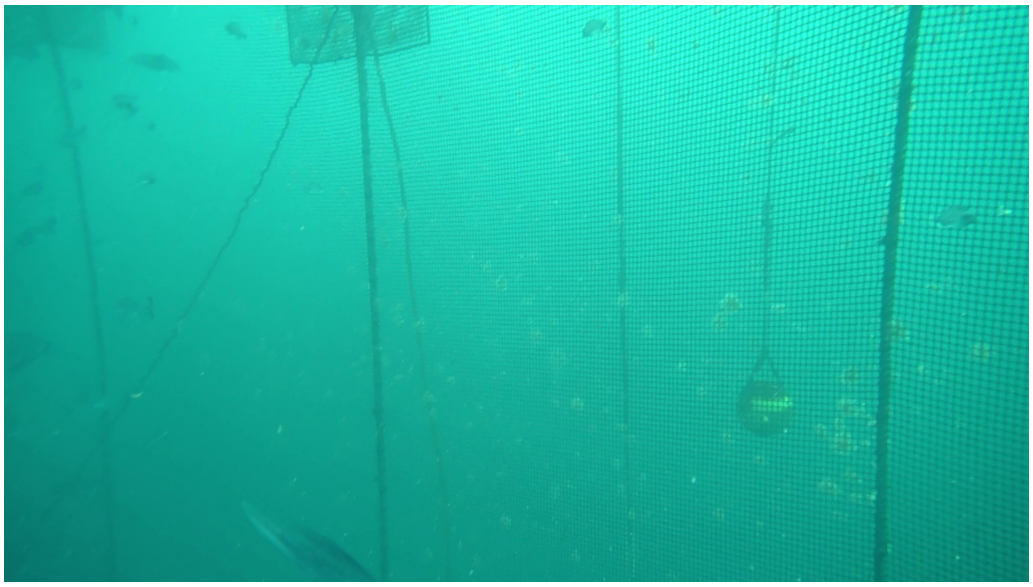


FIGURE 4.6: Entirety of the fish cage environment

4.2.2 Data Association

Bag of Words

In the work of [27] they used a self-created online BOW algorithm, that started with an empty vocabulary and incrementally added new words to the vocabulary when discovered. In this thesis, there has however been used a more straightforward offline approach for constructing the vocabulary. As the vocabulary is created offline, one needs to set a predefined vocabulary size, and this was set to 150 words. The selected vocabulary size was based on the vocabulary size achieved in [27], as they arrived at about 200 words after 200 min of mission time using their online vocabulary. As the ship hull used in their experiments and the fishnet used in this are both low on unique features, it was assumed that the number of unique words found in the fishnet would be pretty similar.

The generation of the BOW vocabulary was done by extracting features from a training dataset consisting of 305 images. SIFT features were extracted from these images, and the descriptors were clustered into 150 words. This was achieved using the k-means method, using ten iterations with different centroid seeds, where the solution is taken from the iteration which obtained the minimum total variance of all clusters. The mean of each cluster is the word description used in the vocabulary. As all PIRF-features extracted during operation need to be assigned to their corresponding words, this would be a linear-time operation if not using any aiding frameworks. To avoid linear run-time, the words generated are structured in a simple binary tree. The k-means method was also used in the generation of the binary tree. This was done by starting at a root node and dividing the 150 words into 2 clusters, obtaining the tree's respective left and right branches. For each branch, the k-means was recursively

applied until each branch/leaf only contained one word. Since the k-means method tries to minimize the Euclidean distance within each cluster, the Euclidean distance was also used when doing word searching. As this tree is constructed in a naive way, one is not able to obtain a fully balanced tree and a search time of $\mathcal{O}(log_2(n))$ [8]. However, the tree structure obtained from this approach had a maximum depth of 15, whereas a fully balanced binary tree consisting of 150 nodes would have a depth of eight ($log_2(150) \approx 7.23 \approx 8$), hence the binary tree used provides almost logarithmic search time.

Finding Loop Closure Candidates

As explained previously, the loop closure detection algorithm used in this work is inspired by the work of [27] and [10]. In short, loop closure candidates are found by a proposed filtering technique based on the net inspection procedure. In figure 4.7, there is a flow chart outlining the processing pipeline of this filtering algorithm. This technique involves filtering the image database based on a global saliency threshold, depth- and heading thresholds concerning the depth and heading a current image was captured, and a histogram similarity threshold. In addition to this, to avoid adding visual factors for every frame, the ten most previous keyframes are not accounted for when searching for loop closures. The filtering concepts will now be explained in more detail.

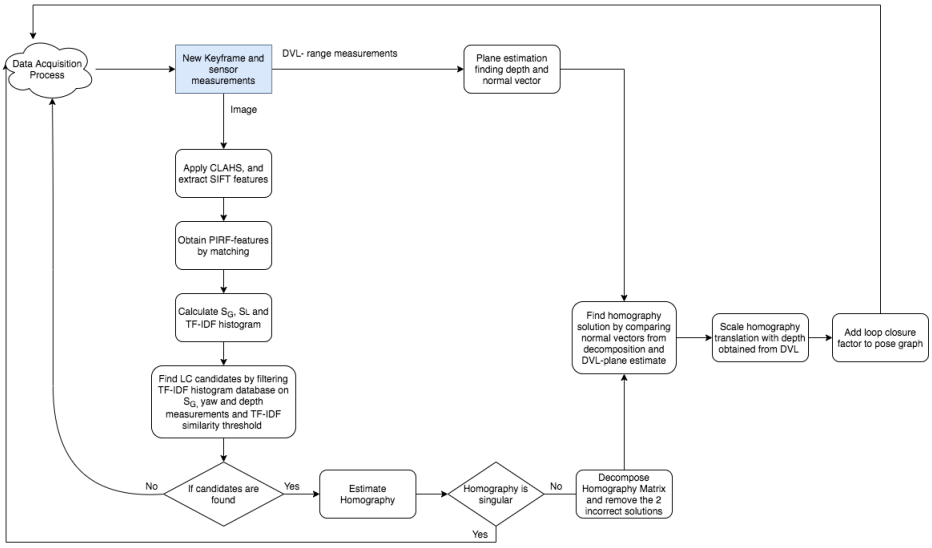


FIGURE 4.7: Flow chart summarizing how loop closure factors are generated

Local/Global Saliency & Global Saliency Thresholding

The words obtained from the BOW algorithm are used to construct local saliency, global saliency, and *TF-IDF* histogram. The concepts of local saliency and global saliency were explained in section 3.3 but will be reviewed again.

The word diversity within a keyframe, $\mathcal{F}_i$, can be found by evaluating the entropy of its BOW histogram as done in equation 4.4.

$$H_i = -\sum_{k}^{W(t)} p(w_k)\log_2(p(w_k)) \tag{4.4}$$

Here $\mathcal{W}(t)$ is the number of detected unique words at time t, and $p(w_k)$ is the *term frequency* (TF) of word w_k within image i. The term frequency is equal to $\frac{n_{ki}}{n_i}$ where n_{ki} is the occurrence of word k in image i, and n_i is the total word count in image i. The entropy can be normalized with respect to the maximum entropy possible which can be proved to be equal to $\log_2(\mathcal{W}(t))$ [27] and one gets:

$$S_{L_i} = \frac{H_i}{\log_2 \mathcal{W}(t)} \tag{4.5}$$

The measure $S_{L_i} \in [0, 1]$ in equation 4.5 is termed local saliency, and is a measure of the intra-image feature diversity.

Global saliency, on the other hand, is a measure of the inter-image uniqueness of an image. This measure wants to describe if an image contains features that are considered rare in the dataset. To capture this rarity the summed *inverse document frequency* (IDF) of all words within an image has been used to obtain this measure, and then normalized the maximum summed IDF-score encountered thus far. This is described mathematically in equation 4.6

$$S_{G_i} = \frac{\mathcal{R}_i(t)}{\mathcal{R}_{max}} \tag{4.6}$$

where $\mathcal{R}_i(t)$ is the summed IDF and equals to:

$$\mathcal{R}_i(t) = \sum_{k \in \mathcal{W}_i} \underbrace{\log_2 \frac{N(t)}{n_{w_k}(t)}}_{\text{IDF of word } w_k} \tag{4.7}$$

The inputs $n_{w_k}(t)$ is the number of images in the vocabulary database containing word w_k, and $N(t)$ is the total number of images comprising the vocabulary database. A thing to note is that as both S_{L_i} and S_{G_i} are measures based on time-dependent variables, hence both need to be updated regularly.

As the local saliency measures how feature-rich an image is, it will determine if a matching between two images will be likely to fail or not. Hence, a minimum local saliency threshold has been set such that images below this threshold are discarded, saving storage. The global saliency is used as a temporary data association measure to determine if there is a possibility of loop closure. Loop closure is a risky procedure. A wrong data association would lead to the entire robot pathing becoming corrupted; one, therefore, wants to be as confident as possible when doing a loop closure. As the global saliency tracks inter-image rarity, images with a high global saliency would represent images that are possible loop closure candidates. Hence, loop closure candidates are filtered based on defined global salience threshold $S_{G,th}$.

The saliency algorithm used in this work is shown in algorithm 5 and is based on the algorithm from [27].

Algorithm 5 Saliency Algorithm

Require: BOW vocabulary $\mathcal{W}$, features of frame i $F_i = [f_1, f_2, ..., f_n]$
Require: statistics $N(t)$, $\mathcal{H}_b(t)$, $\mathcal{H}_{wt}(t)$
{ Generate image word histogram }
Initialize BOW histogram: $\mathcal{H}_i \leftarrow \varnothing$
for each feature $f_j \in F_i$ **do**
    map feature f_j to word in vocabulary $w_k \in \mathcal{W}$
    increment histogram: $\mathcal{H}(w_k)_i \leftarrow \mathcal{H}(w_k)_i + 1$
end for
Update $\mathcal{H}_b(t)$ if there are words in $\mathcal{H}_i$ not seen before
Increment $\mathcal{H}_{wt}(t)$ based on word found in $\mathcal{H}_i$
Increment image tracker $N(t) \leftarrow N(t) + 1$

{ Calculate local saliency }
$S_{L_i} = 0$, $H_i = 0$
for all words k in $\mathcal{W}$ **do**
    Compute $H_i(w_k)$
end for
Compute S_{L_i}

{ Calculate global saliency & TF-IDF histogram }
for all tracked word in $\mathcal{H}_i$ **do**
    Calculate TF: $TF \leftarrow \frac{n_{ki}}{n_i}$
    Calculate IDF: $IDF \leftarrow \log_2 \frac{N(t)}{n_{w_k}(t)}$
    $\mathcal{R}_i += IDF$
    $\mathcal{H}_{tf-idf_i}(w_k) \leftarrow TF * IDF$
end for
Calculate S_{G_i}
if $\mathcal{R}_i > \mathcal{R}_{max}$ **then**
    Update $\mathcal{R}_{max}$: $\mathcal{R}_{max} \leftarrow \mathcal{R}_i$
end if

if $mod(N(t), 10) == 0$ **then**
    Update all local saliencies and global saliencies
end if
return S_{L_i}, S_{G_i}, $\mathcal{H}_i$, $\mathcal{H}_{tf-idf_i}$, $\mathcal{R}_{max}$,

The now undescribed parameters used in 5 is described as follows:

- $\mathcal{H}_i$ - Word count histogram of image i
- $\mathcal{H}_{wt}(t)$ - Histogram tracking number of images containing a specific word $n_{w_k}(t)$
- $\mathcal{H}_b(t)$ - Binary histogram tracking all words which has occurred from dictionary $\mathcal{W}$. Boolean histogram of $\mathcal{H}_{wt}$
- $\mathcal{H}_{tf-idf_i}$ - TF-IDF histogram of image i

The TF-IDF Histogram

As the BOW algorithm provides a framework for efficiently labeling images, one way of determining the similarity of two images would be to calculate the euclidean distance of their word frequency histograms $\mathcal{H}_i$. However, this type of comparison is naive and would lead to false matches of images in many cases, as some features will be occurring more often than others. This is especially evident in the fish cage, as the net corners will be present in each image; hence these features provide little information. A better way of evaluating image similarity would be to compare the TF-IDF histograms of images, a histogram constructed by the product of the term frequency and the inverse document frequency. This is a better measure as weighting TF by IDF would provide a histogram where rare features are favored, and standard features are ignored.

Calculating the Euclidean Distance between two histograms would only result in a scalar number. A more relatable measure would be to use the cosine similarity [47], where one obtains a value between $[0, 1]$, representing how similar the images are. Using the cosine similarity, similar images would obtain a value close to one. The cosine similarity can be calculated as follows:

$$d_{cos} = 1 - \frac{\mathcal{H}_{TF-IDF,i}^{T} \mathcal{H}_{TF-IDF,j}}{\|\mathcal{H}_{TF-IDF,i}\| \, \|\mathcal{H}_{TF-IDF,j}\|}$$

When a new image i arrives, the TF-IDF histogram is calculated using by algorithm 5, and then $\mathcal{H}_{TF-IDF,i}$ is compared to all the other TF-IDF histograms stored in the database using the cosine similarity.

Heading and Depth Filter

When the ROV performs net pen inspection, the ROV traverses the net pen and points towards the fishnet. As the net pen is a concave environment and the camera has a limited FOV, one can filter undesired loop closure candidates by considering the heading and depth measurements (d_i,ψ_i) recorded when a keyframe was taken. This filter is described mathematically in the equations below where $\mathbf{d}_{kf}$ and $\boldsymbol{\psi}_{kf}$ are depth and heading vector, respectively, containing all the previous depth and heading measurements related to stored keyframes.

$$|\mathbf{d}_{kf} - d_{cur}| \leq d_{th} \tag{4.8}$$

$$\sin|\boldsymbol{\psi}_{kf} - \psi_{cur}| \leq \sin\psi_{th} \tag{4.9}$$

The reason why one needs to use the sin function here, is such that the function accounts for the proximity between the angles π and $-\pi$. d_{th} was set to be equal to $0.8m$ while ψ_{th} was set to 7.5 degrees.

Data Association Algorithm - Summary

To summarize, the algorithm for detecting loop closure candidates is based on 4 consecutive filtering steps:

1. Filter candidates below $S_{G_{th}}$
2. Filter candidates below d_{th}
3. Filter candidates below ψ_{th}

4. Filter candidates below a histogram similarity threshold $\mathcal{H}_{th}$

Afterward, the remaining candidates are sorted based on their cosine similarity to $\mathcal{H}_{TF-IDF,cur}$, and the top 3 candidates are selected as loop closure candidates. Below, one can see a summary of all selected parameter variables related to the data association algorithm, which were based on empirical studies.

Summary: Data Association Variables

$$S_{G_{th}} = 0.45, \quad S_{L_{th}} = 0.4, \quad d_{th} = 0.8m, \quad \psi_{th} = 7.5^\circ$$
$$\mathcal{H}_{th} = 0.7, \quad \mathcal{H}_{TF-IDF_{th}} = 0.45, \quad L_{th} = 0.7$$

4.2.3 Loop Closure Factor

If loop closure candidates are obtained after the filtering technique, the algorithm proceeds with feature matching and homography matrix. The matching is done in the same way as obtaining PIRF features, and the homography matrix is afterwards estimated from the matches using findHomography()-function in the OpenCV library [22]. The findHomography()- function tries to find the perspective transformation from a source image to a destination image using the RANSAC algorithm. RANSAC gives the initial estimate by selecting a minimum amount of random points needed to calculate the homography matrix $\tilde{\mathbf{m}}^c = \mathbf{H}\tilde{\mathbf{m}}^d$, and then evaluating the number of point inliers and outliers from this representation. If the inlier/outlier- ratio is not satisfactory, a new set of random points is selected, and the procedure is repeated. However, if the ratio is satisfactory, the solution of $\mathbf{H}$ is optimized by minimizing there-projection error using the Levenberg Marquardt algorithm 2. The re-projection error function is shown in the equation below [45].

$$\sum_i \left(x_i^d - \frac{h_{11}x_i^s + h_{12}y_i^s + h_{13}}{h_{31}x_i^s + h_{32}y_i^s + h_{33}} \right)^2 + \left(y_i^d - \frac{h_{21}x_i^s + h_{22}y_i^s + h_{23}}{h_{31}x_i^s + h_{32}y_i^s + h_{33}} \right)^2$$

If enough inliers are found, then the determinant of the homography needs to be evaluated. As stated in the projectivity definition 1, the homography matrix cannot be singular. Hence, a determinant threshold is set to avoid bad homographies, and this is set to 0.7. If the homography matrix passes this threshold, the matrix is decomposed into a rotation, translation, and normal vector of the plane ($\mathbf{R}$, $\mathbf{t}$ and $\mathbf{n}$) by using the OpenCV decomposeHomographyMat()- function. The decomposeHomographyMat()-function is an implementation of the homography decomposition algorithm derived in [35]. As explained in section 2.2.3, four solutions are obtained from this decomposition due to the ambiguity of the homograph matrix. Two of these can be falsified directly as they will contain negative point depth in either camera frame, which is not physically possible [35]. Negative point depth can be evaluated by the z-component sign of the normal vector, as a negative sign would indicate a plane behind the desired camera. See image 2.6.

One is then left with two solutions: $S_1 = \{\mathbf{R}_a, \mathbf{t}_a, \mathbf{n}_a\}$ and $S_2 = \{\mathbf{R}_b, \mathbf{t}_b, \mathbf{n}_b\}$, and these can be furthered filtered to one solution by using the range measurements from the DVL. As the DVL receives four range measurements from its beams, one can use these measurements to estimate the plane in front of the ROV, using an LS-fitting method described in [2], see section 3.2. One would then obtain the parameters of

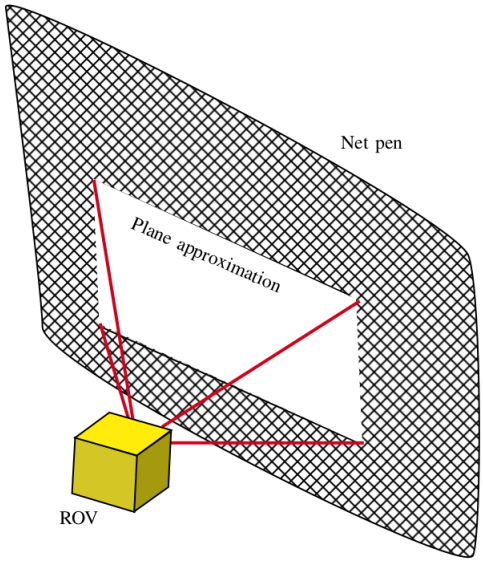


FIGURE 4.8: Plane approximation of the net pen using DVL range measurements. Courtesy of [2]

the plane equation (b, c , d), $f(x, y, z) = -x + by + cz + d = 0$, and these parameters are further be used for estimation of the unit normal vector to the plane, $\mathbf{n}^d_{unit}$, as such [2]:

$$\mathbf{n}^d_{unit} = \frac{\mathbf{n}^d}{||\mathbf{n}^d||_2} = \frac{1}{\sqrt{1+b^2+c^2}} \begin{bmatrix} -1 \\ b \\ c \end{bmatrix}$$

By evaluating the Euclidean distance between $\mathbf{n}^d_{unit}$ and $\mathbf{n}_a$ and $\mathbf{n}_b$, one can obtain a final solution of the homography decomposition by selecting the solution with the minimum Euclidean distance to $\mathbf{n}^d_{unit}$. As explained in section 2.2.3, the solution to the homography decomposition is only up to scale, as an image cannot capture the depth of the scene. The DVL, however, captures this depth. The depth vector obtained in [2] was the plane with respect to the body, but here one is interested in the plane's depth with respect to the camera. The same procedure could obtain $d_{c/net}$ by replacing the $\mathbf{r}^d_{db}$ with $\mathbf{r}^d_{dc}$, where c denotes the camera-frame. Hence, the vector from a random point on the plane to the camera can be expressed as:

$$\mathbf{v}^d_{plane,c} = \mathbf{r}^d_{dc} + \mathbf{p}^d_0 = \begin{bmatrix} x^d_{dc} + x^d_0 \\ y^d_{dc} + y^d_0 \\ z^d_{dc} + z^d_0 \end{bmatrix}$$

The distance from the net to the camera can then be found by the following equation:

$$d_{c,net} = |(\mathbf{v}^d_{plane,c})^T \mathbf{n}^d_{unit}| = \frac{|-x^d_{dc} + by^d_{dc} + cz^d_{dc} - d|}{\sqrt{1+b^2+c^2}}$$

One can then obtain the scaled translation vector by multiplication:

$$\mathbf{t}^* = \mathbf{t}_{solution} * d_{c,net}$$

This way one acquire the transformation matrix that maps from a source-frame to a destination-frame:

$$\mathbf{T}_s^d = \begin{bmatrix} \mathbf{R} & \mathbf{t}^* \\ \mathbf{0} & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}_s^d & \mathbf{t}_s^d \\ \mathbf{0} & 1 \end{bmatrix} \tag{4.10}$$

When a loop closure is found, one can select which frame will be chosen as source-frame and destination-frame. To avoid storing plane-depth measurements, the most recent frame is selected as the source-frame. This is because the translation scaling factor is based on the depth of the source frame ($\mathbf{H} = \mathbf{R}_s^d + \frac{\mathbf{t}_s^d}{d^s}(\mathbf{n}^s)^T$), and the normal vector found from the homography decomposition, $\mathbf{n}^s$, is also related to the source-frame. To create the loop closure factor, the GTSAM BetweenFactor() has been used. This is the same factor used for odometry measurement. The factor considers the transformation going from one pose to another and the related uncertainty of the odometry measurement. To avoid creating a factor that is directed from the current-frame to the previous, the transformation in equation 4.10 needs to be inverted:

$$\mathbf{T}_d^s = \begin{bmatrix} (\mathbf{R}_s^d)^T & -(\mathbf{R}_s^d)^T\mathbf{t}_s^d \\ \mathbf{0} & 1 \end{bmatrix} \tag{4.11}$$

It is difficult to evaluate the uncertainty related to rotation and translation obtained from the homography matrix decomposition. Firstly, because one deals with a net that is affected by the drag-forces of the current, which can cause the net to have slightly shifted when comparing a current- and previous frame. Secondly, few studies have been conducted on how a DVL is affected when receiving rebounds from a flexible structure. However, SINTEF has done a comparative study between the distance measurement from the DVL and distance measurement obtain from laser triangulation [4]. This was conducted on the same dataset as used in this work. The results here showed significant overlap between the two distance measures. Here they also tried to approximate the standard deviation (STD) of distance measurement produced by the DVL and the distance measurement from the laser triangulation. This was obtained by generating a quasi-ground-truth of the distance by fusing the two distance measurements with a linear Kalman filter. From this the STD of the DVL-distance was estimated to be 2.9cm, while the laser had 3.2cm [4].

To avoid being overconfidence in the distance measure, a factor was added such that the STD of the DVL distance was assumed to be equal to $\sigma_{d_{lc}} = 0.06m$. Finally, this uncertainty was propagated to the translation vector using the linearity theorem 2.3.1 for a Gaussian distribution [5]. The covariance of $\mathbf{t}_d^s (= -(\mathbf{R}_s^d)^T\mathbf{t}_s^d)$ could then be calculated accordingly:

$$Cov[\mathbf{t}_d^s] = \Big(-(\mathbf{R}_s^d)^T \frac{\mathbf{t}_s^d}{d^d}\Big)\sigma_d^2 \Big(-(\mathbf{R}_s^d)^T \frac{\mathbf{t}_s^d}{d^d}\Big)^T \tag{4.12}$$

The covariance of the rotation on the other hand was set manually, and used a diagonal matrix:

$$Cov[\mathbf{R}_s^d] = \mathbf{\Sigma}_{\mathbf{R}_{lc}} = \begin{bmatrix} \sigma_{\theta_{lc}}^2 & 0 & 0 \\ 0 & \sigma_{\phi_{lc}}^2 & 0 \\ 0 & 0 & \sigma_{\psi_{lc}}^2 \end{bmatrix} \tag{4.13}$$

where $\sigma_{\theta_{lc}}$ and $\sigma_{\phi_{lc}}$ were set to 5 degrees and $\sigma_{\psi_{lc}}$ was set to 7.5 degrees, being conservative.

4.3 Selection of Noise Parameters

In this section, noise parameters selected in this system will be listed. These variables were selected by various methods. Among others, [28] was used to find sensor precision estimates of the depth measurement, compass, gyro and inclinometer. The article [40] was used to deduce a rough estimate of the USBL uncertainty. The speed uncertainty of the DVL was based on bottom velocity long-term accuracy provided in the technical specification of the Nortek DVL 1000. It should be noted there is few studies on how DVL measurements are affected when used on a flexible structure. Therefore, a factor of safety has been added to avoid DVL measurement overconfidence. However, due to having a large number of tune-able variables in conjunction with limiting time, this thesis has not been focusing on fin-tuning of the system. Hence, this is added to future work for further development of the pose SLAM system. The noise variables selected are listed in the summary below.

Summary: Measurement Noise Variables

$$\mathbf{\Sigma}_{DVL} = \begin{bmatrix} \sigma_u^2 & 0 & 0 \\ 0 & \sigma_v^2 & 0 \\ 0 & 0 & \sigma_w^2 \end{bmatrix}, \quad \mathbf{\Sigma}_{USBL} = \begin{bmatrix} \sigma_x^2 & 0 \\ 0 & \sigma_y^2 \end{bmatrix}, \quad \mathbf{\Sigma}_{tilt} = \begin{bmatrix} \sigma_\theta^2 & 0 \\ 0 & \sigma_\phi^2 \end{bmatrix}$$

where

$$\sigma_u = 0.01m/s, \;\; \sigma_v = 0.01m/s, \;\; \sigma_w = 0.02m/s$$
$$\sigma_x = 1.3m, \;\; \sigma_y = 1.3m, \;\; \sigma_\theta = 5^\circ, \;\; \sigma_\phi = 5^\circ$$
$$\sigma_{depth} = 0.02m, \;\; \sigma_{compass} = 2^\circ$$

Loop Closure Paramters:

$$\mathbf{\Sigma}_{\mathbf{R}_{lc}} = \begin{bmatrix} \sigma_{\theta_{lc}}^2 & 0 & 0 \\ 0 & \sigma_{\phi_{lc}}^2 & 0 \\ 0 & 0 & \sigma_{\psi_{lc}}^2 \end{bmatrix}, \;\; \sigma_{d_{lc}} = 0.06m$$

where

$$\sigma_{\theta_{lc}} = 5^\circ, \;\; \sigma_{\phi_{lc}} = 5^\circ, \;\; \sigma_{\psi_{lc}} = 7.5^\circ$$

Chapter 5

Results & Discussion

In this section the results from running the algorithm on three datasets will be shown and discussed. As there was no GT available, the EKF, mentioned in the previous chapter, was used as a quasi-ground truth.

Datasets & Limitations

The datasets used for testing the pose SLAM algorithm are shown in figures 5.1, 5.2, and 5.3. In these plots, the ROV path is shown in the blue line, and this path is the estimate provided by the 4 DOF EKF filter, which assumes stability in pitch and roll. The red dots on the figures are the DVL range estimates, which are found by averaging the starboard and port side measurements individually. Due to the lack of ground truth for the ROV path, the EKF estimates were used as a quasi-GT.

One issue with these measurements is that the camera and other sensor measurements are not synchronized. The camera used had a frequency of 60Hz, while the other sensor measurements had a sample period of 0.1s. Synchronization between these, therefore, had to be done manually, and as the time of when the video recording was taken could only be tracked to the closest second, there was a synchronization errorof $\pm$0.5s. This offset is an issue when doing loop closures as the DVL plane estimate is not taken simultaneously as in the picture. If tight turns occur or the ROV traverses a corner, this will affect the LC-factor achieved by homography decomposition, depending on the DVL-plane estimation.

Another issue with the datasets is that there was no invented procedure for trailing the net pen for a SLAM system to work. Therefore, it was requested that the ROV should move in a zig-zag pattern when traversing the net pen such that there would be overlap between features and this way allowing for several loop closures. This zig-zag pattern was attempted in dataset II, shown in figure 5.2. However, due to low FOV in the vertical direction, it resulted in little to no overlap between the images captured by the ROV.

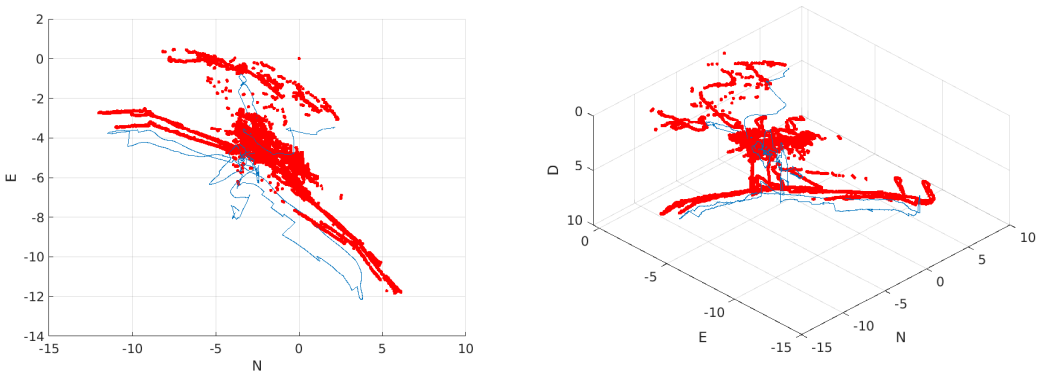


FIGURE 5.1: Dataset I - The red dots DVL range measurements, while the blue line is the ROV path estimate by the EKF

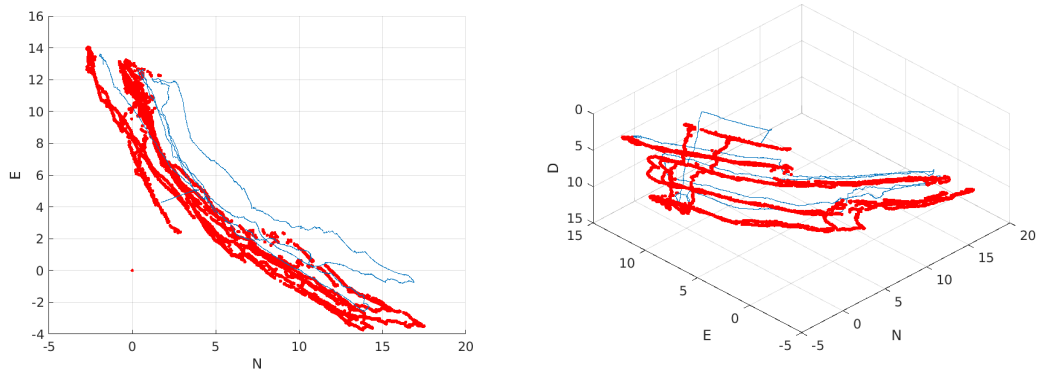


FIGURE 5.2: Dataset II - The red dots DVL range measurements, while the blue line is the ROV path estimate by the EKF

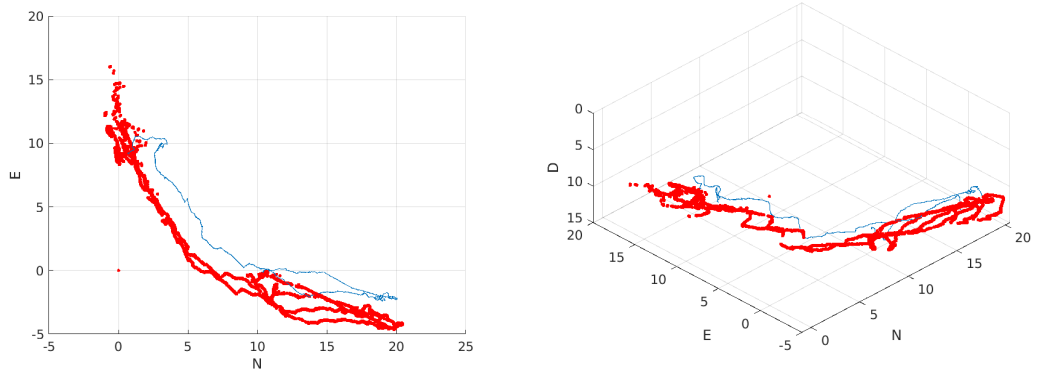
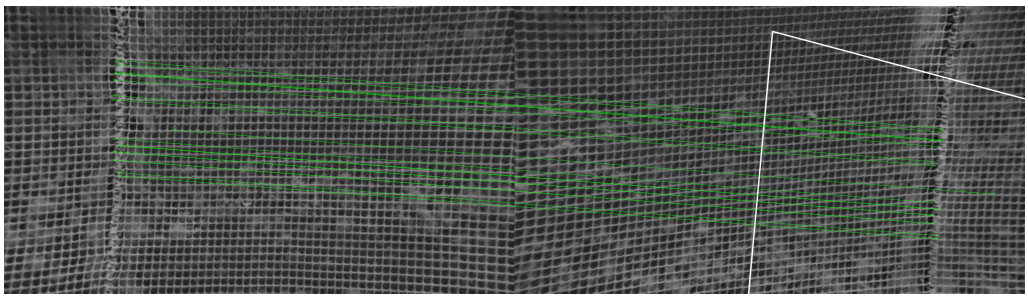
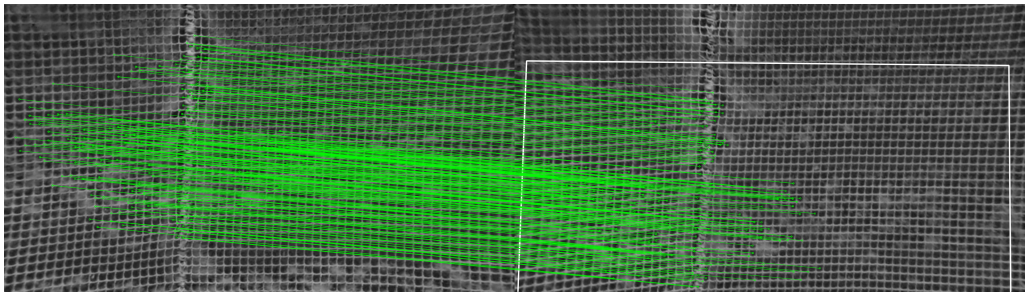


FIGURE 5.3: Dataset III - The red dots DVL range measurements, while the blue line is the ROV path estimate by the EKF

5.1 Results Dataset I - Loop Closure in a Corner



(A) Bad Loop Closure



(B) Good Loop Closure

FIGURE 5.4: Two loop closures which are achieved in rapid succession

The ROV path of dataset I can be seen in figure 5.1. The ROV is, in the beginning, traveling down to greater depths until it settles at 8m depth and starts traversing the net, going back and forth along a couple of planes. When traversing, the ROV has roughly 1.5m distance to the net and travels approximately a linear velocity of 0.15-0.2m/s, where occasional acceleration spikes. Along the path, there are occasional areas with high algae content, which provide possible loop closures.

During this traverse, there is a particular corner where loop closure occurs. This is the corner where the ROV changes direction to traverse back to where it came from. As the ROV has to change its momentum, it uses some time to do this, and several images are therefore captured from this area leading to multiple loop closures as these are globally salient images. The first LC that occurs is one where there is no great overlap between the current and previously taken image, resulting in a bad loop closure. This loop closure is seen in figure 5.4a. From this figure, one see several issues with the LC algorithm. There are no criteria on how matched features should be distributed. As the thick wire connecting the two net pen planes is high in unique features that can be matched between the images, these are used. However, as all of these points lie on a line, this will cause an incorrect projective transformation between the two images, which is evident when comparing the bad LC with the good LC shown in 5.4b. A bad projective transformation will again be propagated into the LC-factor, which is undesirable.

Another point to note is that the algae are not matched as well as hoped. This is evident from both the good and the bad LC candidates, and there are several of the high content algae areas that have not been used. This is partly because of the feature filtering caused by Lowe's algorithm but can also be the result of movement in the algae and the net, causing the SIFT feature of a particular alga to change. It

would therefore be desirable with a feature detector that better maps the algae than the SIFT. A possible solution for solving bad algae pattern detection would be to downsize the image, forcing the SIFT algorithm to be more general and detect larger features. However, image downsizing was not tested in this work.

Another issue with the LC algorithm is the positional difference between the DVL and the camera. The camera is placed on the ROV's starboard, while the DVL is placed at the port. This is problematic when the ROV does loop closure in a corner, as the DVL will approximate the plane on the left side, while the camera will picture more features from the right-side plane. This can cause that both distance estimate and the normal vector estimate obtained from the DVL are not related to the picture, which will cause corruption in the translational and rotational estimate of the homography decomposition. The synchronization issue between the camera and the rest of the sensors adds to this problem.

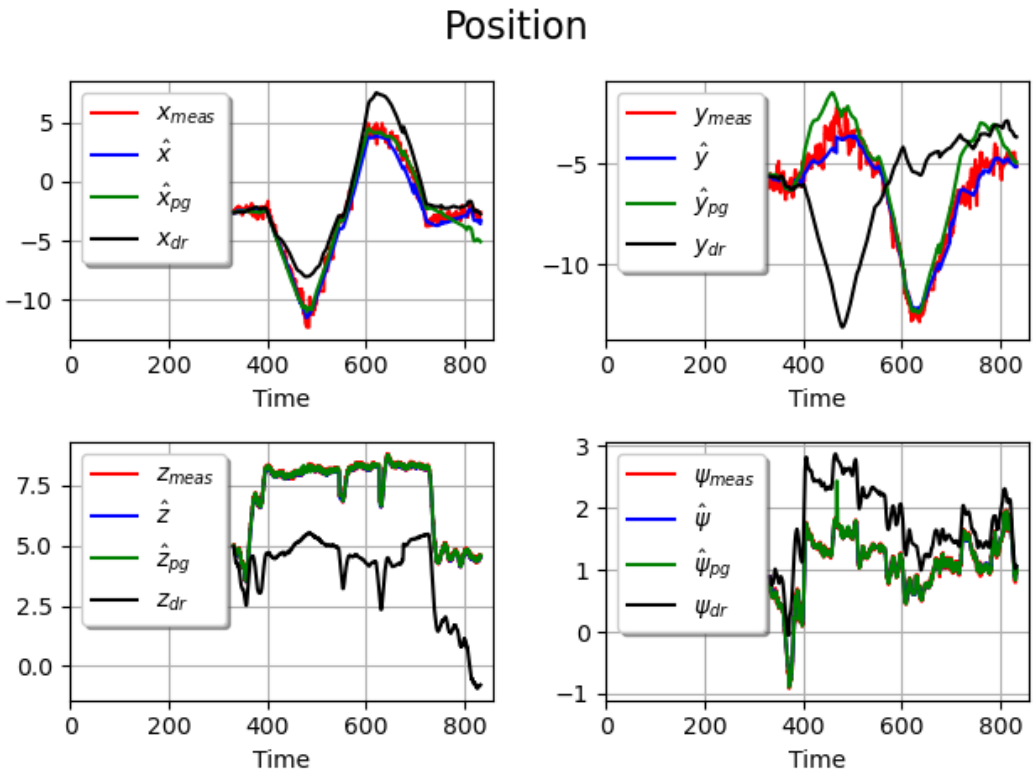
There was conducted four experiments when running the algorithm on dataset I:

- Run algorithm with all factors
- Run algorithm with all factors except visual LC-factor
- Run algorithm with all factors except tilt factor. Assuming stability in pitch and roll
- Run algorithm with all factors except tilt factor and visual LC-factor. Assuming stability in pitch and roll

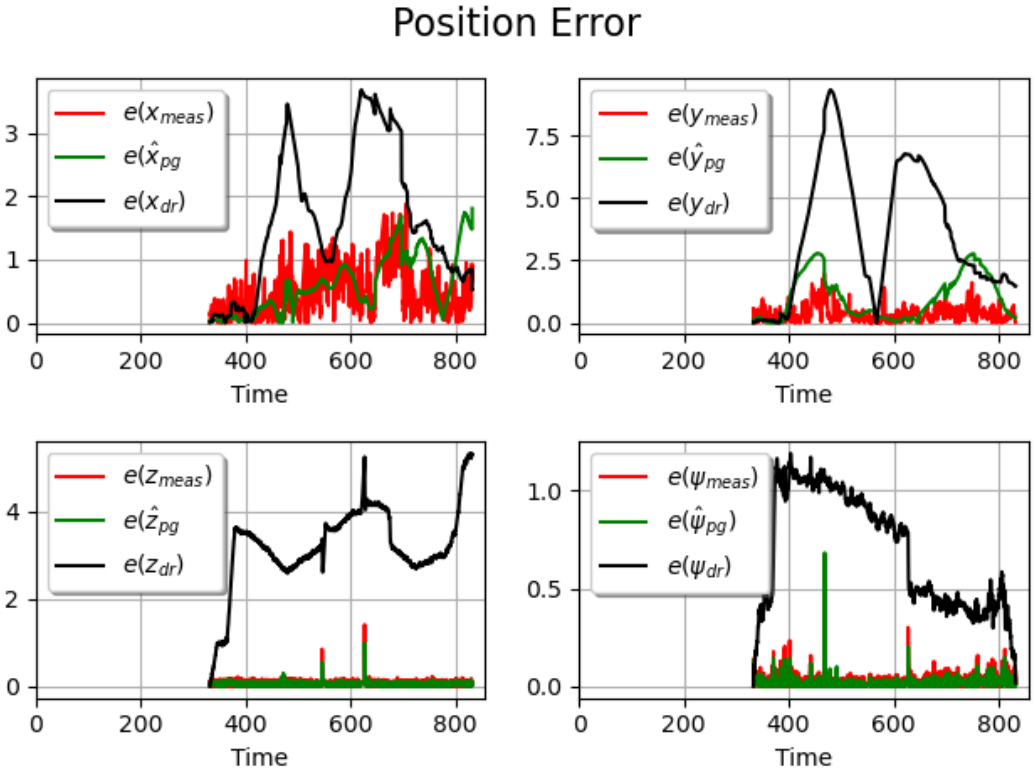
The respective plots for these runs are shown in figures 5.5, 5.6, 5.7, and 5.8. The plots are generated from running the pose SLAM algorithm, created in this thesis, which can be found on GitHub [36]. Here figure (A), for all plots, shows the position estimates for $[x, y, z, \psi]$, while figure (B) shows the position error of $[x, y, z, \psi]$ with respect to the 4DOF-EKF estimate. In these figures the red line is the actual measurement, blue is the 4DOF-EKF estimate, green is the pose-SLAM estimate and the black line is the dead reckoning estimate (DR). When comparing the results from running the algorithm with all factors 5.5 with the one where the LC-factor has been excluded 5.6, these obtain almost identical estimates for the depth and yaw measurements with respect to the 4DOF-EKF estimates. This is, however, expected as these estimates stem from accurate absolute measurements. When looking at the x-y-estimates (north and east respectively), however, the error in x are similar overall, while the y-error is better for the estimate with visual LC. Even though this is a desirable result, it is not fully known why this is the case. It should be mentioned that it is difficult to determine the quality of each estimate as areal GT is not available. However, one possible reason this might be the case is that the visual factors can stabilize the roll and pitch estimates. This is an advantage as roll and pitch are noisy measurements, and bad estimates of them would corrupt the ROV course. It could also be from the fact that the EKF-assumes that roll and pitch are zero, which causes wrong positional estimates of the EKF due to this simplification. Most likely, it is a combination of both. Another point to note here is that the loop closure occurs at time 500s, and it seems to be constraining the error when comparing the two.

In addition to comparing w/wo LC, it was also interesting to see the effect of removing the tilt factor and assume that the ROV was stabilized in pitch and roll as this is what is assumed for the EKF. The result of this and having no LC is seen in figure 5.8,

and here there is almost no positional error, only some deviation in yaw. This is an expected result as the pose graph would only be a single chain, where the optimization of each node is only affected by the former and latter nodes in addition to the absolute measurements. This is similar to the EKF as this incrementally marginalizes pose estimate based on its prior and the new measurement. However, this result shows that the pose graph is implemented correctly. When looking at figure 5.7 one can see that one obtain almost the same overall error with respect to the EKF estimate as one did when using a 6DOF model, including the tilt measurements.

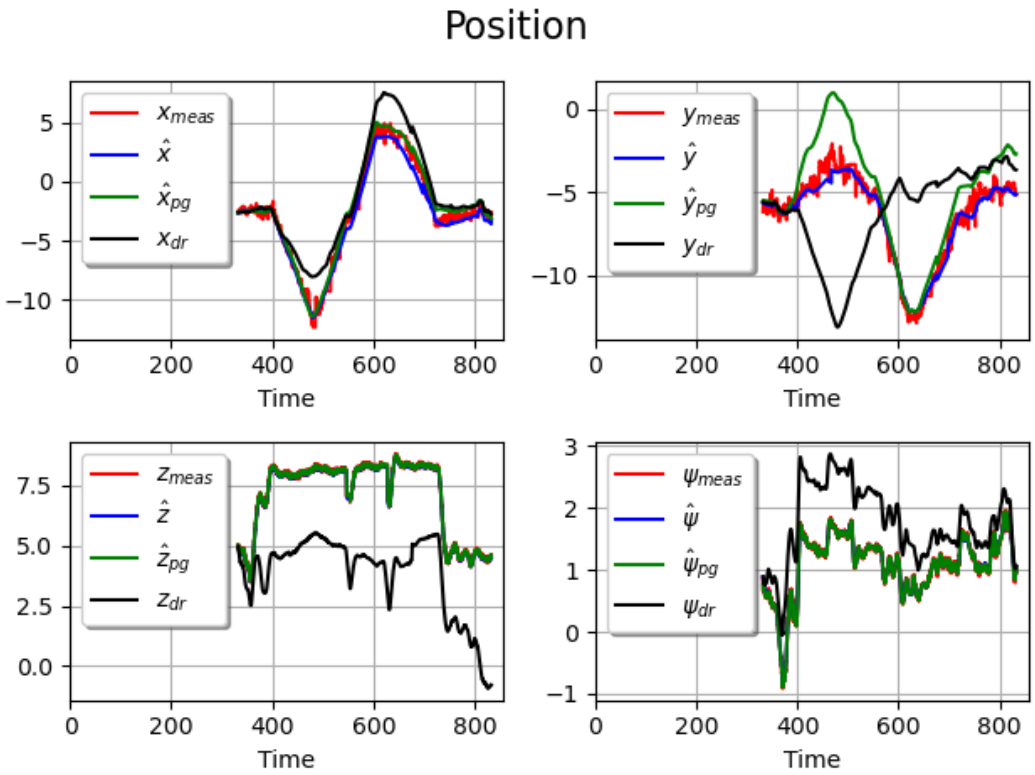


(A)

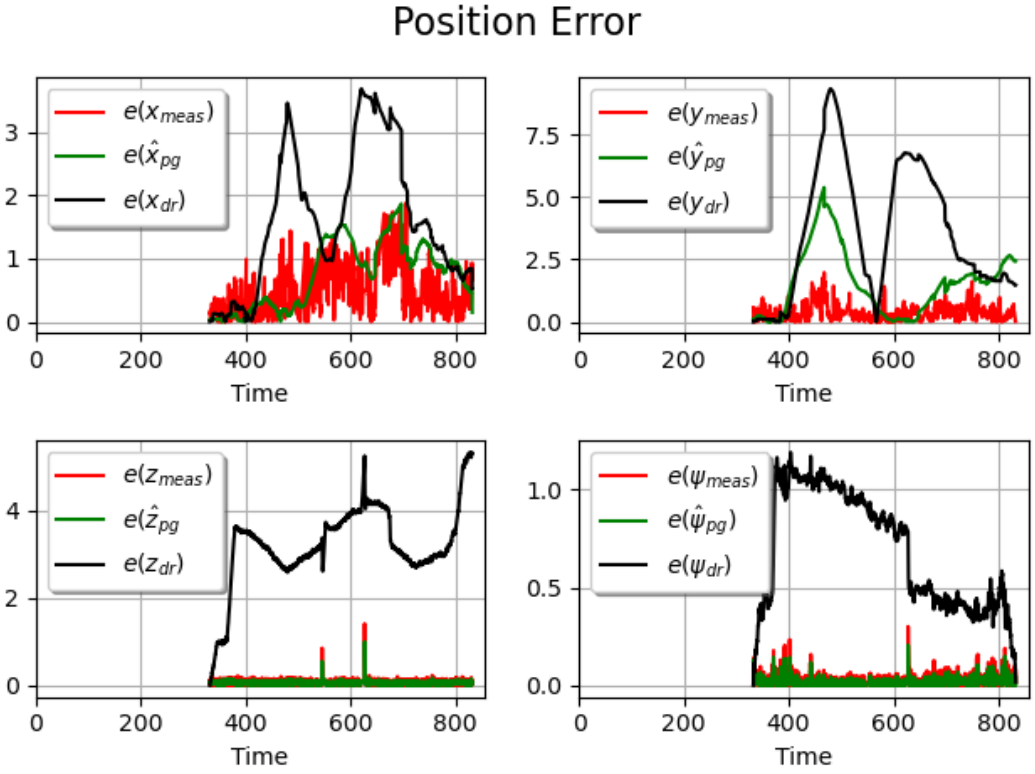


(B)

FIGURE 5.5: Dataset I - Position estimate (A) and position error with respect to EKF-estimate (B) with visual loop closure. The red, blue, green, and black lines represent actual sensor measurements, EKF estimate, pose graph estimate, and dead reckoning, respectively.

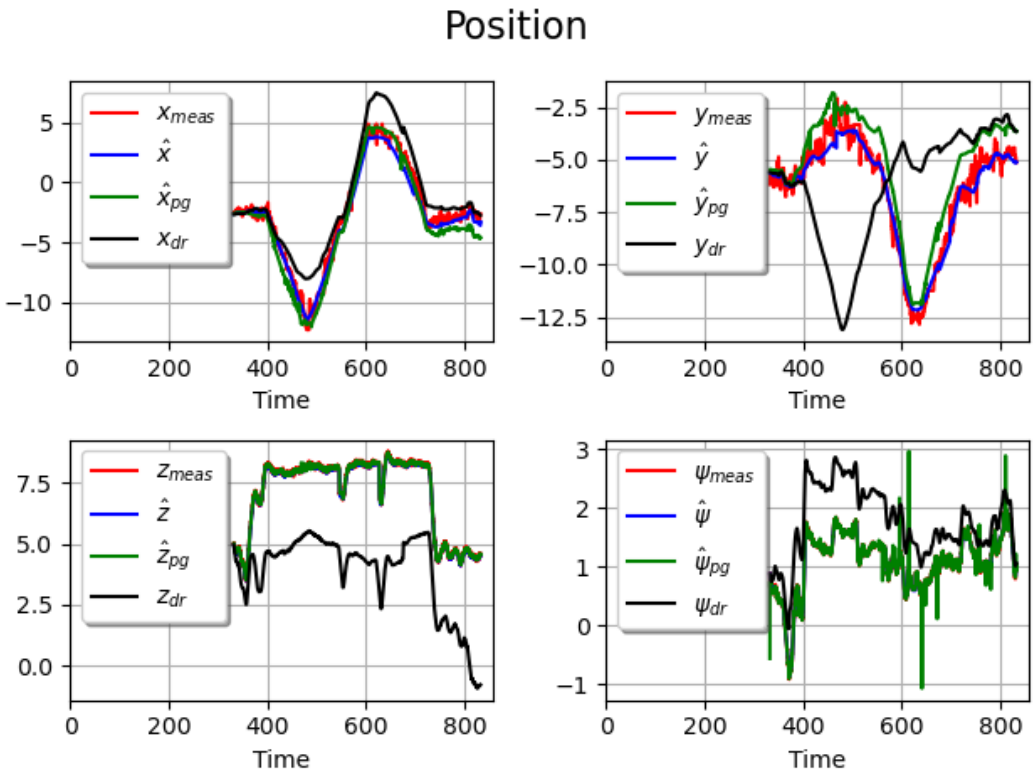


(A)

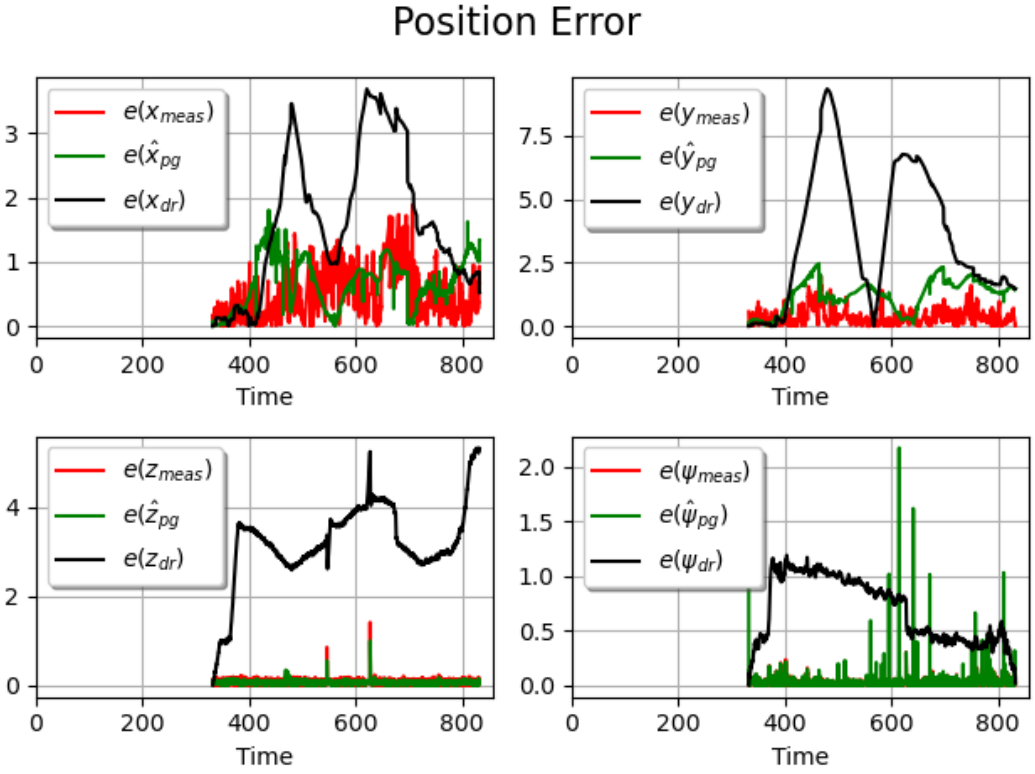


(B)

FIGURE 5.6: Dataset I - Position estimate (A) and position error with respect to EKF-estimate (B) without visual loop closure. The red, blue, green, and black lines represent actual sensor measurements, EKF estimate, pose graph estimate, and dead reckoning, respectively.

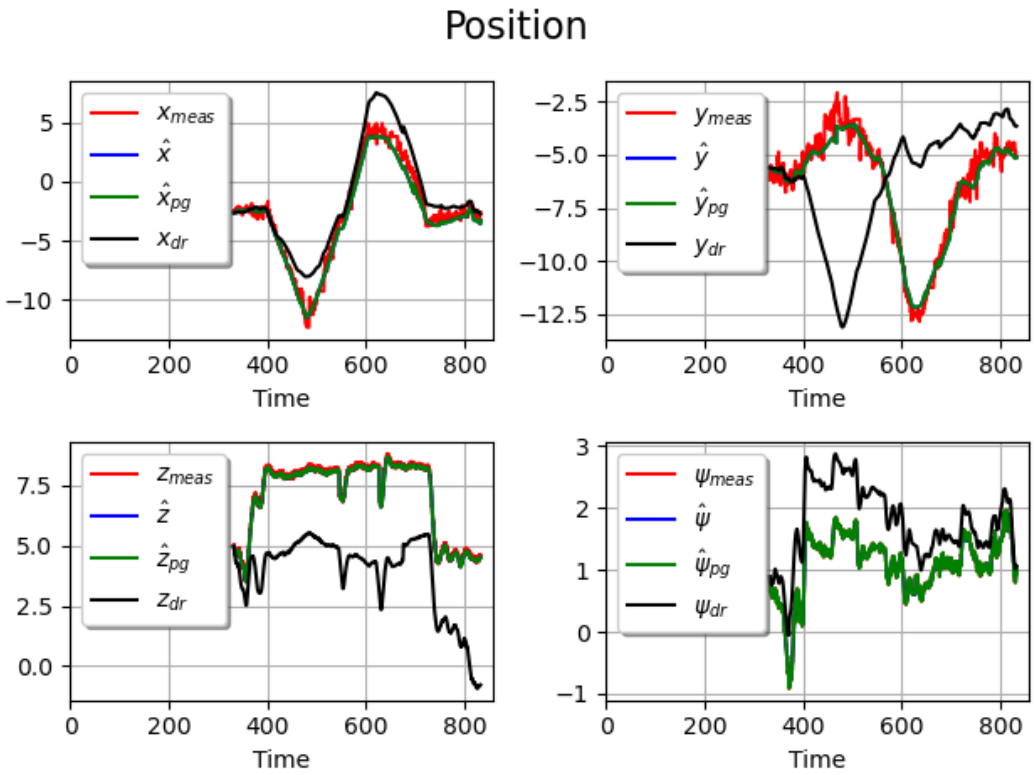


(A)

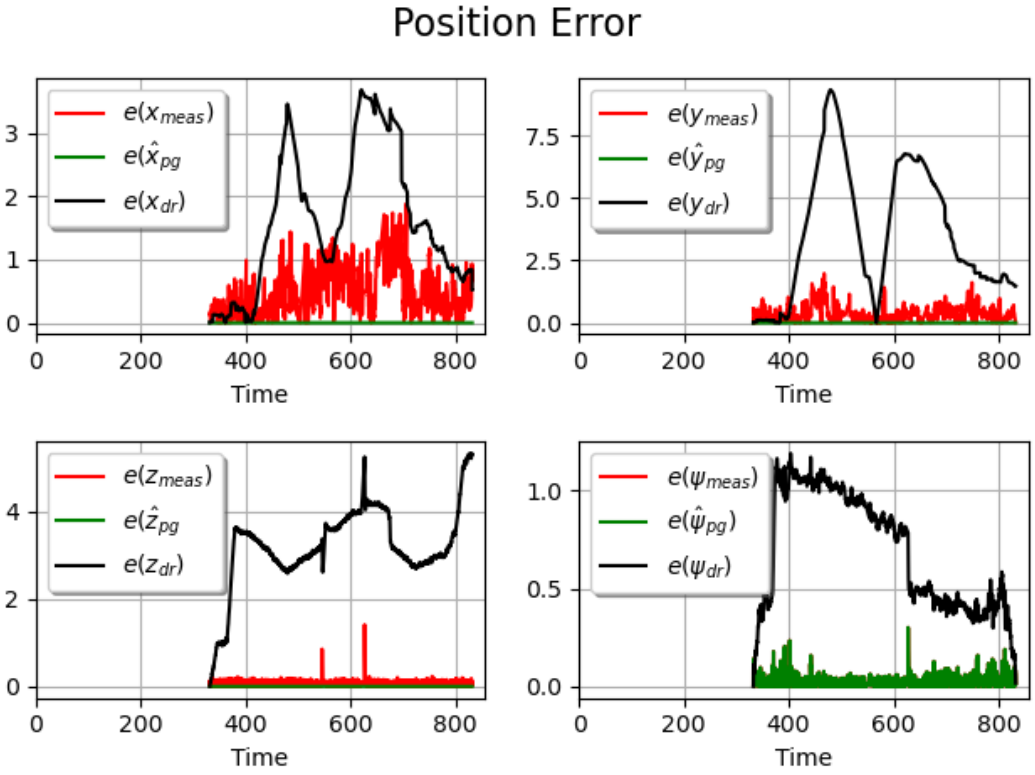


(B)

FIGURE 5.7: Dataset I - Position estimate (A) and position error with respect to EKF-estimate (B) with visual loop closure. Pitch and roll set to zero. The red, blue, green, and black lines represent actual sensor measurements, EKF estimate, pose graph estimate, and dead reckoning, respectively.



(A)

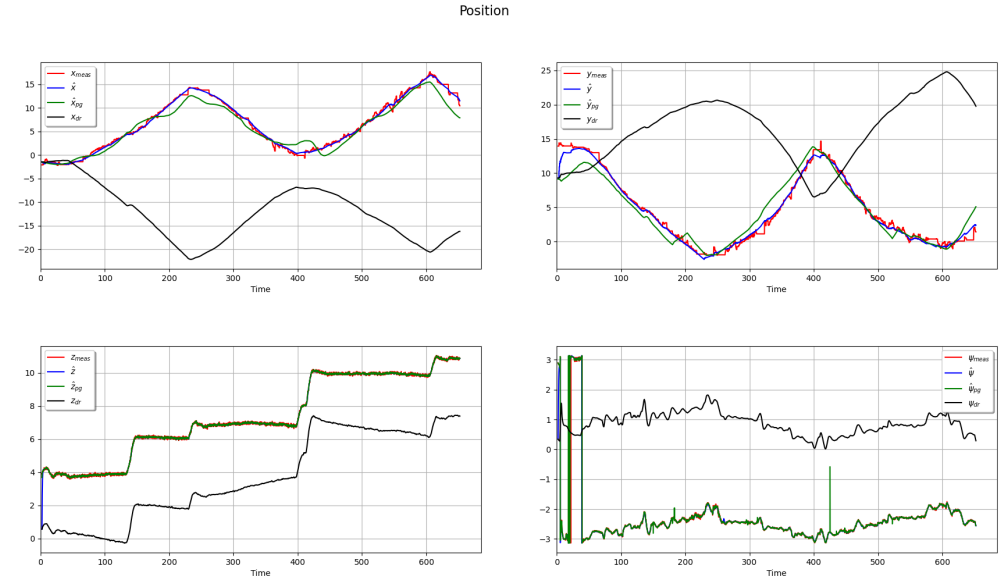


(B)

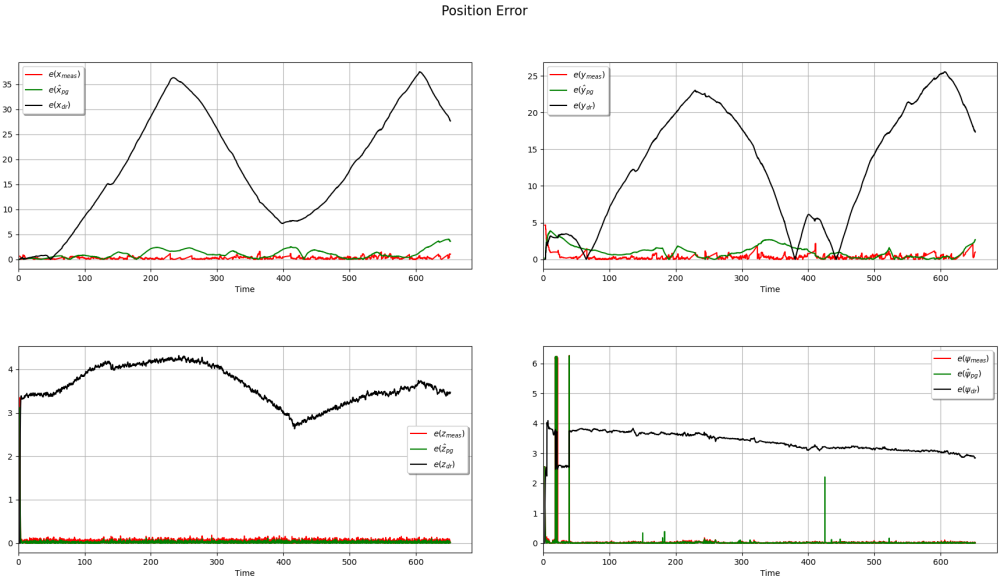
FIGURE 5.8: Dataset I - Position estimate (A) and position error with respect to EKF-estimate (B) without visual loop closure. Pitch and roll set to zero. The red, blue, green, and black lines represent actual sensor measurements, EKF estimate, pose graph estimate, and dead reckoning, respectively.

5.2 Results Dataset II - No Loop Closure

In figure 5.9 one can see the results from running the pose-SLAM algorithm on dataset II. This is a dataset with no loop closure, and the result from this is almost identical to the pose estimate produced by the EKF despite being a six-DOF model. The result further concludes the correct implementation of the pose graph.



(A)



(B)

FIGURE 5.9: Dataset II - Position estimate (A) and position error with respect to EKF-estimate (B) with visual loop closure. The red, blue, green, and black lines represent actual sensor measurements, EKF estimate, pose graph estimate, and dead reckoning, respectively.

5.3 Results Dataset III - Loop Closure at the Net Bottom

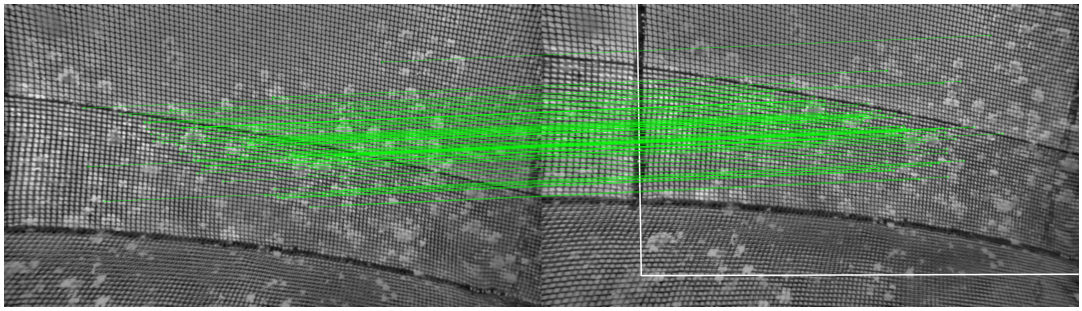
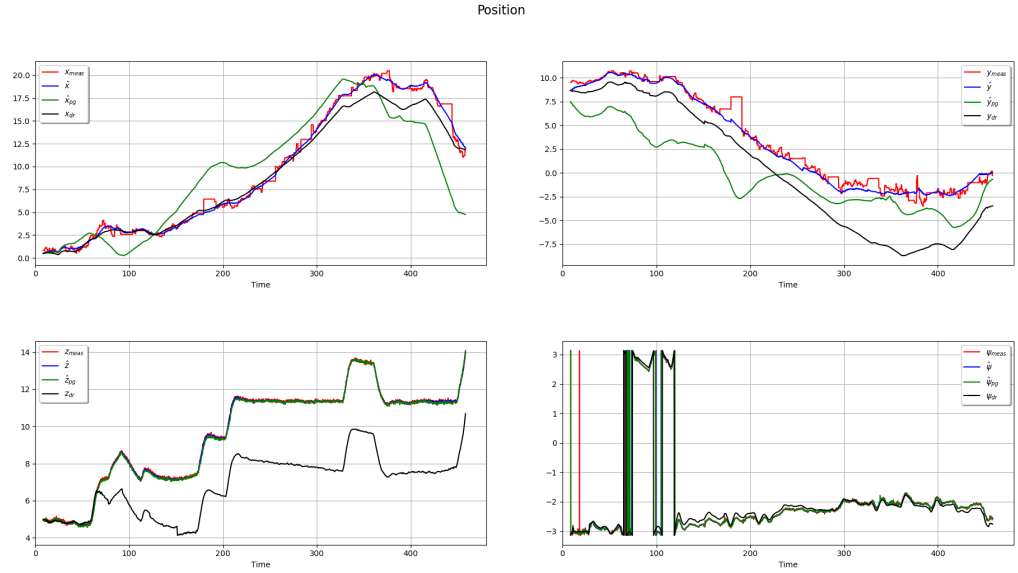


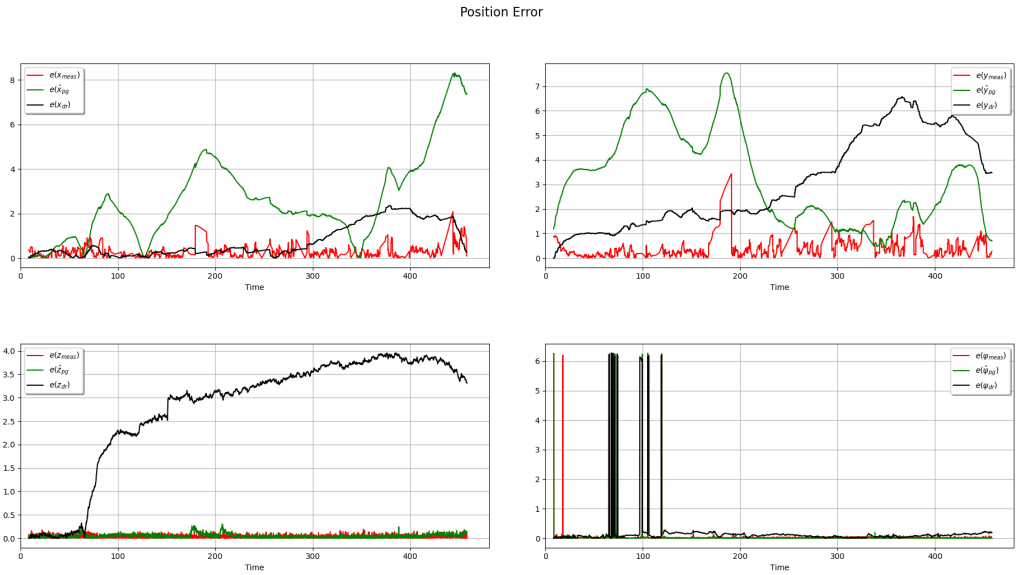
FIGURE 5.10: One of several consecutive LC causing the system to become corrupted

The results of running the code on dataset III is shown in figure 5.11 and 5.12, where the first figure is generated by running the algorithm with all factors while the second excludes the visual LC-factor. When looking at the run without visual LC one can see that there is little to no error regarding the pose SLAM estimate. There are some deviation between the EKF estimate and the pose SLAM estimate in the north direction at the end of the run, but this does not exceed 1.5m, which is approximately the same value used for the x,y STD of the USBL measurement.

Looking at the pose SLAM estimate with LC, it is evident that this fails. The loop closures causing the system to become corrupted happen at about 350s into the operation when the ROV travels at the bottom of the net pen. Here the ROV, travels slowly in the area, which allows for loop closures of pictures taken close in time. The first few LCs do not affect the system that much, but after more are added, the system collapses. The LC causing the collapse is seen in figure 5.10, and in figure 5.13, one can see the position estimate before and after this LC-factor is added to the graph. The LCs taken before the one shown in 5.10 are similar to this one, and looking at it, one can verify that it is from the same area, indicating that there is no issue related to the filtering algorithm. However, the pictures are taken in an area where the net goes from a polygon shape into a cone-shaped structure. This is problematic as the planar net assumption is violated in this region, which both the homography and the DVL plane estimation rely on. As one relies on the normal vector of the DVL plane to determine which of the homography decompositions is true decomposition, this can cause that the wrong solution is selected. Obtaining a bad visual factor could also be the result of DVL outliers. Even though the SINTEF in-house algorithm filters DVL outliers, there are still outliers that occur. Therefore, a better approach for solving this would be to use a sliding window of the DVL measurements to determine the plane, similar to the one explained in plane estimation described in section 3.4. It is not clear why this LC causes all the previous estimates to change.It causes major pose changes from 0-300s. However, this region is not within the pose graph chain connected by the LC factors, and these poses should therefore be constrained mainly by their measurements and priors. This detrimental effect needs to be researched more in the future.

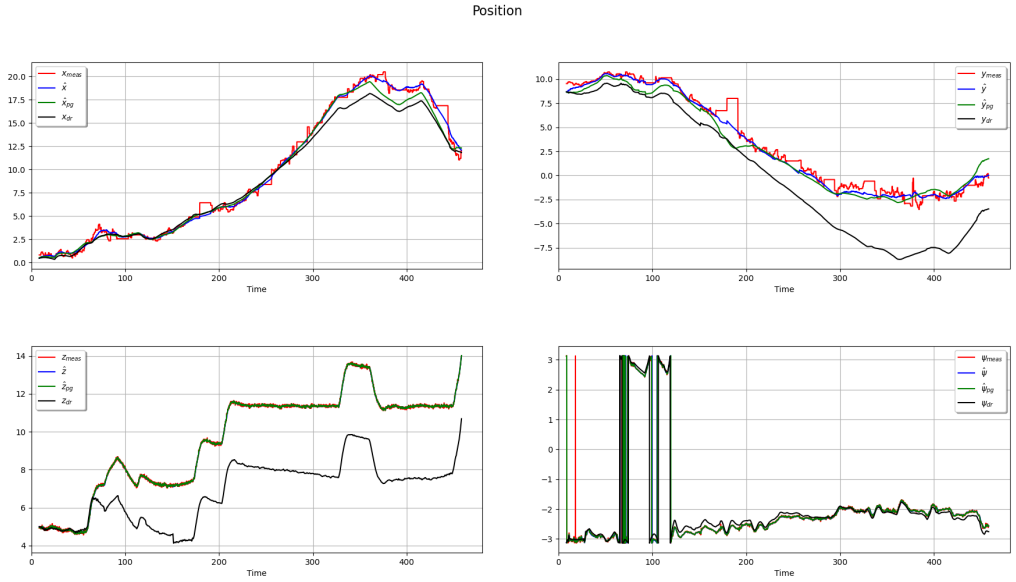


(A)

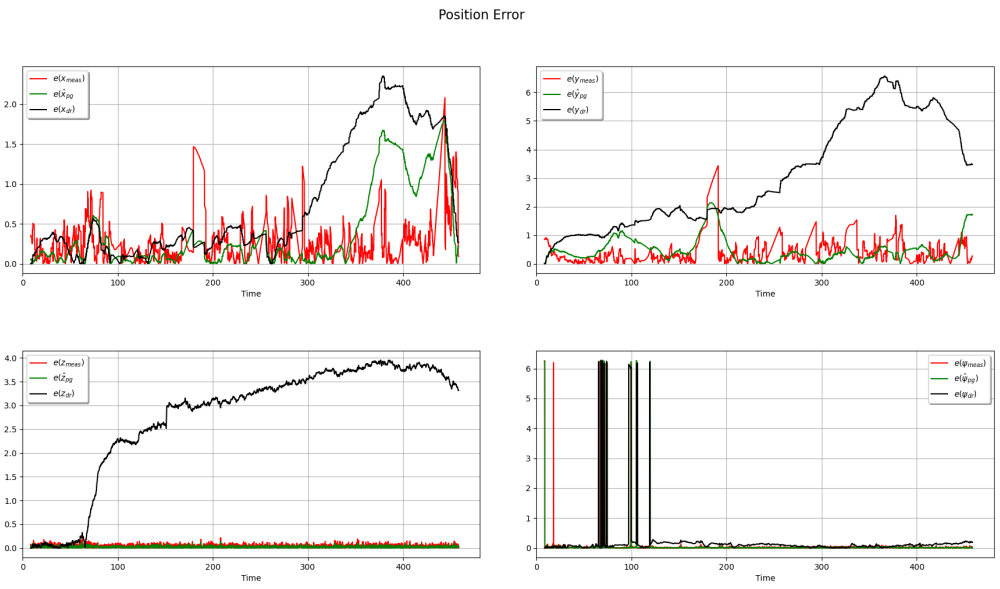


(B)

FIGURE 5.11: Dataset III - Position estimate (A) and position error with respect to EKF-estimate (B) with visual loop closure. The red, blue, green, and black lines represent actual sensor measurements, EKF estimate, pose graph estimate, and dead reckoning, respectively.

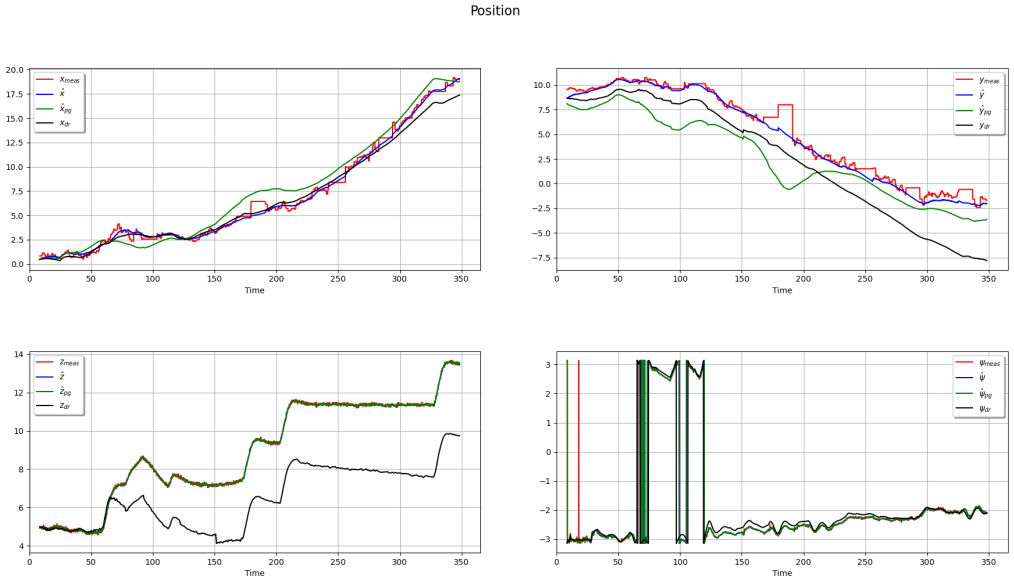


(A)

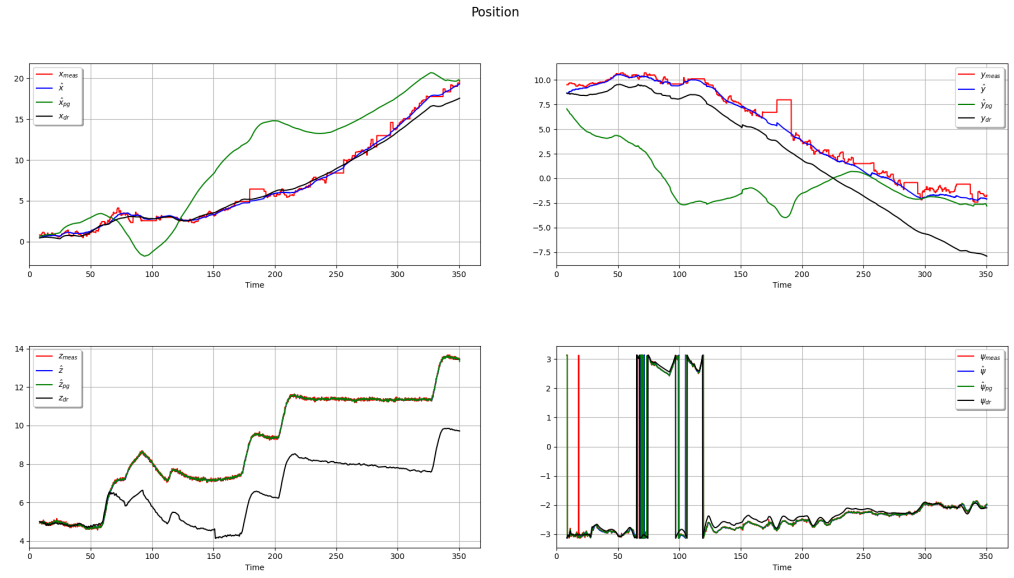


(B)

FIGURE 5.12: Dataset III - Position estimate (A) and position error with respect to EKF-estimate (B) without visual loop closure. The red, blue, green, and black lines represent actual sensor measurements, EKF estimate, pose graph estimate, and dead reckoning, respectively.



(A) Before



(B) After

FIGURE 5.13: Position error with respect to EKF-estimate before (A) and after (B) consecutive LC causing the system to become corrupted. The red, blue, green, and black lines represent actual sensor measurements, EKF estimate, pose graph estimate, and dead reckoning, respectively.

5.4 Discussion of Bag-of-Words Algorithm

One issue regarding this work that is not detectable by looking at the travel path of the ROV is the BOW algorithm. Due to time limitations, the algorithm was implemented naively, and it is neither optimal for real-time operations nor for obtaining a good vocabulary. Firstly, the vocabulary is created offline by using k-means, which requires the user to define the number of words the vocabulary should consist of. Setting a fixed value to the vocabulary size is difficult, as each run would be different, and one cannot know how many words are actually present in each dataset. This is especially problematic if one observes unique words. These words risk being generalized to more common words when using a fixed vocabulary, affecting the local and global saliency measures.

The tree structure used in this work is neither an ideal candidate for how to structure the words. Firstly, it is not optimal for a real-time system as the tree is not balanced, leading to higher run-time when labeling features. Secondly, when generating the tree, the k-means method has been used to divide the tree into its respective branches. This starts at the root, divides the tree into two clusters for the left and right branches, and continues this recursively for each branch until all leaves only contain one word. As the k-means will not provide the correct cluster solution every time as the algorithm depends on its seeds, there is a possibility that some words are put in an incorrect branch. This incorrect branching will, in turn, lead to false positives when it comes to word labeling. However, this binary tree generation's effect on the ratio between false-positives and true-positives has not been studied in this work.

Chapter 6

Conclusion & Further Work

The problem statement of this thesis was to create a functional six-degrees-of-freedom SLAM framework for the fish cage environment that was able to fuse all the sensor measurements of the ROV. It was also desired to create a method of solving the loop closure problem in this environment. This has been accomplished by creating a pose graph SLAM algorithm, formulating the SLAM problem as a non-linear optimization problem, and solving this by using the iSAM2 framework. The loop closure problem was solved by creating a novel data association filtering method that proposed salient image loop closure candidates captured at similar depth and heading. If an image loop closure was found, the correct homography decomposition was determined and scaled using the DVL range measurements to approximate the net structure in front of the ROV as a plane. The algorithm shows promising results when the ROV operates in regions of the net that do not violate the planar assumption; however, further development is required to obtain more reliable and consistent pose measurements.

One part of the algorithm that needs more research is the visual factor. Even though the ROV revisited areas in these datasets, the only LC that was achieved when running the algorithm was LC of images close in time. This is because the ROV was not controlled such that it revisited scenes containing an abundance of features. This needs to be kept in mind if one desires to continue this SLAM approach in the fish cage and must be accounted for in a procedure of how to traverse the net effectively. One point to note here is that the fish cage needs to be traversed slowly capture the scene and avoid image distortion caused by motion. Slow traversing will also be beneficial due to better overlap between images.

There also needs to be done improvements towards the loop closure algorithm. As one can see from some of the images illustrated in this report, one can see that the SIFT features used are not adequate to capture the algae detection of the scene, and it could be desired to replace this feature detector with another solution. Secondly, DVL measurements should be put into a sliding window framework, similar to the one in [39], such that one could have more measurements to estimate the plane in front of the ROV. Using a sliding window would also be beneficial as RANSAC can be used to remove DVL outliers. Another point that needs to be looked more into is how to handle loop closures better at the bottom of the fish cage. Of these, obtaining a procedure for handling the bottom of the fish cage is the most crucial as seen in the results of dataset III illustrated in section 5.3, as this scene violated the plane assumption used in this work.

More research should also be done in system tuning and developing an actual GT used for this tuning. There also needs to be obtained a more precise timestamp when camera recordings were taken, enabling better synchronization between the camera

and the other sensor data.

A final area that needs to be put more effort into solving is reducing the algorithm's run-time. This code is mainly created in python, but transferring this work to C++would significantly improve run-time. The BOW vocabulary should also be structured in a better framework, either by using a balanced tree or trying to implement hash tables to decode the features into words [43]. In addition, it would be desirable to use an online approach for building the vocabulary, as one would this way avoid issues related to insufficient vocabulary size.

Bibliography

[1] Alex M Andrew. "Multiple view geometry in computer vision". In: *Kybernetes* (2001).

[2] Herman B. Amundsen, Walter Charija, and Kristin Y. Pettersen. "Autonomous ROV inspections of aquaculture net pens using DVL". under revision. 2021.

[3] Randal W Beard and Timothy W McLain. *Small unmanned aircraft: Theory and practice*. Princeton university press, 2012.

[4] Magnus Bjerkeng et al. "ROV Navigation in a Fish Cage with Laser-Camera Triangulation". In: *Journal of Marine Science and Engineering* 9.1 (2021), p. 79.

[5] Edmund Brekke. *Fundamentals of Sensor Fusion*. 2019, pp. 1–276.

[6] Cesar Cadena et al. "Past, present, and future of simultaneous localization and mapping: Toward the robust-perception age". In: *IEEE Transactions on robotics* 32.6 (2016), pp. 1309–1332.

[7] *Camera Calibration and 3D Reconstruction*. URL: https://docs.opencv.org/master/d9/d0c/group__calib3d.html. (accessed: 11.06.2021).

[8] Thomas H Cormen et al. *Introduction to algorithms*. MIT press, 2009.

[9] Frank Dellaert, Michael Kaess, et al. "Factor graphs for robot perception". In: *Foundations and Trends® in Robotics* 6.1-2 (2017), pp. 1–139.

[10] Fredrik Dukan and Asgeir J Sørensen. "Sea floor geometry approximation and altitude control of ROVs". In: *Control Engineering Practice* 29 (2014), pp. 135–146.

[11] Hugh Durrant-Whyte and Tim Bailey. "Simultaneous localization and mapping: part I". In: *IEEE robotics & automation magazine* 13.2 (2006), pp. 99–110.

[12] CH Edwards and David E Penney. "Elementary Linear Algebra. 1988". In: *Prince-Hall, Englewood Cliffs* ().

[13] Ryan Eustice et al. "UWIT: Underwater Image Toolbox for optical image processing and mosaicking in MATLAB". In: *Proceedings of the 2002 International Symposium on Underwater Technology (Cat. No. 02EX556)*. IEEE. 2002, pp. 141–145.

[14] Ryan M Eustice, Oscar Pizarro, and Hanumant Singh. "Visually augmented navigation for autonomous underwater vehicles". In: *IEEE Journal of Oceanic Engineering* 33.2 (2008), pp. 103–122.

[15] Kenneth G Foote. "Importance of the swimbladder in acoustic scattering by fish: a comparison of gadoid and mackerel target strengths". In: *The Journal of the Acoustical Society of America* 67.6 (1980), pp. 2084–2089.

[16] THOR FOSSEN, Kristin Y Pettersen, and Henk Nijmeijer. *Sensing and control for autonomous vehicles*. Springer, 2017, pp. 143–160.

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

[18] Trym Vegard Haavardsholm. "A handbook in Visual SLAM". draft. 2021.

[19] Robert M Haralick. "Propagating covariance in computer vision". In: *International journal of pattern recognition and artificial intelligence* 10.05 (1996), pp. 561–572.

[20] Marianne Holmer. "Environmental issues of fish farming in offshore waters: perspectives, concerns and research needs". In: *Aquaculture Environment Interactions* 1.1 (2010), pp. 57–70.

[21] *Introduction to SIFT (Scale-Invariant Feature Transform)*. URL: https://docs.opencv.org/master/da/df5/tutorial_py_sift_intro.html. (accessed: 06.04.2021).

[22] Itseez. *Open Source Computer Vision Library*. https://github.com/itseez/opencv. 2015.

[23] Michael Kaess, Ananth Ranganathan, and Frank Dellaert. "iSAM: Incremental smoothing and mapping". In: *IEEE Transactions on Robotics* 24.6 (2008), pp. 1365–1378.

[24] Michael Kaess et al. "iSAM2: Incremental smoothing and mapping using the Bayes tree". In: *The International Journal of Robotics Research* 31.2 (2012), pp. 216–235.

[25] Aram Kawewong et al. "Online and incremental appearance-based SLAM in highly dynamic environments". In: *The International Journal of Robotics Research* 30.1 (2011), pp. 33–55.

[26] Ayoung Kim and Ryan Eustice. "Pose-graph visual SLAM with geometric model selection for autonomous underwater ship hull inspection". In: *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2009, pp. 1559–1565.

[27] Ayoung Kim and Ryan M Eustice. "Real-time visual SLAM for autonomous underwater hull inspection using visual saliency". In: *IEEE Transactions on Robotics* 29.3 (2013), pp. 719–733.

[28] James C Kinsey, Ryan M Eustice, and Louis L Whitcomb. "A survey of underwater vehicle navigation: Recent advances and new challenges". In: *IFAC Conference of Manoeuvering and Control of Marine Craft*. Vol. 88. Lisbon. 2006, pp. 1–12.

[29] Pascal Klebert et al. "Three-dimensional deformation of a large circular flexible sea cage in high currents: Field experiment and modeling". In: *Ocean Engineering* 104 (2015), pp. 511–520.

[30] Don Koks. "On the Use of Vectors, Reference Frames, and Coordinate Systems in Aerospace Analysis". In: (2017).

[31] Pål Lader et al. "Current induced net deformations in full-scale sea-cages for Atlantic salmon (Salmo salar)". In: *Aquacultural Engineering* 38.1 (2008), pp. 52–65.

[32] Jie Li et al. "Pose-graph SLAM using forward-looking sonar". In: *IEEE Robotics and Automation Letters* 3.3 (2018), pp. 2330–2337.

[33] David G Lowe. "Distinctive image features from scale-invariant keypoints". In: *International journal of computer vision* 60.2 (2004), pp. 91–110.

[34] Prasanta Chandra Mahalanobis. "On the generalized distance in statistics". In: National Institute of Science of India. 1936.

[35] Ezio Malis and Manuel Vargas. "Deeper understanding of the homography decomposition for vision-based control". PhD thesis. INRIA, 2007.

[36] *Master Thesis Code*. URL: https://github.com/KyrreHaugland/Master-Thesis-Underwater-Pose-Graph-SLAM-in-Aquaculture (visited on 06/29/2021).

[37] H Moe, A Fredheim, and OS Hopperstad. "Structural analysis of aquaculture net cages in current". In: *Journal of Fluids and Structures* 26.3 (2010), pp. 503–516.

[38] José Neira and Juan D Tardós. "Data association in stochastic mapping using the joint compatibility test". In: *IEEE Transactions on robotics and automation* 17.6 (2001), pp. 890–897.

[39] Paul Ozog and Ryan M Eustice. "Real-time SLAM with piecewise-planar surface models and sparse 3D point clouds". In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2013, pp. 1042–1049.

[40] Per Rundtop and Kevin Frank. "Experimental evaluation of hydroacoustic instruments for ROV navigation along aquaculture net pens". In: *Aquacultural Engineering* 74 (2016), pp. 143–156.

[41] Salmar. *Offshore Fish Farming*. 2016. URL: https://www.salmar.no/en/offshore-fish-farming-a-new-era/.

[42] Lorenzo Sciavicco and Bruno Siciliano. *Modelling and control of robot manipulators*. Springer Science & Business Media, 2012.

[43] Hossein Shahbazi and Hong Zhang. "Application of locality sensitive hashing to realtime loop closure detection". In: *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE. 2011, pp. 1228–1233.

[44] Even Skjellaug. "Feature-Based Lidar SLAM for Autonomous Surface Vehicles Operating in Urban Environments". In: (2020).

[45] *The OpenCV Reference Manual*. 3.4.15. Itseez. 2021.

[46] Trine Thorvaldsen, Ingunn M Holmen, and Helene K Moe. "The escape of fish from Norwegian fish farms: Causes, risks and the influence of organisational aspects". In: *Marine Policy* 55 (2015), pp. 33–38.

[47] Olga Vysotska and Cyrill Stachniss. *Bag of visual words*. https://github.com/ovysotska/in_simple_english/blob/master/bag_of_visual_words.ipynb. 2020.

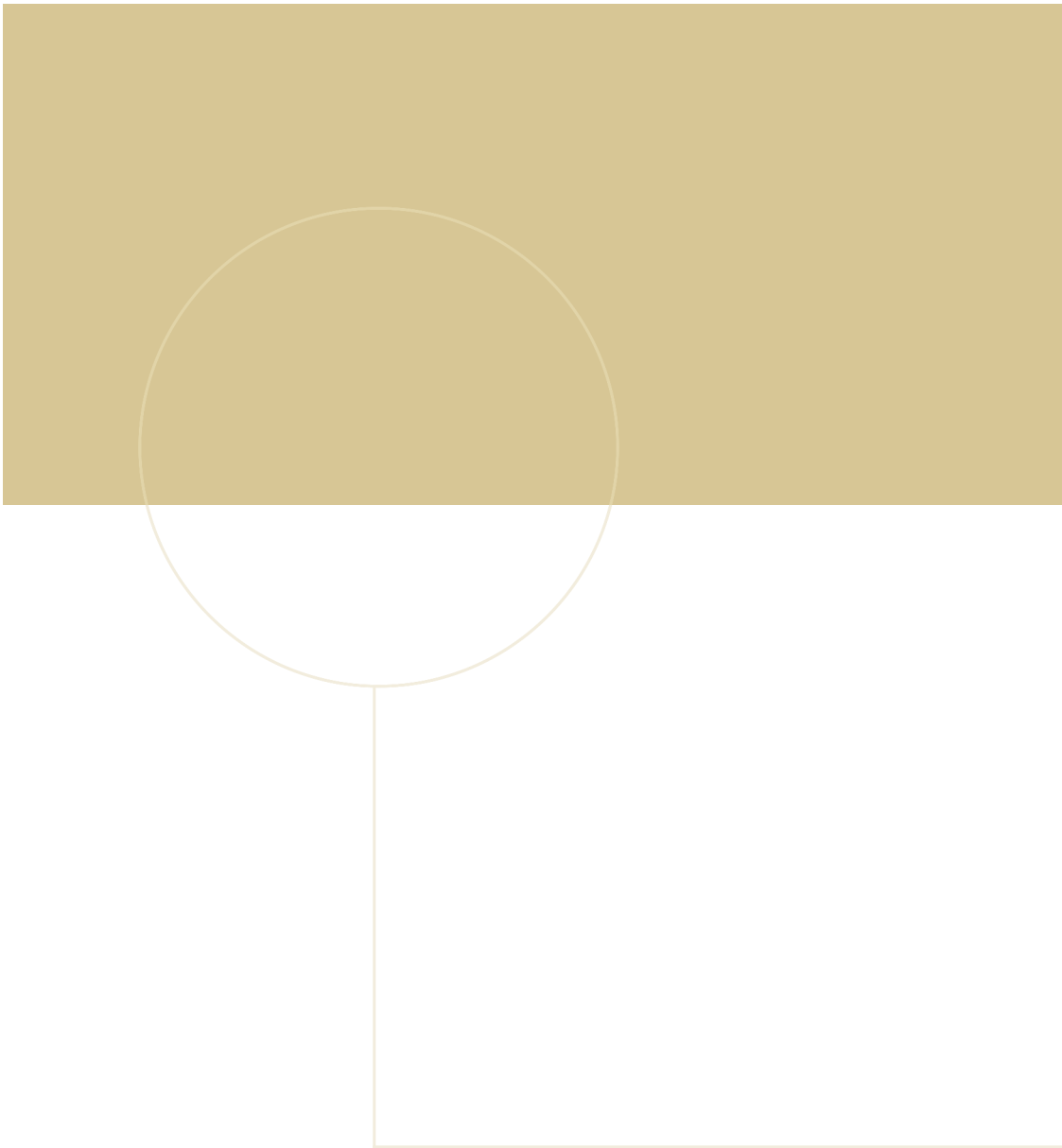
[48] R YANG et al. "Robust Control Application To Ciscrea Underwater Vehicle". In: ().

[49] Zhongfei Zhang and Allen R Hanson. "3D reconstruction based on homography mapping". In: *Proc. ARPA96* (1996), pp. 1007–1012.

Appendix A

Pose Graph SLAM Code

The code created in this work can be found on GitHub [36]. The code is mainly written in python. However, it also utilizes a MATLAB implementation of the CLAHS algorithm, provided by Ryan Eustice, and the inclinometer-, depth- and USBL factor are written in C++.



NTNU
Norwegian University of
Science and Technology

SINTEF